Ansätze kompositionaler und zustandsbasierter Zugriffskontrolle für Web-basierte Umgebungen

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von
Sandra Wortmann

Dortmund
2011

Tag der mündlichen Prüfung: 4. Juli 2011
Dekan: Prof. Dr. Peter Buchholz
Erstgutachter: Prof. Dr. Joachim Biskup
Zweitgutachter: Prof. Dr. Heiko Krumm

Zusammenfassung

Moderne verteilte Rechensysteme müssen flexibel an wechselnde Rahmenbedingungen und Aufgabenstellungen angepasst werden können. Notwendig hierfür ist, dass diese Rechensysteme in dynamisch veränderlicher Struktur aus verschiedenen informationellen Diensten zusammengesetzt sind. Kompositionalität ist in diesem Kontext eine wünschenswerte Eigenschaft, sowohl der Rechensysteme als auch der den Diensten zugeordneten Zugriffskontrollpolitiken und ihren Implementierungen. Zugriffskontrollpolitiken drücken hier aus, welche Dienste welchen Teilnehmern unter welchen Bedingungen verfügbar sein sollen. Bei anspruchsvollen Anwendungen – wie beispielsweise *strukturierten Diensten* – müssen die Zugriffskontrollpolitiken nicht nur für einzelne, atomare Funktionalitäten der Dienste festgelegt werden, sondern auch für komplexe Folgen der Funktionalitäten.
Diese Arbeit schlägt eine kompositionale und zustandsbasierte Lösung für die beschriebenen Herausforderungen vor. Es wird eine kompositionale Algebra für Zugriffskontrollpolitiken für strukturierte Dienste entwickelt. Für diese sogenannten zustandsdynamischen Zugriffskontrollpolitiken werden konzeptionelle Durchsetzungsmechanismen erarbeitet. Es werden des Weiteren zentrale und dezentrale Architekturen für Zertifikat-basierte Zugriffskontrollsysteme entworfen, in die die vorgeschlagene Lösung eingebettet werden kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Definitionsverzeichnis

Motivation

Die Zugriffskontrolle eines Rechensystems verfolgt im Wesentlichen zwei Ziele: Eine Kontrollinstanz muss entscheiden, ob ein Zugriffswunsch eines Subjekts bezüglich einer bestimmten Systemressource autorisiert ist, und die jeweilige Zugriffsentscheidung muss wirksam durchgesetzt werden. Damit eine Kontrollinstanz eines Rechensystems eine Zugriffsentscheidung treffen kann, muss die Menge aller möglichen Zugriffe innerhalb des Systems bestimmt werden. Traditionellerweise kann dies durch die Spezifizierung des Tupels (S, O, A) erfolgen, vgl. [42]. Dabei bezeichnet S die Menge aller Subjekte (beispielsweise Teilnehmer oder Prozesse), O die Menge aller von der Zugriffskontrolle geschützten atomaren Objekte und A die Menge der atomaren Aktionen, welche durch die Subjekte auf den Objekten ausgeführt werden können. In einem Rechensystem wird eine Zugriffskontrollpolitik typischerweise durch eine Menge von Erlaubnissen der Form (s, o, a) gegeben, wobei $s \in S$, $o \in O$ und $a \in A$ gilt. Eine solche Erlaubnis besagt, dass das Subjekt s autorisiert ist, die Aktion a auf dem Objekt o ohne einschränkende Bedingungen auszuführen. Dies bedeutet insbesondere, dass jede in der Zugriffskontrollpolitik vorkommende Erlaubnis unabhängig von den anderen genutzt werden kann.

Traditionelle Zugriffskontrollmodelle in zentralen Systemen [69] überprüfen die Authentizität einer Zugriffsanfrage eines Teilnehmers und treffen die Zugriffsentscheidung aufgrund zuvor vergebener Zugriffsrechte. Zwei klassische Strategien für Zugriffskontrollmodelle sind diskretionäre Zugriffskontrolle (*discretionary access control*) und mandatorische Zugriffskontrolle (*mandatory access control*), vgl. [33]. Zugriffskontrollsysteme basieren heute überwiegend auf der diskretionären Zugriffsrechtevergabe. In diesem Modell wird für jedes zu schützende Objekt im System individuell und meist lokal festgelegt, welche Aktionen einzelne Subjekte auf diesem Objekt ausführen dürfen. Dieses Modell wird häufig in Kombination mit dem so genannten Eigentümerprinzip verwendet: Der Besitzer eines Objekts darf sämtliche Erlaubnisse bezüglich seines Objekts in die Zugriffskontrollpolitik einfügen beziehungsweise löschen. Im Gegensatz hierzu werden in mandatorischen Modellen ausgehend von existierenden Hierarchiestrukturen unter den Subjekten und Objekten systemweite Regeln definiert. Beide Modelle können durch eine zentrale Administrierungsstelle oder mehrere dezentrale Administrierungsstellen verwaltet werden. Zur Effizienzsteigerung bei der Zugriffsrechtevergabe und -verwaltung haben sich Mechanismen wie rollenbasierte Zugriffskontrolle [37], Benutzer- oder Objektgruppenkonzepte etabliert. Den klassischen Zugriffskontrollmodellen gemein ist, dass sich diese am Aufbau fester Organisationseinheiten im System orientieren.

Moderne verteilte Rechensysteme [77] mit einer potenziell sehr großen An-

zahl unbekannter Teilnehmer und einer Vielfalt von Diensten erfüllen die obigen Voraussetzungen nicht mehr. So avancierte in den letzten Jahren der Begriff der *virtuellen Organisation* [58] zu einem Schlagwort. Eine virtuelle Organisation ist ein kooperatives Netzwerk von autonom agierenden und kommunizierenden Organisationseinheiten, welche ihre Dienste für einen befristeten Zeitraum zu einem losen Verbund zusammenschließen. Solche Systeme erfordern die Verwendung neuerer Zugriffskontrollmodelle, die hier festlegen, welche Teilnehmer welche Dienste in Anspruch nehmen dürfen. Für dieses heterogene Umfeld versprechen so genannte attributbasierte Zugriffskontrollansätze (*attribute-based access control*) Lösungen. Die Grundidee attributbasierter Zugriffskontrolle (vgl. [7, 15, 61, 64]) besteht darin, Zugriffsrechte für Teilnehmer nicht nur an Identitäten, sondern auch an beliebige und dynamische Merkmale von Teilnehmern zu binden. Hierdurch können Teilnehmer Zugriffsrechte pseudonym oder anonym in Anspruch nehmen.

Die bisher genannten Zugriffskontrollmodelle behandeln (vorrangig) die Vergabe und Kontrolle von Zugriffsrechten für einzelne Funktionalitäten angebotener Dienste. Einige Vorschläge zur Zugriffskontrolle einzelner Funktionalitäten im Anwendungsgebiet strukturierter Dienste (z.B. *workflow*-Systeme, *web services*) finden sich in [20, 14, 28, 48, 51]. Werden allerdings komplexe Anwendungen im Bereich virtueller Organisationen betrachtet, so müssen Zugriffskontrollpolitiken nicht nur für einzelne, atomare Funktionalitäten der Dienste festgelegt werden, sondern auch für komplexe Folgen der Funktionalitäten strukturierter Dienste. Dieses Konzept der Zustandsabhängigkeit findet sich in so genannten *history-based* oder *state-dependent* Zugriffskontrollansätzen (vgl. [11, 17, 32, 50, 52, 71]) wieder.

In dieser Arbeit werden die Konzepte zustandsbasierter Zugriffskontrollmodelle für strukturierte Dienste angewendet. Im Hinblick auf die Zugriffskontrolle ihrer zugeordneten Organisationseinheiten können administrierende Teilnehmer (*Besitzer*) explizit erlaubte Folgen spezifizieren. Als Grundkonzept einer Zugriffskontrollpolitik definieren wir $\mathcal{T} = (S \times O \times A)^*$ als die Menge der *möglichen Ausführungssequenzen* des zugrunde liegenden Rechensystems. Ein Besitzer kann von der Menge $\mathcal{T}$ der möglichen Ausführungssequenzen bestimmte auswählen, die er innerhalb seiner Organisationseinheit als *erlaubt* ansieht. Eine von einem Besitzer spezifizierte *erlaubte Ausführungssequenz* $\tau \in \mathcal{T}$ hat die Form $\tau = \varepsilon_1 \ldots \varepsilon_n$ mit $\varepsilon_i \in (S \times O \times A)$. Die Nutzung der in der Sequenz τ vorkommenden Erlaubnisse ε_i soll nur in der angegebenen Reihenfolge gewährt werden. Eine notwendige Erweiterung der zustandsbasierten Zugriffskontrollmodelle ergibt sich unmittelbar aus den möglichen Kompositionsarten strukturierter Dienste. Verschiedene Ansätze in der Literatur, welche Vorschläge für die kompositionale Konstruktion von Diensten erarbeiten (vgl.[13, 55, 76]), verdeutlichen, dass nicht nur Folgen von Funktionalitäten sondern auch (parallele) Strukturen kontrolliert werden müssen. Bisher sind unseres Wissens noch keine kompositionalen Lösungen zur Kontrolle und Überwachung der so strukturierten Dienste vorgeschlagen worden.

In dieser Arbeit wird eine **kompositionale Algebra auf zustandsdynamischen Zugriffskontrollpolitiken** entwickelt, die es einem Besitzer ermöglicht, erlaubte Ausführungssequenzen durch entsprechende Operatoren zu *erlaubten Ausführungsstrukturen* zu komponieren. Die vorgeschlagene Algebra gründet im Wesentlichen auf dem in Bonatti et al. [19] vorgestellten algebraischen Ansatz zur Komposition verschiedener Zugriffskontrollpolitiken. In diesem Ansatz können die Zugriffskontrollpolitiken von beliebigen dezentralen Administrierungsstellen autonomer Organisationseinheiten spezifiziert werden. Die Zugriffskontrollpolitiken können durch die elementaren Kompositionsoperationen Vereinigung, Durchschnitt, Differenz, Selektion und regelbasierte Ableitung komponiert werden, wobei das Kompositionsergebnis jeweils wieder eine Zugriffskontrollpolitik ist. Da der in dieser Arbeit vorgestellte Ansatz auf der Arbeit von Bonatti et al. basiert, wird hierauf in Kapitel 2 detailliert eingegangen.

Ein weiterer relevanter Vertreter einer kompositionalen Algebra auf Zugriffskontrollpolitiken wird in [84] vorgestellt. In diesem Ansatz werden die Zugriffskontrollpolitiken für eine Organisationseinheit spezifiziert; sie werden also nicht dezentral spezifiziert. Die Grundidee basiert auf einem anfänglichen Zustand (*state*) eines Teilnehmers, an den durch Kompositionsoperatoren Zugriffsrechte angeheftet werden können. Ein signifikanter Unterschied zu der Algebra, die Bonatti et al. vorstellen, ist, dass Zugriffskontrollpolitiken sowohl Erlaubnisse als auch Verbote der Form $(s, o, -a)$ enthalten können. Mögliche Konflikte, welche durch die Komposition von Zugriffskontrollpolitiken auftreten können, werden dadurch umgangen, dass die Kompositionsoperationen nur dann durchgeführt werden, falls das Kompositionsergebnis (eine Zugriffskontrollpolitik) konsistent ist.

Ein verbreiteter Implementierungsmechanismus zur Sicherung von Zugriffskontrolle ist der Einsatz so genannter *Credentials*, vgl. [22, 26]. Ein signifikanter Vorteil für den Einsatz von *Credentials* in offenen Rechensystemen ist, dass diese auf Grund ihrer kryptographisch gesicherten Beglaubigungen über ungesicherte Kanäle versendet werden können. Im Rahmen einer attributbasierten Zugriffskontrolle können *Credentials* zur Kodierung von Teilnehmermerkmalen genutzt werden. Für einen typischen Zugriffsentscheidungsvorgang in einem *Credential*-basierten Zugriffskontrollsystem sendet ein Teilnehmer eine Baumstruktur von *Credentials* als Zugriffsanfrage an eine Zugriffskontrollkomponente, vgl. [27]. Für eine Zugriffserlaubnis muss die Struktur von mindestens einem Teilnehmer (*Zertifizierer*) ausgehen, den die Zugriffskontrollkomponente als vertrauenswürdig einschätzt. Eine prinzipielle Frage ist, was in diesem Kontext unter „Vertrauen“ verstanden wird. Diese Frage wird in den Arbeiten [3, 5, 6] behandelt. Üblicherweise tritt als vertrauenswürdiger Zertifizierer eine vertrauenswürdige dritte Partei (*certification authority*) auf, von welcher angenommen wird, dass sie stets korrekte Informationen zertifiziert und nicht zu betrügen beabsichtigt. Prinzipiell allerdings bleibt es den Teilnehmern eines Systems überlassen, welche anderen Teilnehmer als (allgemein) vertrauenswürdige Zertifizierer angesehen werden. Die mindesten Anforderungen hierfür sind, dass ein vertrauenswürdiger Zertifizierer seinen öffentlichen Schlüssel in authentischer Form vorlegt und den geheimen Schlüssel adäquat schützt.

Zertifizierungen im Allgemeinen werden durch eine so genannte Public Key Infrastruktur ermöglicht. Ein verbreiteter und in kommerziellen Anwendungen genutzter Standard hierfür ist X.509 [40, 45]. Ein weiterer relevanter, aus universitärem Umfeld stammender Vorschlag ist SPKI/SDSI [34, 35, 66].

Diese Arbeit schlägt zunächst eine **Credential-basierte Implementierung kompositional spezifizierter Zugriffskontrollpolitiken** vor, die sich auf die Arbeiten [1, 4] stützt. Anhand dieser *Credential*-basierten Implementierung werden Entwürfe allgemeiner **Architekturen eines kompositionalen und zustandsbasierten Zugriffskontrollsystems** erarbeitet, in die der Ansatz zustandsdynamischer Zugriffskontrollpolitiken eingebettet werden kann. Die Architekturen sind modular aufgebaut und ermöglichen somit eine logische Trennung der Spezifikation der zustandsdynamischen Zugriffskontrollpolitik, der Zugriffskontrollentscheidung und des tatsächlichen Durchsetzens der Zugriffsentscheidung. Hierbei wird der Fokus auf die Module innerhalb der Schutzdomäne einer Organisationseinheit des Systems gelegt, die übertragbar auf weitere im System vorkommende Organisationseinheiten sind.

Basierend auf den Architekturentwürfen werden zwei konzeptuelle Mechanismen entwickelt, die eine **Durchsetzung der zustandsdynamischen Zugriffskontrollpolitik in einer verteilten Umgebung ohne zentrale Kontrollinstanz** ermöglichen. Zum einen wird eine *Token*-basierte Realisierung entwickelt, welche allerdings zur Kommunikation der involvierten Teilnehmer untereinander auf ein zentrales *Schwarzes Brett* zurückgreifen muss. Ein ähnliches Konzept findet sich in einigen der zustandsbasierten Zugriffskontrollansätzen [11, 32]. In diesen Vertretern wird die gesamte Spezifikation der erlaubten Ausführungsfolgen an die betroffenen Teilnehmer verteilt und ein weitergebbares *token* markiert den jeweils aktuellen Zustand. Für eine weitere Variante einer Durchsetzung wird, in Anlehnung an den Ansatz [2], eine spezielle Datenstruktur, der *Container*, vorgeschlagen, der in seiner Implementierung als eine Art „Laufzettel“ an die betroffenen Organisationseinheiten weitergereicht wird. Diese Variante verzichtet auf die zentrale Kommunikationskomponente und bietet somit die Möglichkeit einer „echt“ verteilten Realisierung.

Aufbau der Arbeit

Teil I führt in die Umgebung ein, die dieser Arbeit zugrunde liegt und betrachtet relevante Ansätze aus der Literatur.

- In Kapitel 1 werden die Begrifflichkeiten erläutert, die Web-basierten Umgebungen zugrunde liegen. Eine spezieller Aufbau einer solchen Umgebung wird durch sogenannte Schutzdomänen gebildet, die das gewünschte Einsatzszenario der in

dieser Arbeit entwickelten Zugriffskontrollansätze sind. Dieses Kapitel schließt mit den Anforderungen an eine adäquate Zugriffskontrolle für Web-basierte Umgebungen, die sowohl kompositionale als auch zustandsbasierte Eigenschaften aufweisen sollte.

- In Kapitel 2 wird in die Arbeiten zur klassischen und modernen Zugriffskontrolle eingeführt. Wichtige Vertreter verschiedener Zugriffskontrollmodelle werden vorgestellt und hinsichtlich ihrer Einsatzmöglichkeiten in Web-basierten Umgebungen verglichen. Dieses Kapitel schließt mit einer zusammenfassenden Bewertung der aktuellen Literatur zur kompositionalen und zustandsbasierten Zugriffskontrolle.

Teil II legt den Fokus auf Zertifikat-basierte Realisierungsmöglichkeiten kompositionaler und explizit verteilter Zugriffskontrolle.

- Kapitel 3 führt in die *Public Key* Infrastruktur SPKI/SDSI ein, die den in dieser Arbeit vorgestellten Realisierungen zugrunde liegt. Daran anschließend wird ein Überblick über Vertrauensstrukturen in Zertifikat-basierten Rechensystemen gegeben, und es wird aufgezeigt, welchen Einfluss diese Vertrauensstrukturen auf unsere Umgebung der Schutzdomänen hat. Die Vertrauensstrukturen beeinflussen die Spezifikation von Zugriffskontrollpolitiken, das Ausstellen von Zertifikaten und den Rückruf bereits ausgestellter Zertifikate. Dieser Teil des Kapitels basiert auf den Veröffentlichungen [3, 5, 6]. Dieses Kapitel schließt mit einem Web-basierten Szenario einer digitalen Forschungsdatenbank, welches auch im Folgenden der Arbeit als Einsatzszenario dient.

- In Kapitel 4 wird eine Zertifikat-basierte Durchsetzung kompositionaler Zugriffskontrollpolitiken vorgestellt, die bereits in den Veröffentlichungen [1, 4] publiziert wurde. Die SPKI/SDSI-basierte Realisierung implementiert einen zentralen Teil eines in Teil I, Kapitel 2 vorgestellten Vertreter einer kompositionalen Zugriffskontrolle.

- In Kapitel 5 wird ein Ansatz vorgestellt, mit dem, ebenfalls basierend auf dem Einsatz der *Public Key* Infrastruktur SPKI/SDSI, eine explizit dezentrale Zugriffskontrolle durchgesetzt werden kann. Dieser Ansatz wurde in seinen Grundzügen bereits in der Veröffentlichung [2] vorgestellt und wurde für Web-basierte Umgebungen entworfen.

Teil III stellt ein Rahmenwerk für eine Zugriffskontrolle vor, die die Konzepte kompositionaler und zustandsbasierter Zugriffskontrolle vereint.

- In Kapitel 6 wird das Konzept der Domänenkonfigurationen entworfen. Domänenkonfigurationen sind gegeben durch sogenannte Ausführungssequenzen, die durch Kompositionsoperatoren kompositional spezifiziert werden können. Bereits mit

diesem Rahmenwerk kann sowohl traditionelle Zugriffskontrolle als auch (einfache) zustandsbasierte Zugriffskontrolle abgedeckt werden.

- In Kapitel 7 wird das Konzept der Domänenkonfigurationen dahingehend erweitert, dass beliebig definierte Domänenkonfigurationen unabhängig voneinander genutzt werden können. Dies wird durch die zustandsdynamischen Zugriffskontrollpolitiken erreicht, welche durch eine Menge identifizierter Domänenkonfigurationen gegeben werden. Zur kompositionalen Spezifizierung zustandsdynamischer Zugriffskontrollpolitiken werden ebenfalls Kompositionsoperatoren angegeben.

Teil IV behandelt Zertifikat-basierte Realisierungsmöglichkeiten, mit denen zustandsdynamische Zugriffskontrollpolitiken implementiert werden könnten.

- In Kapitel 8 wird eine zentrale Realisierung zustandsdynamischer Zugriffskontrolle erarbeitet. Des Weiteren wird eine SPKI/SDSI-basierte Möglichkeit aufgezeigt, mit der eine Einbindung von Rollenkonzepten ermöglicht werden kann. Ausgehend vom zentralen Entwurf werden verschiedene dezentrale Realisierungsmöglichkeiten vorgestellt, die abschließend, im Hinblick auf den anfallenden Nachrichtenverkehr, bewertet werden.

- In Kapitel 9 werden zwei SPKI/SDSI-basierte, dezentrale Realisierungsmöglichkeiten zustandsdynamischer Zugriffskontrolle vorgestellt und diskutiert. Die Realisierungsmöglichkeiten gründen auf den Konzepten der Realisierungen aus Teil II, Kapitel 4 und Kapitel 5, und wurden angepasst für das Einsatzszenario Web-basierter Umgebungen. Dieses Kapitel schließt mit Sicherheitsbetrachtungen, in denen mögliche Angriffe auf die Realisierungsmöglichkeiten untersucht werden.

Teil V bewertet die Ergebnisse dieser Arbeit abschließend.

Eigenanteil an den in Kooperation entstandenen Ergebnissen

Meine ursprüngliche Arbeit an kompositionalen Zugriffskontrollpolitiken [4] entstand in Zusammenarbeit mit Joachim Biskup. Sein Beitrag zu den Ergebnissen meiner Veröffentlichungen [2, 3, 4] erstreckt sich im Wesentlichen darauf, dass wir gemeinsam vielversprechende Ansätze und Konzepte erarbeiteten und diskutierten. Meine Ausarbeitungen der Ansätze und Konzepte hat Joachim Biskup darüber hinaus Korrektur gelesen und mich durch allgemeine Hilfestellungen unterstützt.

Aus meiner ursprünglichen Arbeit entstanden die Veröffentlichungen zur kompositionalen Zugriffskontrolle im Bereich der Webdienste [1] und deren Einfluss auf Vertrauensstrukturen [3, 5, 6]. Die grundlegenden Konzepte dieser Arbeiten habe ich

entworfen und ausgearbeitet. Barbara Sprick hat Ideen und konkrete Ausarbeitungen mit mir diskutiert. Sudhir Agarwal trug, als Experte für semantische Webdienste, zu möglichen Anwendungsszenarien bei und erarbeitete eine Einbettung der von mir entwickelten kompositionalen Zugriffskontrolle in die Beschreibungssprache DAML-S. Die von mir betreuten Diplomanden Julia Hielscher und Christoph Kobusch diskutierten mit mir Spezialfälle meiner Konzepte und gaben hilfreiche Anregungen. Die Veröffentlichung [2] über die echt verteilte Realisierung kompositionaler Webdienste ist, neben der Mitwirkung von Joachim Biskup, gemeinsam mit Frank Müller und in Kooperation mit Barbara Carminati, sowie deren Institutsleiterin Elena Ferrari entstanden.

Schließlich basiert die in Teil III besprochene zustandsdynamische Zugriffskontrollpolitik auf einem von Joachim Biskup in Zusammenarbeit mit mir gestellten DFG-Antrag.

Liste eigener Veröffentlichungen

[1] Sudhir Agarwal, Barbara Sprick, and Sandra Wortmann. Credential based access control for semantic web services. In Terry Payne, editor, *AAAI Spring Symposium - Semantic Web Services*, pages 44–52. AAAI Press, 2004.

[2] Joachim Biskup, Barbara Carminati, Elena Ferrari, Frank Müller, and Sandra Wortmann. Towards secure execution orders for web services. In L.-J. Zhang, K.P. Birman, and J. Zhang, editors, *Proceedings of the IEEE International Conference on Web Services*, pages 489–496. IEEE Computer Society, 2007.

[3] Joachim Biskup, Julia Hielscher, and Sandra Wortmann. A trust- and property-based access control model. *Electronic Notes in Theoretical Computer Science*, 197(2):169–177, 2008.

[4] Joachim Biskup and Sandra Wortmann. Towards a credential-based implementation of compound access control policies. In Trent Jaeger and Elena Ferrari, editors, *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 31–40. ACM Press, 2004.

[5] Barbara Sprick and Sandra Wortmann. Time dependent trust structures. *Computer Systems: Science & Engineering*, 20(6), 2005.

[6] Sandra Wortmann, Barbara Sprick, and Christoph Kobusch. Dynamically changing trust structure in capaility based access control systems. In Sokratis K. Katsikas, Javier Lopez, and Günther Pernul, editors, *Proceedings of the 1st International Conference on Trust and Privacy in Digital Business (TrustBus)*, volume 3184 of *Lecture Notes in Computer Science*, pages 50–59. Springer Verlag, 2004.

Teil I.

Umgebung und Literaturüberblick

Dieser Teil führt in die Umgebung ein, die dieser Arbeit zugrunde liegt. Es werden die Begrifflichkeiten erläutert, die Web-basierten Umgebungen zugrunde liegen. Eine spezieller Aufbau einer solchen Umgebung wird durch sogenannte Schutzdomänen gebildet, die im Folgenden das vorgestellte Einsatzszenario der in dieser Arbeit entwickelten Zugriffskontrollansätze sind.

Es wird in die Arbeiten zur klassischen und modernen Zugriffskontrolle eingeführt. Wichtige Vertreter verschiedener Zugriffskontrollmodelle werden vorgestellt und hinsichtlich ihrer Einsatzmöglichkeiten in Web-basierten Umgebungen verglichen.

Kapitel 1.

Hinführendes, Schutzdomänen und unsere Umgebung

Die Zugriffskontrolle eines Rechensystems verfolgt im Wesentlichen zwei Ziele: Eine Kontrollinstanz muss entscheiden, ob ein Zugriffswunsch eines Subjekts bezüglich einer bestimmten Systemressource autorisiert ist, und die jeweilige Zugriffsentscheidung muss wirksam durchgesetzt werden. Damit eine Kontrollinstanz eines Rechensystems eine Zugriffsentscheidung treffen kann, muss die Menge aller möglichen Zugriffe innerhalb des Systems zunächst inhaltlich festgelegt und dann entsprechend formalisiert in einer *Wissensbank* abgelegt werden.

Innerhalb der diskretionären Zugriffskontrolle wird für jedes zu schützende *Objekt* von allen oder ausgewählten Teilnehmern festgelegt, welche *Aktionen* von welchem *Subjekt* ausgeführt werden darf. Traditionsgemäß kann dies durch die Spezifizierung des Tupels (S, O, A) erfolgen, vgl. [42]. Dabei bezeichnet S die Menge aller Subjekte (beispielsweise Teilnehmer oder Prozesse), O die Menge aller von der Zugriffskontrolle geschützten atomaren Objekte und A die Menge der atomaren Aktionen, welche durch die Subjekte auf den Objekten ausgeführt werden können. Abgesehen von einer geeigneten Spezifikation für Zugriffskontrollpolitiken benötigt man für ihre tatsächliche Durchsetzung konkrete Sicherheitsmechanismen. Typischerweise wird zur Durchsetzung das *Referenzmonitorprinzip* oder Varianten hiervon verwendet, siehe [30]. Dieses konzeptuelle Prinzip besagt, dass jeder Zugriff von einem aktiven Subjekt auf ein zu schützendes Objekt kontrolliert werden muss und keine Möglichkeit zur Umgehung dieses Mechanismus existieren darf. Die Abbildung 1.1 veranschaulicht einen Ablauf eines gewährtem Zugriffswunsches eines Subjekts (Benutzer). Der Referenzmonitor greift zur Entscheidung auf festgelegte Richtlinien zu und führt die gewünschte Zugriffsoperation auf dem angefragten Objekt aus. Die Richtlinien eines geschlossenen Rechensystem sind im grundlegenden Konzept eine *Zugriffskontrollpolitik*, die durch eine Menge *elementarer Erlaubnisse* der Form (s, o, a) gegeben ist, wobei $s \in S$, $o \in O$ und $a \in A$ gilt. Eine solche Erlaubnis besagt, dass das Subjekt s autorisiert ist, die Aktion a auf dem Objekt o ohne einschränkende Bedingungen auszuführen. Dies bedeutet insbesondere, dass jede in

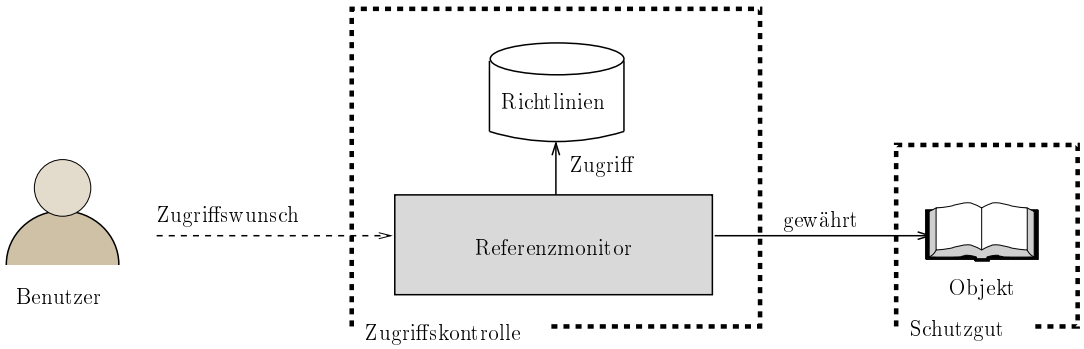


Abbildung 1.1.: Referenzmonitorprinzip bei gewährtem Zugriffswunsch.

der Zugriffskontrollpolitik vorkommende Erlaubnis unabhängig von den anderen genutzt werden kann.

In diesem Kapitel führen wir unser zugrunde liegendes Szenario einer *virtuellen Organisation* ein, in dem wir die Einsetzbarkeit unseres Rahmenwerkes zeigen. Basierend auf verschiedenen Übersichten dieser Thematik, wie beispielsweise [31, 36, 39, 85], stellen wir zunächst die wichtigsten Komponenten einer solchen verteilten Anwendung auf: *Dienste*, *Webdienste*, *dienstorientierte Architekturen* und virtuelle Organisationen. Basierend auf diesen grundlegenden Komponenten definieren wir unsere Umgebung und stellen dann die hieraus resultierenden Anforderungen an eine adäquate Zugriffskontrolle vor.

1.1. Dienste, Webdienste und virtuelle Organisationen

1.1.1. Dienste

Ein zentraler Punkt für die Definition von Diensten ist das Ergebnis der Dienstleistung und nicht die Art und Weise der Diensterbringung. Dies gilt ebenso für Dienste, die in konkreten Rechensystemen implementiert sind, wie für allgemeine Dienste, die in der realen Welt angeboten werden. Das Deutsche Institut für Normung gibt in DIN EN ISO 8402:1994 beziehungsweise in DIN EN ISO 9000:2005 eine Definition für den allgemeinen Begriff realer Dienste, in der dieser zentrale Punkt festgemacht wird. Hiernach ist ein Dienst

„*das Ergebnis mindestens einer Tätigkeit, die notwendigerweise an der*

> *Schnittstelle zwischen dem Lieferanten und dem Kunden ausgeführt wird und üblicherweise immaterial ist.*“

Aber auch Definitionen der Dienste, welche in (verteilten) Softwaresystemen eingesetzt werden, fokussieren auf den Kernpunkt des Ergebnisses der Dienstleistung. Bereits im OSI-Referenzmodell werden die Begriffe *Schicht* (*layer*) und *Dienst* (*service*) definiert. Jede Kommunikationsschicht im Referenzmodell nutzt Dienste der unmittelbar darunterliegenden Schicht und bietet der darüberliegenden Schicht wiederum Dienste an. Die durch eine Schicht bereitgestellten Dienste werden an definierten Dienstzugangspunkten (*service access point*) zur Verfügung gestellt. Diese Art der Verknüpfung der Dienste verdeutlicht, dass die Schichten untereinander die konkrete Implementierung der Dienste geheim halten. Ein Vorteil dieser Art der Verknüpfung ist, dass eine Änderung der konkreten Implementierung der Dienste ermöglicht wird, sofern die entsprechenden Dienstzugangspunkte unverändert bleiben. In späteren Jahren wurden verschiedene Definitionen eines Dienst verfeinert. Zwar wird weiterhin von der konkreten Dienstimplementierung abstrahiert, aber andere Aspekte als die reine Diensterbringung treten in den Vordergrund. In der Arbeit [31] wird der Dienstbegriff wie folgt definiert:

> „*A service is defined as a functionality that is provided with a certain quality and cost at a Service Access Point (SAP). [...] The functionality of a service consists of two parts: (i) the functionality of the service itself, and (ii) the functionality of sub-services that are involved in the service provisioning with respect to the service hierarchy.*“

In dieser Definition kommt die *Funktionalität* eines angebotenen Dienstes zur Sprache. Auch bezüglich der Funktionalität eines angebotenen Dienstes wird von der tatsächlichen Realisierung abstrahiert, indem nur der Dienstzugangspunkt als relevant hervorgehoben wird. Zudem wird die Möglichkeit der Komposition der Funktionalitäten bereitgestellter Dienste hervorgehoben. Dieses lässt bereits anklingen, dass Dienste typischerweise nicht nur atomar, sondern in strukturierter Form angeboten werden.

1.1.2. Webdienste

Der Begriff Webdienst wurde vom World Wide Web Consortium (W3C) geprägt. In der aktuellen Version des W3C-Glossars [83] findet sich folgende Definition:

> „*A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-*

> *messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*"

Die zentrale Aussage dieser Definition ist, dass ein Webdienst zur Realisierung von Kommunikationsbeziehungen zwischen Maschinen untereinander eingesetzt wird. Als Basis für diese Kommunikation dienen spezifische Standards, wie die *Extensible Markup Language* (XML) [79], die *Web Services Description Language* (WSDL) [81] zur Schnittstellenbeschreibung von Webdienste, das Kommunikationsprotokoll SOAP [80] als das Format zum Nachrichtenaustausch zwischen Webdiensten und schließlich *Universal Description, Discovery and Integration* (UDDI) [59] zur Verwaltung von Dienstverzeichnissen.

Es existieren verschiedene Beschreibungssprachen, mit denen strukturierte Formen von Webdiensten als *workflow* modelliert werden können. Der bekannteste Standard ist die *Business Process Execution Language* (BPEL) [60]. Der ausschlaggebende Vorteil von BPEL ist, dass aus einer gegebenen *Workflow*-Spezifikation direkt ausführbarer *Code* erzeugt werden kann. Dies funktioniert, da eine *Workflow*-Spezifikation in BPEL, ein sogenannter *BPEL-Prozess*, selbst einen Webdienst darstellt. Ein BPEL-Prozess besteht aus einer Prozess-Schnittstelle und einem Prozess-Schema. Die Prozess-Schnittstelle fungiert als Dienstzugangspunkt, während das Prozess-Schema den eigentlichen, strukturierten Ablauf der eingebundenen Webdienste spezifiziert.

1.1.3. Dienstorientierte Architekturen

Im Hinblick auf den Einsatz von Webdiensten in einem größeren Kontext können diese als Softwarekomponenten interpretiert werden. In der Informatik wird die Strukturierung eines Softwaresystems durch die sogenannte *Softwarearchitektur* beschrieben. Im Kontext der Softwarearchitektur besitzen Softwarekomponenten die folgenden Eigenschaften:

- Jede Softwarekomponente übernimmt eine spezifische Aufgabe im Gesamtsystem.
- Eine Softwarekomponente bietet der Außenwelt über eine wohldefinierte Schnittstelle einen oder eine Menge von Dienst(en) an.
- Die konkrete Realisierung einer Softwarekomponente ist gekapselt.

Wenn sich eine Softwarearchitektur aus Webdiensten zusammensetzt, wird von einer *dienstorientierten Architektur* (*Service-oriented architecture, SOA*) gesprochen.

In diesem Fall werden Webdienste als Schnittstellentechnologie eingesetzt. Mit diesem noch relativ neuen Paradigma der dienstorientierten Architektur wird eine Art der Systemmodellierung implementiert, die es erlaubt, ganze Systemarchitekturen

unabhängig voneinander zu entwickeln, zu warten und zu betreiben.

Von der tatsächlich vorhandenen Infrastruktur wird im Rahmen einer dienstorientierten Architektur so stark abstrahiert, dass nur noch die angebotenen Dienste, nicht jedoch ihre Implementierung oder die notwendigen Transportmechanismen im Vordergrund stehen.

Das W3C hat zu dienstorientierten Architekturen folgende Definition herausgegeben, siehe [82]:

> „*A Service Oriented Architecture (SOA) is a form of distributed systems architecture.* [It consists of] *a set of components which can be invoked, and whose interface descriptions can be published and discovered.*“

Die Organisationsstrukturen von Unternehmen sind in den letzten Jahren einem Wandel unterlegen, der den Organisationen eine schnelle Anpassbarkeit ihrer Geschäftsprozesse bei gleichzeitiger intensiver Kollaboration auch über Organisationsgrenzen hinweg abverlangt. Existierende, zentralistische Architekturen erzielen nicht die erforderliche Flexibilität, um Dienste innerhalb von Organisationen und über Organisationsgrenzen hinweg zu erbringen. Dienstorientierte Architekturen hingegen ermöglichen die Unterstützung und Durchsetzung von virtuellen Organisationen.

1.1.4. Virtuelle Organisationen

In den letzten Jahren avancierte der Begriff der virtuellen Organisation [58] zu einem Schlagwort. Eine virtuelle Organisation ist ein kooperatives Netzwerk von autonom agierenden und kommunizierenden Organisationseinheiten, welche ihre Dienste für einen befristeten Zeitraum zu einem losen Verbund zusammenschließen.

Der Begriff der *Virtualität* wird heute in vielen wissenschaftlichen Disziplinen verwendet. In den späten 50er Jahren wurde der Begriff in der Informatik eingeführt: Großrechner, welche die gemeinsame Nutzung einer zentral beherbergten Anwendungssoftware ermöglichten, wurden als virtuelle Computer bezeichnet. Diese Rechner ermöglichten es mehreren Personen gleichzeitig an einer Anwendung zu arbeiten ohne Einschränkungen in der Verfügbarkeit oder Performanz. Seitdem steht der Begriff Virtualität für Anpassungsfähigkeit, hohe Verfügbarkeit und Interaktivität.

Durch die steigende Digitalisierung und Automatisierung einer Vielzahl von Produkten entstehen zu immer mehr realen Objekten virtuelle Ausprägungen, beispielsweise virtuelle Produkte oder *virtuelle Organisationen*. Mitte der 80er Jahre verwendete Mowshowitz in [58] nach eigenem Bekunden als erster den Begriff der virtuellen Organisation. Davidow und Malowne verwenden 1992 als erste diesen Begriff im Zusammenhang mit Unternehmenskooperation, siehe [85]. Wichtige Aspekte in der

Verwaltung virtueller Organisationen sind die dynamische und flexible Zuordnung von Dienstanbietern zu Dienstnutzern und dem konkreten Ort der Diensterbringung. Den wachsenden Anforderungen eines sich verschärfenden Wettbewerbes kann eine Organisation nur dann entsprechen, wenn sie flexibel und gleichzeitig effizient ist. Diese Balance erreicht die virtuelle Organisation einerseits durch ihre Fähigkeit, sich durch flexibles Aufnehmen und Ausschließen von Organisationseinheiten an die aktuellen Aufgaben anzupassen. Andererseits können sich die Organisationseinheiten sehr stark spezialisieren und so eine hohe Effizienz bezüglich ihrer eigenen Abläufe erreichen.

Trotz verschiedener Varianten der Definition einer virtuellen Organisation, lassen sich folgende Merkmale feststellen:

- *Autonomie:* Eine virtuelle Organisation ist ein spontaner Zusammenschluss aus zwei oder mehreren rechtlich unabhängigen Organisationseinheiten. Diese Organisationseinheiten verfügen über eigenverantwortliche Entscheidungskompetenz und Verantwortung hinsichtlich der jeweiligen Diensterbringung. Dies bedeutet, dass eine virtuelle Organisation (weitgehend) auf zentrale Kontrollinstanzen und hierarchische Gestaltungsprinzipien verzichtet.

- *Heterogenität:* Die Organisationseinheiten einer virtuellen Organisation schließen (Teile) ihre(r) Dienste für einen befristeten Zeitraum zum Zwecke der Erreichung bestimmter Ziele zusammen. Die eingebundenen virtuellen Organisationseinheiten decken somit unterschiedlichste Dienstangebote ab.

- *Räumliche und zeitliche Verteiltheit:* Für die tatsächliche Ausführung des gewünschten Dienstes ist der konkrete Ort und (oft auch) Zeitpunkt nicht relevant. Die einzelnen Organisationseinheiten sind meist räumlich getrennt, treten aber einem Dienstnutzer gegenüber einheitlich auf.

Ein wichtiger Aspekt in virtuellen Organisationen ist nicht nur der Schutz personenbezogener Daten, sondern vor allem auch die Wahrung von Betriebs- und Geschäftsgeheimnissen der eingebundenen Organisationseinheiten. Beide Schutzgüter weisen im Hinblick auf die strukturellen Fragen der Verantwortung für die Wahrung der erforderlichen Vertraulichkeit sowie der Gewährleistung der Datensicherheit Parallelen auf.

Im Rahmen einer Durchsetzung des in dieser Arbeit entworfenen Rahmenwerks, werden wir uns in Teil IV mit virtuellen Organisationen als mögliches Anwendungsszenario beschäftigen.

1.2. Unsere Umgebung – Die Domänen

In diesem Abschnitt legen wir die Umgebung fest, in der die Einsetzbarkeit der in dieser Arbeit entwickelten Zugriffskontrolle gezeigt wird. Wir betrachten hierfür eine virtuelle Organisation, welche aus verschiedenen Organisationseinheiten zusammengesetzt ist. Der zentrale Punkt jeder Organisationseinheit ist eine *Schutzdomäne*. Innerhalb einer Schutzdomäne befinden sich alle Schutzgüter einer Organisationseinheit. Im Rahmen unserer Betrachtungen vereinfachen wir diese Sicht, indem wir Schutzgüter ausschließlich auf die angebotenen Dienste beziehungsweise auf deren Funktionalitäten fokussieren und weitere Schutzgüter der eingebundenen Organsiationseinheiten außer Acht lassen. Eine Schutzdomäne verfügt über einen Zugriffspunkt beziehungsweise eine Kommunikationsschnittstelle, die mit geeigneten Zugriffskontrollmechanismen geschützt wird. Ausgehend von einem *besitzerorientiertem Ansatz* unterstellen wir die zu schützenden Objekte einer Schutzdomäne einem ausgezeichneten Teilnehmer, dem Besitzer. Wir untersuchen, inwieweit sich virtuelle Organisationen im Allgemeinen und deren Zugriffskontrolle im Speziellen mit dieser Sicht auf Schutzdomänen vereinbaren lassen.

1.2.1. Eine Schutzdomäne

Der einfachste Fall liegt in der Betrachtung genau einer Schutzdomäne. Vergleichbar ist dieses Szenario mit klassischer Zugriffskontrolle für zentrale Systeme. Teilnehmer können ihre Zugriffswünsche auf die geschützten Objekte an genau einen Zugriffspunkt der Schutzdomäne richten. Die Zugriffskontrollinstanz greift auf eine zentrale Stelle zu, an der die Menge aller als erlaubt angesehenen Zugriffe spezifiziert sind. Die Abbildung 1.2 veranschaulicht dieses einfache und zentrale Szenario. Alle in der Schutzdomäne befindlichen Dienste sind dem Besitzer zugeordnet. Dieser Besitzer agiert als *Verwalter der Schutzdomäne*, indem er die Zugriffskontrollpolitik für seine Dienste spezifiziert und über deren Durchsetzung wacht. Unabhängig davon, ob die Schutzdomäne nur einen oder mehrere Dienste enthält, spezifiziert der Verwalter alle als erlaubt angesehenen Funktionalitäten seines Dienstes oder seiner Dienste, indem er explizit alle erlaubten Zugriffe festlegt. Da in diesem Szenario eine zentrale und somit direkte Kontrolle und Überwachung aller Zugriffswünsche und tatsächlichen Zugriffe möglich ist, ist es unerheblich, ob diese *erlaubten Funktionalitäten* atomar oder strukturiert spezifiziert werden. Diese zentrale besitzerorientierte Sicht löst sich beim Betrachten virtueller Organisationen auf.

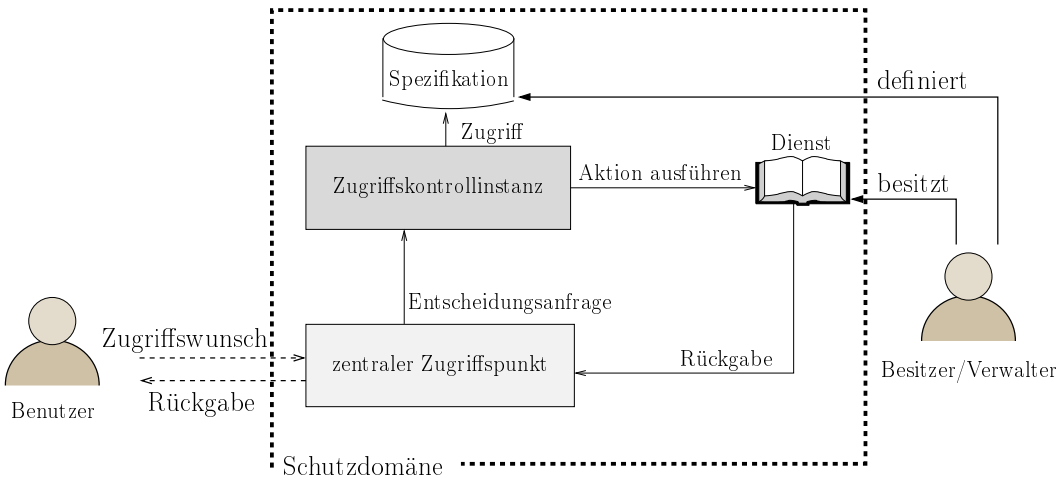


Abbildung 1.2.: Eine Schutzdomäne mit zentralem Zugriffspunkt.

1.2.2. Mehrere Schutzdomänen

Einer virtuellen Organisation liegen mindestens zwei Organisationseinheiten zugrunde, welche einige ihrer Dienste zur Verfügung stellen. Im einfachsten Fall sind die Dienste der Organisationseinheiten disjunkt. Dann existieren zwei Schutzdomänen, welche unabhängig voneinander verwaltet werden. Dieser Aspekt wurde in Abschnitt 1.2.1 besprochen. Aufgrund der Dynamik virtueller Organisationen kann es vorkommen, dass zwei Organisationseinheiten denselben Dienst anbieten. Dies bedeutet, dass der besitzerorientierte Ansatz im Hinblick auf die Zugriffskontrolle gelockert wird. Angebotene Dienste werden in diesem Modell nicht mehr nur genau einer Schutzdomäne zugeordnet. Da wir zudem autonom agierende Organisationseinheiten fordern, kann dies zur Folge haben, dass ein Dienst unterschiedlichen Zugriffskontrollen unterliegt. Jeder Verwalter einer Schutzdomäne kann autonom Zugriffskontrollpolitiken für die enthaltenen Dienste spezifizieren, unabhängig davon, ob dieser Teilnehmer Besitzer des jeweiligen zu schützenden Objekts ist, und unabhängig von den Zugriffskontrollpolitiken anderer Schutzdomänen. Die Abbildung 1.3 veranschaulicht das Szenario zweier Schutzdomänen mit einem überlappenden Dienst. Die Zugriffskontrolle in diesem Szenario läuft nicht ausschließlich über nur einen Zugriffspunkt. Möchte ein Teilnehmer eine Funktionalität eines Dienstes in Anspruch nehmen, so muss er eine Schutzdomäne identifizieren, die den Dienst beinhaltet. Dann wendet sich der potenzielle Dienstnutzer mit seinem Zugriffswunsch an genau einen Zugriffspunkt einer ausgewählten Schutzdomäne. Die Spezifikation und Durchsetzung der zugehörigen Zugriffskontrollpolitik liegt allein bei dem Verwalter, an dessen Zugriffspunkt der Zugriffswunsch gerichtet wurde. Dieser

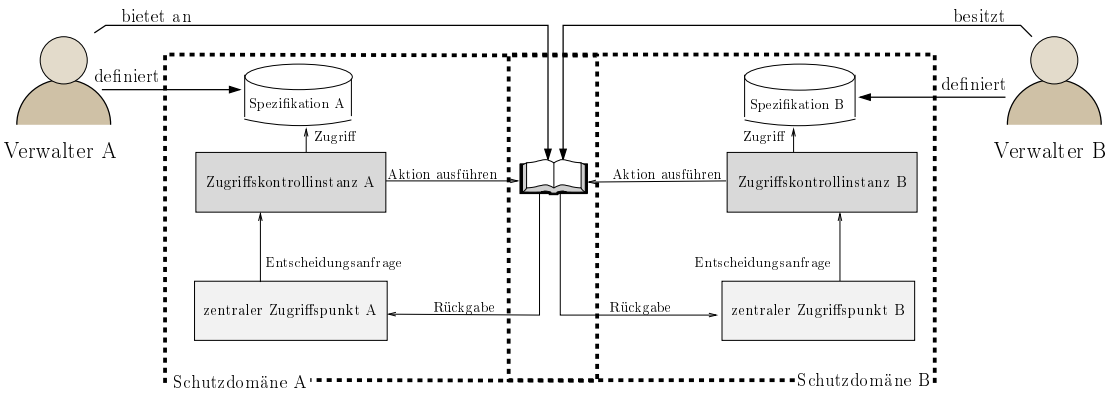


Abbildung 1.3.: Zwei überlappende Schutzdomänen mit gemeinsamen Dienstangebot.

Verwalter ist in diesem Szenario nicht unbedingt der Besitzer des gewünschten Dienstes. Es müssen allerdings zwischen dem Verwalter und dem Besitzer Absprachen oder Verträge bestehen. Aufgrund der Umgebung einer virtuellen Organisation, in der die eingebundenen Organisationseinheiten Absprachen untereinander treffen, können wir hiervon ausgehen. In Kapitel 3.2 gehen wir detaillierter auf die notwendigen Vertrauensstrukturen der Teilnehmer untereinander ein. In unserer Publikation [4] ist untersucht worden, wie die Zugriffskontrolle in einem solchen *verwalterorientierten Ansatz* realisiert werden kann. In diesem Vorschlag existieren sich überlappende Schutzdomänen (*organisational entities*), die gemeinsame Dienste teilen, aber bedingt durch unterschiedliche Verwalter unterschiedliche Zugriffsanforderungen haben können. In Kapitel 3.3 greifen wir dieses Szenario erneut auf und stellen beispielhaft eine mögliche Zugriffskontrolle vor.

1.2.3. Mehrere Schutzdomänen – freie Mediatoren

Den obigen Überlegungen liegt die implizite Annahme zugrunde, dass ein Zugriffswunsch für die Inanspruchnahme einer Funktionalität eines Dienstes nur dann von einem Teilnehmer geäußert werden kann, wenn dieser über eine zugehörige Schutzdomäne und deren zentralen Zugriffspunkt in Kenntnis ist. Vorstellbar ist dies in Szenarien kleiner (virtueller) Organisationen. Betrachtet man allerdings den Einsatz virtueller Organisationen realistisch, insbesondere den Anspruch dynamischer und flexibler Zuordnung von Dienstanbietern zu Dienstnutzern, so reicht ein solch beschränktes Szenario nicht aus. Um größere virtuelle Organisationen abbilden zu können, müssen wir berücksichtigen, dass potenzielle Dienstnutzer nicht zwangsläufig in Kenntnis über den

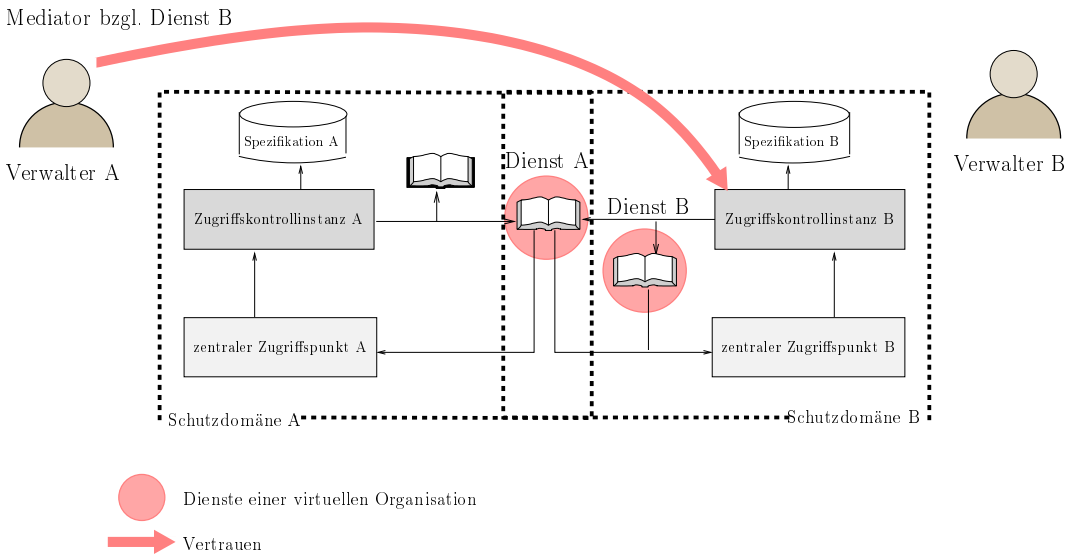


Abbildung 1.4.: Erweiterte Schutzdomänen mit Einbeziehung eines Mediators.

Zugriffspunkt sind, dessen Schutzdomäne den gewünschten Dienst enthält. Dies bedeutet, dass die verwaltenden Teilnehmer untereinander kooperieren müssen. Somit können bei einem Zugriffspunkt angeforderte Dienste außerhalb der jeweiligen Schutzdomäne liegen. In einem solchen Fall gehen wir davon aus, dass der zuständige Verwalter als *Mediator* fungiert: Der Mediator überprüft die Einhaltung der Zugriffskontrollanforderung entsprechend der Zugriffskontrollpolitik der zugehörigen Schutzdomäne, und leitet den eigentlichen Zugriffswunsch weiter. Dieser Zugriffswunsch wird dann an die Zugriffskontrollinstanz der betreffenden Schutzdomäne weitergeleitet. Veranschaulicht wird das Szenario der erweiterten Schutzdomänen in Abbildung 1.4. Dieses Szenario ermöglicht die Kontrolle verteilter und dynamischer virtueller Organisationen, basiert aber grundlegend auf notwendigen Vertrauensstrukturen zwischen den agierenden Teilnehmern. Vorstellbar ist, dass Besitzer Verträge mit einem oder mehreren Verwaltern unterhalten, die diese dazu berechtigen als Mediatoren zu agieren. Offensichtlich muss hier von unterschiedlichen Verträgen beziehungsweise von unterschiedlichen Vertrauenstrukturen der einzelnen Teilnehmer untereinander ausgegangen werden. Dieser Aspekt wird in Kapitel 3.2 Vertrauenstrukturen erläutert.

1.3. Festlegungen und Anforderungen

In diesem Kapitel haben wir die Umgebung definiert, für die in dieser Arbeit eine zustandsbasierte und kompositionales Rahmenwerk für eine Zugriffskontrolle realisiert werden soll. Die Umgebung, welche weiterhin betrachtet wird, baut auf einer virtuellen Organisation auf, welche aus mehreren Schutzdomänen, mit und ohne Mediatoren, besteht.

Zusammenfassend listen wir folgende Festlegungen, die so in dieser Arbeit verwendet werden.

Definition 1.1 (Dienst). *Ein Dienst ist ein Objekt oder ein Zusammenschluss mindestens zweier Objekte gemeinsam mit den zugehörigen Aktionen.*

Definition 1.2 (Funktionalität). *Die Funktionalität eines Dienstes sind die (möglicherweise) strukturierten Aktionen, die von den den Dienst ausmachenden Objekten angeboten werden.*

Definition 1.3 (Organisationseinheit). *Eine Organisationseinheit ist ein Zusammenschluss mehrerer Dienste.*

Definition 1.4 (Schutzdomäne). *Eine Schutzdomäne beinhaltet die angebotenen Dienste und die Zugriffskontrolle einer Organisationseinheit. Eine Schutzdomäne unterliegt genau einem Verwalter und verfügt über genau einen Zugriffspunkt, an den anfragende Subjekte ihren Zugriffswunsch richten. Sie verfügt über eine spezielle Kommunikationsschnittstelle, an die Mediatoren einen bereits überprüften und als erlaubt angesehenen Zugriffswunsch weiterleiten können.*

Definition 1.5 (virtuelle Organisation). *Eine virtuelle Organisation ist ein Zusammenschluss mehrerer Organisationseinheiten, welche temporär einige Dienste ihrer Schutzdomänen zur Verfügung stellen. Eine virtuelle Organisation wird durch eine virtuelle Schutzdomäne repräsentiert, welche die angebotenen Dienste enthält. Somit hat jede virtuelle Organisation einen zuständigen Verwalter und eine Zugriffskontrolle.*

Virtuelle Organisationen sind eine Ausprägung moderner und vor allem verteilter Rechensysteme. Diese Rechensysteme verfügen über eine sehr große Anzahl unbekannter Teilnehmer, die als potenzielle Dienstnutzer auftreten können. Für eine solche Teilnehmervielfalt mit komplexen angebotenen Diensten muss das grundlegende Konzept atomarer Erlaubnisse beziehungsweise atomarer Zugriffe auf einzelne Objekte geeignet verfeinert werden, um eine tragfähige Zugriffskontrolle gewährleisten zu können.

In dem Schutzdomänenszenario ist allein der jeweilige Verwalter verantwortlich für das Spezifizieren und Durchsetzen der Zugriffskontrollpolitik seiner Schutzdomäne. Die

Zugriffskontrollpolitik enthält elementare oder strukturierte Erlaubnisse der atomaren Form (s, o, a). In dem Fall, dass mehrere Schutzdomänen betrachtet werden, müssen verschiedene solcher Zugriffskontrollpolitiken geeignet kombiniert werden, um die Zugriffskontrollpolitik einer virtuellen Organisation zu erhalten. Folgende Anforderungen stellen wir an eine adäquate Zugriffskontrolle für verteilte Rechensysteme, in denen einer großen Anzahl potenziell unbekannter Teilnehmer strukturierte Dienste angeboten werden.

Anforderung 1.6 (Kompositionalität). *Der Zusammenschluss mehrerer Schutzdomänen beziehungsweise mehrerer Organisationseinheiten zieht nach sich, dass mehrere Verwalter unterschiedliche Zugriffskontrollpolitiken spezifizieren können müssen. Dieses wiederum impliziert eine kompositionale Vorgehensweise im Hinblick auf die Spezifizierung: Involvierten Verwaltern muss erlaubt werden, autonom lokale Zugriffskontrollpolitiken zu spezifizieren und diese im Rahmen der globalen kombinierten Zugriffskontrollpolitik der virtuellen Organisation durchgesetzt zu wissen.*

Anforderung 1.7 (Zustandsabhängigkeit). *Um komplexe Dienste der einzelnen Organisationseinheiten unterstützen zu können, müssen Zugriffskontrollpolitiken nicht nur für einzelne atomare Erlaubnisse festgelegt werden, sondern insbesondere für komplexe Folgen von Erlaubnissen. Um solche strukturierte Dienste zu kontrollieren, müssen zustandsbasierte Zugriffskontrollpolitiken spezifiziert und durchgesetzt werden können.*

Anforderung 1.8 (formale Parameter). *Möglicherweise muss eine sehr große Anzahl unbekannter Teilnehmer behandelt werden können. Hierfür muss eine Zugriffskontrollpolitik nicht nur (komplexe) Erlaubnisse basierend auf identifizierten Subjekten und Objekten ausdrücken können. Teilnehmer und Objekte sollen mittels formaler Parameter oder ähnlichen Mechanismen beschrieben werden können.*

Anforderung 1.9 (Berechenbarkeit). *Die Zugriffskontrollspezifikation muss eine effiziente und berechenbare Zugriffskontrolle ermöglichen.*

Aus dem speziellen Einsatzszenario der Zugriffskontrolle in virtuellen Organisationen ergibt sich zudem die folgende Anforderung in Hinblick auf die teilnehmenden Organisationseinheiten.

Anforderung 1.10 (Gleichstellung). *Alle teilnehmenden Organisationseinheiten sollen über dieselben Möglichkeiten zur Durchsetzung der Zugriffskontrolle verfügen. Während der Betriebsphase der virtuellen Organisation sollen alle teilnehmenden Organisationseinheiten gleichgestellt sein. Es sollen keine ausgezeichneten Organisationseinheiten mit speziellen Aufgaben oder Pflichten hinsichtlich der Zugriffskontrolle existieren.*

Kapitel 2.

Einführung in Sicherheitspolitiken und bisherige Arbeiten

Eine der ersten Definitionen für den Begriff der *Sicherheitspolitik* (*Security Policy*) stammt von J. A. Goguen und J. Meseguer aus dem Jahre 1982, siehe [38]. In ihrer Arbeit definieren sie im Allgemeinen, dass eine Sicherheitspolitik „*... the security requirements for a given system*“ festlegt. In den unter dem Namen *Orange Book* bekannten *Trusted Computer System Evaluation Criteria* (TCSEC) [30] wird eine Sicherheitspolitik als eine

> „*Sammlung von Gesetzen, Regeln und Methoden, die festlegen, wie eine Organisation sensitive Informationen verwalten, schützen und zu verteilen hat*“

definiert und eine Zugriffskontrollpolitik als eine

> „*Sammlung von Regeln, die vom System verwendet werden, um zu entscheiden, ob der Zugriff eines bestimmten Subjekts auf ein spezielles Objekt erlaubt werden kann*“.

Betrachten wir eine Organisation und das diese repräsentierende Rechensystem, so lassen sich die Sicherheitspolitiken für verschiedene Ebenen definieren: Für

1. die Organisationsebene,
2. die Systemebene und
3. die technische Ebene.

Für jede Ebene wird eine andere Semantik einer Sicherheitspolitik zugrunde gelegt. Auf der Organisationsebene werden im Allgemeinen die zu schützenden Güter der Organisation sowie die strategischen Sicherheitsziele festgelegt. Auf der Systemebene werden die Richtlinien für die Verwaltung sensibler Informationen für das konkrete

Rechensystem festgelegt. Auf der technischen Ebene schließlich wird spezifiziert, wie Hard- und Software die Ressourcen des Rechensystems nutzen dürfen. Auf dieser Ebene wird eine Sicherheitspolitik durch eine Zugriffskontrollpolitik operationalisiert, welche Autorisierung von Zugriffsoperationen auf zu schützende Objekte implementiert. Eine solche operationalisierte Sicherheitspolitik ist zentral für die vorliegende Arbeit und wird in den folgenden Abschnitten genauer betrachtet. Während eine Sicherheitspolitik auf Organisationsebene überwiegend informell beschrieben wird, werden zur Festlegung von Sicherheitspolitiken auf der Systemebene (*system security policy*) und der technischen Ebene (*technical security policy*) weitestgehend formale Methoden eingesetzt. Eine Formalisierung ist relevant, da neben der Spezifizierung einer Zugriffskontrollpolitik auch deren Validierung und Durchsetzung erreicht werden sollte. Hierfür wird das betrachtete Rechensystem mittels einem *Sicherheitsmodell* beschrieben. Ein Sicherheitsmodell eines Rechensystems beinhaltet die formale Beschreibung von Sicherheitseigenschaften. Um das Modell realisieren zu können, ist unter anderem die Spezifikation einer Zugriffskontrollpolitik in einer geeigneten *(Zugriffskontroll)Politiksprache* notwendig. Diese Politiksprache dient dem Zugriffskontrollmechanismus des Rechensystems als Grundlage für die Zugriffsentscheidung.

Im Folgenden werden einige aus der Literatur bekannte Sicherheitsmodelle und Zugriffskontrollsprachen vorgestellt. Die Auswahl der vorgestellten Ansätze wurde anhand der Relevanz für das Verständnis dieser Arbeit getroffen. In Abschnitt 2.1 beginnen wir zunächst mit den klassischen Zugriffskontrollmodellen. Weitere spezielle Sicherheitsmodelle wie beispielsweise *Vertrauensmodelle* werden nicht an dieser Stelle sondern in Abschnitt 3.2 angesprochen. In den folgenden Abschnitten stellen wir weiterhin einige Vertreter für konkrete Zugriffskontrolle aus der aktuellen Literatur vor. Im Hinblick auf die Zielsetzung dieser Arbeit betrachten wir beispielsweise in Abschnitt 2.2 Ansätze zur kompositionalen Zugriffskontrolle, in Abschnitt 2.3 Ansätze für Zugriffskontrolle in verteilten Umgebungen und schließlich in Abschnitt 2.4 Ansätze für zustandsabhängige Zugriffskontrolle.

In den folgenden Abschnitten abstrahieren wir zunächst von unserer speziellen Umgebung, wie sie in Kapitel 1.2 vorgestellt wurde. Die Modelle, die beschrieben werden, sind allgemeingültig, und zunächst gehen wir von einem Rechensystem aus, in dem die Subjekte als *Benutzer* oder *Besitzer* agieren. Auf den Objekten des Rechensystems können *Operationen* ausgeführt werden, die mit Hilfe eines Zugriffskontrollmechanismus kontrolliert werden.

2.1. Klassische Zugriffskontrollmodelle

In traditionellen Zugriffskontrollmodellen in zentralen Systemen [69] wird zunächst die Authentizität einer Zugriffsanfrage eines Teilnehmers überprüft und dann die Zugriffsentscheidung aufgrund zuvor vergebener Zugriffsrechte getroffen. Zwei klassische Strategien für Zugriffskontrollmodelle sind die *diskretionäre Zugriffskontrolle* (*discretionary access control*) und die *mandatorische Zugriffskontrolle* (*mandatory access control*), vergleiche beispielsweise [39]. Überwiegend verwendet wird in heutigen Rechensystemen die diskretionäre Zugriffsrechtevergabe. In diesem Zugriffskontrollmodell wird für jedes zu schützende Objekt im Rechensystem individuell und meist lokal festgelegt, welche Zugriffsoperationen einzelne Subjekte auf diesem Objekt ausführen dürfen. Dieses Modell wird häufig in Kombination mit dem so genannten *Eigentümerprinzip* verwendet: Der Besitzer eines Objekts darf festlegen, welche Zugriffsoperationen für welches Subjekt auf seinem Objekt als erlaubt angesehen werden. Der Besitzer allein entscheidet über Vergabe, Weitergabe und Änderungen solcher Zugriffsrechte an seinen Objekten. Während die diskretionäre Zugriffskontrolle eine individuelle Definition von Zugriffsrechten ermöglicht, basiert das mandatorische Zugriffskontrollmodell auf systemweit festgelegten Regeln, die auf existierenden Hierarchiestrukturen unter den Subjekten und Objekten basieren. Neben der direkten Zugriffskontrolle auf einzelne Objekte wird in diesem Modell auch der Informationsfluss berücksichtigt: Ein Subjekt, welches durch eine Zugriffsoperation sensible Informationen über ein Objekt erhält, soll daran gehindert werden diese Informationen an andere Subjekte weiterzugeben, die keinen Zugriff auf das betreffende Objekt haben.

Zur Spezifizierung, Validierung und Durchsetzung der jeweiligen Zugriffskontrollpolitiken können für beide Modelle entweder eine zentrale oder mehrere dezentrale *Administrierungsstelle(n)* verwaltet werden. Zur Effizienzsteigerung bei der Zugriffsrechtevergabe und -verwaltung haben sich Mechanismen wie *rollenbasierte Zugriffskontrolle* (*Role-based access control*, RBAC) [37], Benutzer- oder Objektgruppenkonzepte etabliert.

2.1.1. Diskretionäre Zugriffskontrollmodelle

Die *diskretionäre Zugriffskontrolle* ist das weitverbreiteste Zugriffskontrollmodell. Zugrunde liegt hier die direkte Zugriffsrechtevergabe zwischen Subjekt und Objekt. Häufig wird die diskretionäre Zugriffskontrolle als *identitätsbasierte Zugriffskontrolle* (*identity based access control*) bezeichnet. Ob eine Zugriffsoperation auf einem Objekt als erlaubt angesehen wird, wird bei diesem Modell auf der Basis der Identität des Subjekts entschieden.

Da dieses Zugriffskontrollmodell auf dem Eigentümerprinzip fußt, gibt es (mindestens) ein ausgezeichnetes Subjekt, das Besitzer eines Objekts ist. Standardmäßig wird der Benutzer, der ein neues Objekt erzeugt hat, Besitzer (Eigentümer) dieses Objekts. Der Besitzer hat dann das Recht, an andere Benutzer beliebige Zugriffsrechte auf das Objekt weiter zu geben: In objektbezogenen *Zugriffskontrolllisten* (*access control list*, ACL) werden für jeden Identifikator die zugehörigen Zugriffsrechte hinterlegt. Alternativ können Zugriffsrechte auch in subjektbezogen Berechtigungen, sogenannten *capabilities* definiert werden. Die Zugriffskontrollliste für ein Objekt o_1 stellt die Teilmenge der Tupel einer Zugriffsrelation ZR dar, für die gilt: $ACL_{o_1} = \{(s, o, a) \in ZR \mid o = o_1\}$. Der Zugriffsschutz zu einem gegebenen Zeitpunkt t lässt sich dann durch eine $|S_t| \times |O_t|$ Matrix M_t mit folgenden Eigenschaften modellieren:

- Die Zeilen der Matrix werden durch die Menge S_t der Subjekte und
- die Spalten der Matrix werden durch die Menge O_t der Objekte definiert.
- Es gilt: $M_t : S_t \times O_t \to 2^A$, wobei A die Menge der Aktionen festlegt.

Ein Eintrag $M_t(s, o) = \{a_1, \ldots, a_n\}$ beschreibt dann die Menge der Zugriffsrechte, die das Subjekt s zum Zeitpunkt t an dem Objekt o besitzt.

In großen Rechensystemen ist von einer großen oder sogar anwachsenden Menge von Subjekten und Objekten auszugehen. Aus diesem Grund wurde die rollenbasierte Zugriffskontrolle entwickelt, welche das Konzept der *Rolle* einführt. Die Rollen fungieren als Vermittler zwischen Subjekten und Zugriffsrechten und orientieren sich an der organisatorischen Struktur des zugrunde liegenden Rechensystems. Im Gegensatz zu der Menge von Subjekten ist eine Rollendefinition im Zeitablauf relativ unveränderlich. Dieses vereinfacht die Zuweisung von Zugriffsrechten, da die Zuweisung von Subjekten zu Rollen und die Rollendefinition unabhängig voneinander vorgenommen werden können.

Rollenbasierte Zugriffskontrolle

In der Literatur finden sich verschiedene Ansätze, die Rollen als Mittel der Zugriffskontrolle verwenden, siehe beispielsweise [37, 70]. In dieser rollenbasierten Zugriffskontrolle werden (mehreren) Subjekten sogenannte Rollen zugeordnet, sie dienen der Abstraktion der Subjekte. Die Zugriffsrechte werden mit Rollen und nicht mit einzelnen Subjekten verknüpft. Eine Rolle beschreibt die Funktion, die ein Subjekt aktuell ausübt. Für die Ausübung dieser Funktion sind dabei, je nach Rolle, möglicherweise unterschiedliche Zugriffsrechte notwendig. Systemseitig werden ausschließlich auf Basis dieser Rollen die Zugriffsrechte für die Objekten vergeben, für die Zugriffskontrollentscheidung ist die eigentliche Identität des Benutzers selbst nicht

mehr maßgeblich.

Das RBAC-Basismodell aus [70] kann durch folgendes Tupel definiert werden: $RBAC = (S, O, RL, P, sr, pr, session)$. Dabei gilt:

- S ist die Menge der Subjekte.
- O ist die Menge der zu schützenden Objekte.
- RL ist die Menge der Rollen.
- P ist die Menge der Zugriffsrechte.

Die Abbildung sr modelliert die Rollenzugehörigkeit eines Subjekts $s \in S$, so dass $sr : S \to 2^{RL}$ gilt. Falls $sr(s) = \{R_1, \ldots, R_n\}$ gilt, dann ist das Subjekt s autorisiert die Rollen $R_1, \ldots, R_n$ zu aktivieren. Die Berechtigungen einer Rolle $R \in RL$ werden über die Abbildung $pr : RL \to 2^P$ definiert. Die Relation $session \subseteq S \times 2^{RL}$ definiert Paare (s, r_l), die als Sitzungen bezeichnet werden, wobei gelten muss: $r_l \subseteq sr(s)$. Für ein Subjekt $s \in S$ und für eine Sitzung $(s, r_l) \in session$ gilt dann, dass s alle Berechtigungen der Rollen $R \in r_l$ besitzt.

Attributbasierte Zugriffskontrolle

Eine neuere Variante der diskretionären Zugriffskontrolle findet sich in der *attributbasierten Zugriffskontrolle* (*Attribute-Based Access Control*, ABAC), siehe beispielsweise [7, 61, 87]. In [7] wurde dieses Konzept zuerst als *Metadata-based access control* eingeführt. Subjekte und Objekte sind durch eine Menge von Attributwerten repräsentiert: Subjektattribute beschreiben und charakterisieren unabhängig von den Objekten die Subjekte. Objekte werden analog durch Objektattribute beschrieben. Beispiele für Subjektattribute sind der Identifikator, zugehörige Organisationseinheiten oder auch Rollenbezeichnungen. Objektattribute beschreiben ebenso bestimmte Objekteigenschaften. Die Zugriffsrechte werden zwischen den Subjekt- und Objektattributen definiert.

Das Grundmodell lässt sich nach [87] folgendermassen darstellen:

- S, O und A sind die Mengen der Subjekte ($s \in S$), Objekte ($o \in O$) und Aktionen ($a \in A$), welche auf den Objekten ausgeführt werden können[1].
- SA_i $(1 \leq i \leq I)$ und OA_k $(1 \leq k \leq K)$ sind vordefinierte Attribute für Subjekte und Objekte.

[1] Zur Vereinfachung wird davon ausgegangen, dass alle Aktionen auf allen Objekten ausführbar sind.

- $ATTRIBUTE(s)$ ist eine Zuordnungsrelation, die einem Subjekt s eine Menge an Subjektattributen zuweist: $ATTRIBUTE(s) \subseteq SA_1 \times SA_2 \times \ldots \times SA_I$.
- $ATTRIBUTE(o)$ ist eine Zuordnungsrelation, die einem Objekt o eine Menge an Objektattributen zuweist: $ATTRIBUTE(o) \subseteq OA_1 \times OA_2 \times \ldots \times OA_K$.

Die Entscheidung, ob ein Subjekt eine Aktion auf einem Objekt ausführen darf, wird auf eine Menge von allgemeinen Zugriffskontrollpolitiken, den *Policies*, zurückgeführt, die wiederum auf Subjekt- und Objektattributen basieren. Abstrakt formuliert lassen sich damit boolesche Zugriffsfunktionen wie folgt darstellen:

$$Policy : erlaubt(s, o, a) \leftarrow f(ATTRIBUTE(s), ATTRIBUTE(o), a).$$

Die Formulierung von Regeln als flexible Zugriffskontroll-*Policies* ermöglicht die von Subjekten und Objekten unabhängige Darstellung von Zugriffsrechten. Beispielsweise kann die *Policy*, dass der Besitzer eines Objekts dieses löschen darf, objektunabhängig und systemweit wie folgt dargestellt werden:

Beispiel 2.1 (Besitzerzugriff).
Policy $P1$:
$erlaubt(s, o, a) \leftarrow ((Identifikator(s) = Besitzer(o)) \wedge (a = \text{'Löschen'}))$ $\Diamond$

2.1.2. Mandatorische Zugriffskontrollmodelle

Die *mandatorische Zugriffskontrolle* ist ein Konzept für die Kontrolle und Steuerung von Zugriffsrechten auf Rechensystemen, bei der die Entscheidung über den Zugriff nicht auf Basis der Identität eines Subjekts und des Objekts gefällt wird, sondern aufgrund allgemeiner Regeln und Eigenschaften des Subjekts beziehungsweise des Objekts, siehe [29]. Voraussetzung ist, dass Subjekte niemals direkt sondern nur durch einen Referenzmonitor auf Objekte zugreifen können. Die Regeln des Referenzmonitors können nicht von individuellen Benutzern verändert werden. Da dieses Modell auf systemweit festgelegten Regeln basiert, wird es häufig auch als *Rule(set) based access control* bezeichnet.

Ursprünglich wurden die mandatorischen Zugriffskontrollmodelle vor allem im Bereich des Militärs entwickelt. Hier handelt es sich bei der Datenverarbeitung primär um sensible Informationen. Aber auch in anderen Bereichen, in denen sensible Informationen verarbeitet werden, wie beispielsweise die Nachrichtentechnik, wurden früh mandatorische Zugriffskontrollmodelle eingesetzt.

Der Sicherheitstufenansatz

Ein früher Vertreter dieses Konzepts ist das von D. E. Bell und L. J. LaPadula vorgestellte *Bell-LaPadula Modell*, siehe [12]. Dieser Ansatz verfolgt das Ziel der Vertraulichkeit und Kontrolle von Informationsfluss. Hierfür werden sogenannte *Sicherheitsstufen* verwendet. Die Sicherheitsstufen unterliegen einer partiellen „kleiner-gleich"-Ordnung, wobei je zwei Sicherheitsstufen über eine kleinste obere Schranke verfügen. Des weiteren existiert genau eine kleinste Sicherheitsstufe und die Anzahl aller Sicherheitsstufen ist endlich.

Jedem Benutzer b wird zunächst statisch eine Sicherheitsstufe b_s zugeordnet. Die Sicherheitsstufe o_s eines Objekts o ergibt sich aus seiner Geschichte auf folgende Art und Weise: Ein Benutzer b darf genau dann eine Leseoperation auf dem Objekt o ausführen, wenn die zum Zugriff aktuelle Sicherheitsstufe o_s des Objekts kleiner oder gleich der Sicherheitsstufe b_s des Benutzers ist (*no-read-up* Regel). Jeder Benutzer b darf eine Schreiboperation auf dem Objekt o ausführen. Ein Benutzer b darf eine Schreiboperation auf dem Objekt o ohne Veränderung der Sicherheitsstufe o_s ausführen, wenn die Sicherheitsstufe des Objekts größer oder gleich der Sicherheitsstufe des Benutzers ist. Ein Benutzer b, dessen Sicherheitsstufe b_s nicht kleiner oder gleich der Sicherheitsstufe o_s des Objekts o ist, bewirkt, dass sich die Sicherheitsstufe des Objekts auf den Wert der kleinsten oberen Schranke der Sicherheitsstufen o_s und b_s erhöht (*no-write-down* Regel).

Chinese-Wall Modell

Das *Chinese-Wall* Modell [23] ist aus den britischen Gesetzen für Börsenhändler entstanden. Es wurden Gesetze erlassen, die die Interessenskonflikte zwischen den Investmentbanken und Emissionsgeschäften verhindern sollten. Die Grundidee des Chinese-Wall Ansatzes basiert darauf, dass die zukünftigen Zugriffe eines Subjekts durch bereits durchgeführte Zugriffe beschränkt werden. Die zu schützenden Objekte des Systems werden als Baum strukturiert, der eine Zuteilung von den Objekten zu den zugehörigen Unternehmen und zu sogenannten *Conflict-of-Interest*-(CoI)-Klassen vornimmt. Für öffentlich zugängliche Objekte wird eine spezielle CoI-Klasse eingeführt. Die Zugriffskontrolle stellt sicher, dass ein Benutzer nur dann lesend auf ein Objekt zugreifen darf, wenn er das entsprechende Zugriffsrecht hat und er bis zu diesem Zeitpunkt auf kein anderes Objekt zugegriffen hat, das einem anderen Unternehmen aber der gleichen CoI-Klasse angehört und keine öffentlichen Informationen enthält. Ein Benutzer darf schreibend auf ein Objekt zugreifen, wenn er das entsprechende Zugriffrecht hat und er bis zu diesem Zeitpunkt nur Lesezugriffe auf solche Objekte hatte, die zum gleichen Unternehmen gehören oder nur öffentliche Informationen enthalten.

2.2. Kompositionale Zugriffskontrollmodelle

Vielfach ist es sinnvoll, in einem Rechensystem mehrere Zugriffskontrollpolitiken zu kombinieren, welche unabhängig voneinander spezifiziert worden sind. In einem solchen System soll zwar die Durchsetzung solch komponierter Zugriffskontrollpolitiken einheitlich stattfinden, aber die Spezifizierung und Wartung der einzelnen Komponentenpolitiken sollen trotzdem autonom und unabhängig voneinander geschehen. In der neueren Literatur finden sich hierzu einige Ansätze, die im folgenden kurz vorgestellt werden.

2.2.1. Ansatz nach Bonatti: eine Einführung

In Bonatti et al. [19] wird eine allgemeine kompositionale Vorgehensweise zur Spezifizierung und zur Durchsetzung von Zugriffskontrollpolitiken vorgeschlagen. Dieser Vorschlag entwickelt eine algebraische kompositionale Struktur auf der Klasse der Zugriffskontrollpolitiken und grundlegenden Operationen. Die Zugriffskontrollpolitiken werden von beliebigen administrierenden Teilnehmern autonomer Organisationseinheiten spezifiziert. Die Zugriffskontrollpolitiken sind gegeben durch Mengen von Erlaubnisse der Form (s, o, a). Jede Erlaubnis (s, o, a) drückt die Erlaubnis eines Subjekts s aus, eine *Aktion* a auf einem Objekt o ausführen zu dürfen. Die grundlegenden Kompositionsoperatoren sind Vereinigung, Durchschnitt, Differenz, Selektion und regelbasierte Ableitung.

Bonatti et al. stützen sich auf das Konzept der diskretionären Zugriffskontrolle, die im Abschnitt 2.1.1 vorgestellt wurde. Nach Bonatti et al. sind die Mengen der Subjekte (S) und die Mengen der Objekte (O) gegeben. Der Zugriff auf ein Objekt wird durch die Mengen der Aktionen (A) bestimmt. Es werden keinerlei Einschränkungen oder Annahmen bezüglich der Elemente dieser Mengen vorgenommen. Das bedeutet beispielsweise, dass keine Aussagen darüber getroffen werden, welche Aktionen auf den Objekten zu kontrollieren sind. Abhängig von der zugrunde liegenden Organisationsstruktur können Subjekte Benutzer, Gruppen, Rollen oder auch Anwendungen sein. Objekte können beispielsweise Dateien, XML-Dokumente oder auch Klassen sein.

Grundannahmen

Ein Grundautorisierungsterm ist ein Tripel der Form $<s, o, a>$. Der Bezeichner s repräsentiert ein Element der Menge S, der Bezeichner o ein Element der Menge O, der Bezeichner a ein Element der Menge A. Eine Zugriffskontrollpolitik P besteht aus einer Menge von Grundautorisierungstermen. Diese Terme definieren die als erlaubt

angesehenen Zugriffsoperationen bezüglich der Zugriffskontrollpolitik P. Eine Zugriffskontrollpolitik repräsentiert also die Durchsetzung einer Autorisierungsspezifikation, unabhängig von der tatsächlichen Spezifikationssprache, in der diese Spezifikation verfasst worden ist.

Ein Grundautorisierungsterm $<s, o, a>$ einer Zugriffskontrollpolitik P kann als subjektbezogene *capability* interpretiert werden: Ein begünstigtes Subjekt s ist innerhalb dieser Zugriffskontrollpolitik autorisiert, die *capability* $<o, a>$ in Anspruch zu nehmen.

Syntax und Semantik der Algebra Die Syntax der Ausdrücke einer kompositionalen Zugriffskontrollpolitik ist nach [19] durch die folgende Grammatik gegeben:

$$\begin{array}{lll} \text{E} & ::= & \mathbf{id} \mid E + E \mid E \& E \mid E - E \mid E \hat{} C \mid o(E, E, E) \mid E * R \mid T(E) \mid (E) \\ \text{T} & ::= & \tau\ \mathbf{id}.\text{T} \mid \tau\ \mathbf{id}.\text{E} \end{array}$$

Der Bezeichner **id** steht für eine Zugriffskontrollpolitik. Mit dem Nichtterminalzeichen E werden die jeweiligen Ausdrücke beschrieben. Der Bezeichner T steht für ein so genanntes Templatekonstrukt. Die Bezeichner C und R sind Elemente der Sprachen $\mathcal{L}_{acon}$ und $\mathcal{L}_{rule}$, die im folgenden Abschnitt 2.2.1 erläutert werden. Die Priorität der einzelnen Kompositionsoperatoren ist nach [19] in der Tabelle 2.1 definiert.

Operation	Priorität
τ	0
.	1
$+, \&, -$	2
$\star, \hat{}$	3

Tabelle 2.1.: Prioritäten der Kompositionsoperatoren

Formal stellt die Semantik eine Funktion dar, die jedem Ausdruck eine Menge von Grundautorisierungstermen zuordnet. Der einfachste darstellbare Ausdruck ist ein Bezeichner P [2] für eine Zugriffskontrollpolitik. Dieser Bezeichner P wird durch eine partielle Zuordnung e einer Menge von Grundautorisierungstermen zugeordnet $e(P)$. Im Folgenden wird statt der Zuordnung $e(P)$ die Notation $[[P]]_e$ verwendet.

Syntax und Semantik der Kompositionsoperatoren

Addition (+) Der Additionsoperator vereinigt zwei Zugriffskontrollpolitiken P_1 und P_2

[2] Der Bezeichner P wird für den in der Grammatik auf Metaebene eingeführten Bezeichner **id** eingesetzt.

$$[[P_1 + P_2]]_e = [[P_1]]_e \cup [[P_2]]_e.$$

Die Komposition zweier Zugriffskontrollpolitiken P_1 und P_2 durch die Addition kann benutzt werden, wenn die Inanspruchnahme einer *capability* $<o, a>$ durch s genau dann erlaubt werden soll, wenn der Grundautorisierungsterm $<s, o, a>$ Element einer der beiden Zugriffskontrollpolitiken P_1 oder P_2 ist.

Durchschnitt (&) Der Durchschnittsoperator bildet den Durchschnitt zweier Zugriffskontrollpolitiken P_1 und P_2, indem er auf den Mengen der Autorisierungsterme den Durchschnitt bildet:

$$[[P_1 \& P_2]]_e = [[P_1]]_e \cap [[P_2]]_e.$$

Im Gegensatz zur Komposition durch Addition ist bei der Komposition durch die Durchschnittsbildung die Inanspruchnahme einer *capability* $<o, a>$ durch s genau dann erlaubt, wenn der Grundautorisierungsterm sowohl Element der Zugriffskontrollpolitik P_1 als auch Element der Zugriffskontrollpolitik P_2 ist.

Subtraktion (-) Der Subtraktionsoperator schränkt eine gegebene Zugriffskontrollpolitik P_1 ein, indem er die Autorisierungen eliminiert, die in einer zweiten Zugriffskontrollpolitik P_2 angegeben sind:

$$[[P_1 - P_2]]_e = [[P_1]]_e \setminus [[P_2]]_e.$$

Mittels der Komposition durch Subtraktion ist es möglich, Ausnahmen für eine bestehende Zugriffskontrollpolitik zu treffen.

Abschluss (*) Die Algebra schließt eine gegebene Zugriffskontrollpolitik P_1 unter einer gegebenen Menge von Ableitungsregeln R ab. Die Regeln R sind in einer parametrisierbaren Sprache $\mathcal{L}_{rule}$ definiert. Diese Sprache kann beispielsweise Hornklauseln benutzen, die auf Autorisierungstermen und einfachen Prädikaten aufbauen:

$$[[P_1 * R]]_e = \mathit{closure}(R, [[P_1]]_e),$$
$$* : \wp(S \times O \times A) \times \wp(\mathcal{L}_{rule}) \to \wp(S \times O \times A).$$

Ableitungsregeln können beispielsweise Weitergaben von Autorisierungen regeln. Solche Weitergaben können bei der Abwesenheit eines Subjekts wichtig sein. Ableitungsregeln, die Weitergaben definieren, können aufgrund bestehender Hierarchien auf den Subjekten bzw. Objekten definiert werden. Der Abschluss einer Zugriffskontrollpolitik P_1 bezüglich einer Regelmenge R produziert eine Zugriffskontrollpolitik P, die all jene Autorisierungen enthält, die aus P_1 aufgrund R abgeleitet werden können.

Selektion ($\hat{}$) Die Algebra schränkt eine gegebene Zugriffskontrollpolitik P_1 bezüglich einer Bedingung C ein. Die Bedingung C ist in einer parametrisierbaren Sprache $\mathcal{L}_{acon}$ definiert:

$$[[P_1 \hat{} C]]_e = \{<s,o,a> | <s,o,a> \in [[P_1]]_e \ , <s,o,a> \textit{ satisfy } C\}, C \in \mathcal{L}_{acon}.$$

Der Selektionsoperator kann dann eingesetzt werden, wenn die Inanspruchnahme eines Privilegs $<o,a>$ durch s eingeschränkt werden soll. Die Begünstigten s der Zugriffskontrollpolitik P_1, die nicht die angegebene Bedingung C erfüllen, können das Privileg $<o,a>$ nicht im Kontext der Selektion nutzen.

Ersetzung (o) Der Ersetzungsoperator ersetzt einen Teil einer bestehenden Zugriffskontrollpolitik P_1 mit einem passenden Teilstück einer zweiten Zugriffskontrollpolitik P_2. Der Teil der Grundautorisierungsterme, der ersetzt werden soll, wird in einer dritten Zugriffskontrollpolitik P_3 definiert. Formal kann der Ersetzungsoperator durch die schon vorgestellten Kompositionsoperatoren definiert werden:

$$[[o(P_1, P_2, P_3)]]_e = [[(P_1 - P_3) + (P_2 \& P_3)]]_e.$$

2.2.2. Wijesekera und Jajodia

In der Arbeit *A propositional policy algebra for access control* veröffentlichen Wijesekera und Jajodia [84] eine Algebra zur Spezifizierung kompositionaler Zugriffskontrollpolitiken. Kernpunkt des Ansatzes bilden sogenannte Zustände (*states*), die eine Abbildung von logischen Aussagen (*propositions*) auf Mengen von elementaren Erlaubnissen der Art (s,o,a) darstellen, für die die logischen Aussagen gültig sind. Von einem anfänglichem *state* ausgehend, der per definitionem auf eine leere Erlaubnismenge abbildet, werden durch die Ausführung spezifizierter Kompositionsoperatoren (weitere) Erlaubnisse hinzugefügt. Die Kompositionsoperatoren unterscheiden sich in *interne Operatoren* und *externe Operatoren*. Im Gegensatz zu internen Operatoren haben externe Operatoren

keinen Einfluss auf die innere Struktur der Erlaubnismenge. Die internen Operatoren entsprechen weitgehend den in Bonatti et al. verwendeten, vergleiche Abschnitt 2.2.1. Die tatsächliche Operation wird nur dann durchgeführt, wenn die Menge der möglichen Zugriffe dabei konsistent bleibt. Dies bedeutet, dass bei Widersprüchen innerhalb einer Erlaubnismenge die Operation nicht (vollständig) durchgeführt wird.

2.2.3. Woo und Lam

In der Arbeit *Authorizations in distributed systems: A new approach* [86] erarbeiten Woo und Lam ein Rahmenwerk zur Spezifizierung von Autorisierungsregeln, welches auf einer *nichtklassischen Logik* basiert. Die Autoren entwickeln eine Default Logik (*default logic*), siehe [65], mit der sogenannte Politikbasen, die *policy bases* spezifiziert werden. Eine Politikbasis ist eine Menge von Autorisierungsregeln, die wiederum bestimmte Eigenschaften aufweisen müssen. Je nach Eigenschaft der Menge spezifiziert eine solche Politikbasis, dass die genannten Subjekte Zugriffsrechte für die genannten Objekte haben oder dass explizite Zugriffsverbote für die genannten Subjekte vorliegen.

Im Prinzip ist die in [86] eingeführte Politikbasis eine alternative Darstellung von klassischen Zugriffskontrolllisten. Die Durchsetzung der Zugriffskontrollpolitiken erfolgt durch die Übersetzung der Politikbasen in entsprechende Logikprogramme. Zur Implementierung dieser Zugriffskontrolle schlagen die Autoren den Einsatz sogenannter *policy server* vor. Jeder Server unterhält eine lokale Kopie der aktuellen Politikbasis.

2.2.4. Siewe et al.

Die Arbeitsgruppe um Hussein Zedan hat ein logikbasiertes Rahmenwerk zur Spezifikation kompositionaler Zugriffskontrollpolitiken entwickelt. Zu diesem Ansatz wurden verschiedene Schriften publiziert, von denen [47, 74, 75] hier ausgewählt wurden. Als Grundlage für den Ansatz dient die lineare Intervalltemporallogik ITL; siehe [8] als einleitenden Überblick. Diskrete Zeitpunkte entsprechen in ITL diskreten Systemzuständen, welche zu endlichen Intervallen zusammengefasst werden. Die ITL-Semantik basiert auf diesen Intervallen I, die eine (un)endliche Sequenz von Systemzuständen bzw. Belegungen $I = (\sigma_0, \ldots)$ beschreiben. Jede Belegung σ_i bildet Variablen auf Werte ihrer Domäne ab, wobei zwischen statischen Variablen und Zustandsvariablen unterschieden wird. Statische Variablen haben eine für ein Intervall feste Belegung, während Zustandsvariablen zu jeden Zeitpunkt eine neue Belegung annehmen können. Die Autoren verwenden neben Booleschen Operatoren (z.B. $\neg$, $\wedge$) und den üblichen Temporaloperatoren (z.B. *always, eventually, next*), insbesondere den so genannten *always-followed-by* Operator ($\mapsto$), um Politikregeln zu spezifizieren. Für

eine LTL-Formel f und eine Zustandsformel w kann $f \mapsto w$ umschrieben werden als:

> *Immer wenn f innerhalb eines endlichen Intervalls gilt, dann gilt w in dem letzten Zustand dieses Intervalls.*

Im Hinblick auf Zugriffskontrolle, wird eine *always-followed-by* Regel so interpretiert, dass eine Entscheidungskomponente des Systems w entscheidet, sofern zuvor das Verhalten f beobachtet wurde. Eine Zugriffskontrollpolitik wird durch eine Menge von Regeln gegeben. Das folgende Beispiel definiert eine Zugriffskontrollpolitik, welche besagt, dass allen Subjekten einer im System definierten Subjektmenge S das Lesen aller Objekte einer im System definierten Objektmenge O erlaubt ist:

Beispiel 2.2 (generelle Leseerlaubnis). $true \mapsto Autho^{+}(S, O, read)$ ◊

Da die Autoren das Spezifizieren expliziter Erlaubnisse ($Autho^{+}(\ldots)$) und expliziter Verbote ($Autho^{-}(\ldots)$) erlauben, kann eine Konfliktresolutionspolitik der folgenden Art hilfreich sein:

Beispiel 2.3 (Konfliktresolution).
$\lceil \neg Autho^{-}(S, O, read) \wedge Autho^{+}(S, O, read) \rceil^{0} \mapsto Autho(S, O, read)$ ◊

2.3. Zugriffskontrollmodelle für verteilte Systeme

Den oben vorgestellten Zugriffskontrollmodellen gemein ist, dass sie sich am Aufbau fester Organisationseinheiten im System orientieren. Moderne verteilte Rechensysteme erfüllen diese Voraussetzungen nicht. Insofern ist es evident, obige Zugriffskontrollmodelle für verteilte Rechensysteme zu verfeinern oder aber neue (Varianten der) Zugriffskontrollmodelle zu entwickeln. Beginnend von dem in Abbildung 1.1 vorgestellten Basismechanismus des Referenzmonitors müssen vor allem die Komponenten einer Zugriffskontrolle geeignet dezentralisiert werden. In den Standards [78, 46] der *International Telecommunication Union* und der *International Organisation for Standardization* wird eine domänenunabhängige Referenzarchitektur für den Bereich der Zugriffskontrolle vorgestellt. Ein zentraler Punkt in beiden Standards ist die Trennung zwischen fachfunktionalen und sicherheitsfunktionalen Komponenten (*separation of concerns*). Zusätzlich wird vorgeschlagen, ein *verteiltes Referenzmonitorprinzip* zu verfolgen: Der Kernbereich der Zugriffskontrolle wird auf die *Zugriffskontroll-Entscheidungsfunktion* (*Access Control Decision Function*, ACDF) und die *Zugriffskontroll-Durchsetzungsfunktion* (*Access Control Enforcement Function*, ACEF) aufgeteilt. Dabei wird die Zugriffskontroll-Durchsetzungsfunktion der eigentlichen Fachfunktionalität vorgeschaltet mit dem Ziel, unberechtigte Zugriffe zu verhindern. Die tatsächliche Entscheidung, ob ein Zugriffswunsch erlaubt wird, wird von der

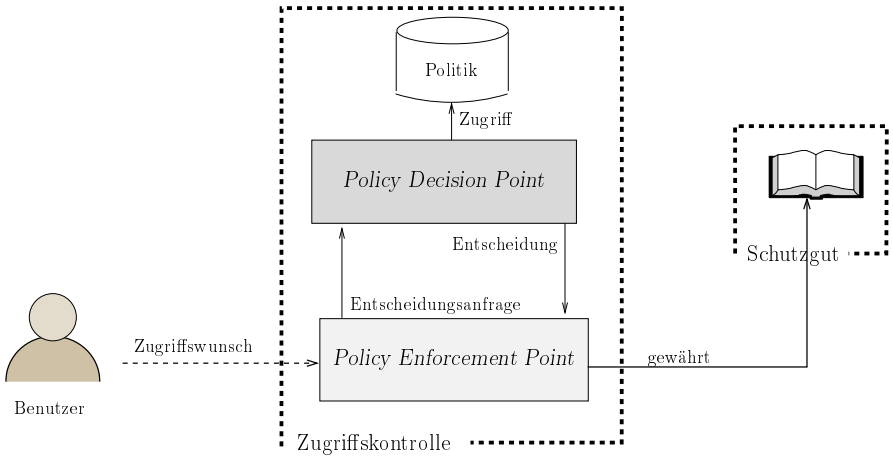


Abbildung 2.1.: Verteiltes Referenzmonitorprinzip für offene Rechensysteme.

Zugriffskontroll-Durchsetzungsfunktion an die oft örtlich getrennte Zugriffskontroll-Entscheidungsfunktion ausgelagert. Diese prüft den Zugriffswunsch und sendet eine Freigabe oder Zurückweisung an die Zugriffskontroll-Durchsetzungsfunktion zurück.

In den Referenzen [78, 46] finden sich keine Angaben über die Anzahl oder Positionierung der beiden Zugriffskontrollkomponenten in einem Rechensystem. Ein in sich geschlossenes Rechensystem nutzt innerhalb seiner Systemgrenzen integrierte Zugriffskontroll-Durchsetzungspunkte und Zugriffskontroll-Entscheidungspunkte. Hierdurch besteht eine natürliche Vertrauensbeziehung der Komponenten untereinander. In offenen und dezentralen Rechensystemen hingegen ist es ein Ziel, diese Zugriffskontrollkomponenten systemübergreifend anzubieten. Normalerweise befinden sich beide Zugriffskontrollkomponenten in verteilten Systemen innerhalb der Schutzdomäne des zu schützenden Objekts. Das Vertrauen eines anfragenden Subjekts in die größtmögliche Geheimhaltung seiner personenbezogenen Daten, die zur Durchführung der Zugriffskontrolle den verschiedenen Komponenten offengelegt werden muss, ist ein aktuelles Forschungsthema.

Die Begriffsbildung im Englischen hat sich in den letzten Jahren verändert. In aktueller Literatur und Standards, siehe beispielweise [39, 41] werden die Begriffe *Policy Enforcement Point* (PEP) anstelle von *Access Control Enforcement Function* und *Policy Decision Point* (PDP) anstatt *Access Control Decision Function* benutzt.

Die Abbildung 2.1 veranschaulicht ein verteiltes Referenzmonitorprinzip.

im Folgenden stellen wir aktuelle Zugriffskontrollmodelle vor, die für verteilte und offene Rechensysteme entworfen wurden.

2.3.1. Minsky et al.

Naftaly Minsky hat mit dem *Law Governed Interaction* (LGI) Mechanismus einen umfassenden Ansatz zur dezentralen Zugriffskontrolle in verteilten Systemen entwickelt. Als Referenz dienen folgende, ausgewählte Publikationen: [9, 10, 57]. Die Arbeitsgruppe um Naftaly Minsky hat die Anwendbarkeit des LGI Mechanismus für spezifische Anwendungsfälle gezeigt, als auch dessen praktische Relevanz mit der Java-basierten Middleware *Moses* [56]. Minsky legt die Begriffe einer *Koalition* C und einer zugehörigen *Koalitionspolitik* P_C zugrunde. Jede Koalition C besteht aus einer dynamischen Menge von teilnehmenden Entitäten $\{E_1, \ldots, E_n\}$, die für die Interaktion innerhalb der Koalition internen, weitgehend autonom spezifizierten Politiken P_i unterliegen. Jede Koalition $C = \{E_1, \ldots, E_n\}$ wird durch eine lose Komposition von Politiken reguliert: $[P_C, \{P_i\}]$.

Als essentiell in diesem Ansatz wird das *Flexibilitätsprinzip* vorgestellt, welches sicher stellen soll, dass jede teilnehmende Einheit ihre interne Zugriffskontrollpolitik weitgehend autonom spezifizieren und korrekt durchsetzen kann. Um Konflikte zwischen den lokalen Politiken und der Koalitionspolitik zu vermeiden, verfolgt der LGI Ansatz die Idee der Verfeinerung. Es wird eine Politikenhierarchie definiert, die besagt, dass alle internen Politiken eine Verfeinerung der globalen Koalitionspolitik sein müssen. Verfeinerung im Sinne des LGI Ansatzes ist als Vererbung im Sinne der objektorientierten Programmierung zu verstehen. So gedeutet, stellt eine globale Koalitionspolitik eine abstrakte Klasse dar, welche von den internen Politiken als konkrete Klassen implementiert wird. Somit ist zum einen gewährleistet, dass die von der globalen Koalitionspolitik aufgestellten (Meta)Regeln eingehalten werden und zum anderen, dass die internen Politiken untereinander konfliktfrei sind.

Der Ansatz unterliegt diversen Annahmen: Jede teilnehmende Entität E_i verwendet einen lokalen und vertrauenswürdigen Referenzmonitor. Dies bedeutet, dass jede lokale Administrierungsstelle die korrekte Durchsetzung der zugehörigen internen Politik P_i und (somit) der globalen Koalitionspolitik P_C sicher stellt. Die ausschließliche Kommunikation der Entitäten über ihre lokalen Verwalter gewährleistet einen sicheren Nachrichtenverkehr. Die Vertrauenswürdigkeit der lokalen Verwalter, beziehungsweise deren korrekten Implementierung im Sinne eines vom Initiator zur Verfügung gestellten Templates, wird durch die Zertifizierung einer vertrauenswürdigen dritten Partei belegt. Alle teilnehmenden Entitäten müssen sich auf die Anerkennung einer zertifizierenden, vertrauenswürdigen dritten Partei einigen.

2.3.2. Pretschner et al.

Die Arbeitsgruppe um David Basin erarbeitete am Lehrstuhl für Informationssicherheit an der ETH Zürich Kontrollmechanismen, mit denen es ermöglicht werden soll, eine Nutzungskontrolle von Daten in verteilten Rechensystemen durchzusetzen, siehe [44, 62]. Dieser Ansatz der Nutzungskontrolle liegt insofern orthogonal zu der in dieser Arbeit erarbeiteten Zugriffskontrolle, als dass nicht der Zugriff auf zu schützende Objekte innerhalb einer Schutzdomäne kontrolliert wird, sondern (Kopien der) zu schützenden Objekte an Benutzer ausgegeben werden. Diese Art der Zugriffskontrolle, die sogenannte *verteilte Nutzungskontrolle* wurde originär in den Arbeiten von Ravi Sandhu eingeführt, siehe [61].

Die aktuelleren Arbeiten der verteilten Nutzungskontrolle, siehe [44, 63], fokussieren zwei Kernpunkte. Zum einen muss durch traditionelle Zugriffskontrolle nachgewiesen werden können, ob ein Benutzer autorisierten Zugriff auf ein Objekt hat. Dieses wird durch sogenannte *Provisionen* spezifiziert. Zum anderen muss der Benutzer bestimmte, mit dem Objekt verknüpfte Verpflichtungen, sogenannte *obligations*, einhalten. Eine Nutzungskontrollpolitik für ein Objekt beinhaltet somit Provisionen und *obligations*. Die Einhaltung der *obligations* ist die eigentliche Herausforderung bei der verteilten Nutzungskontrolle. Die Arbeiten schlagen zwei verschiedene Mechanismen vor: Kontrollmechanismen, wie sie beispielsweise für *Digital Rights Management* eingesetzt werden, und Beobachtungsmechanismen, die durch bei den Benutzern lokal beherbergte Monitoren durchgesetzt werden sollen. Die Arbeit [63] betrachtet insbesondere die Problematik, *obligations* nicht statisch sondern dynamisch durchsetzen zu wollen und schlägt hierfür ein Rahmenwerk vor, welches eine Aktualisierung beziehungsweise *re-distribution* von *obligations* seitens der Dienstanbieter erlaubt.

2.4. zustandsabhängige Zugriffskontrollmodelle

Die bisher genannten Zugriffskontrollmodelle behandeln die Vergabe und Kontrolle von Zugriffsrechten für einzelne Funktionalitäten angebotener Dienste. Einige Vorschläge zur Zugriffskontrolle einzelner Funktionalitäten im Anwendungsgebiet strukturierter Dienste finden sich in [14, 20, 28, 48, 51]. Werden allerdings komplexe strukturierte Dienste im Bereich virtueller Organisationen betrachtet, so müssen Zugriffskontrollpolitiken auch für komplexe Folgen der Funktionalitäten strukturierter Dienste entwickelt werden. Dieses Konzept der Zustandsabhängigkeit findet sich in sogenannten zustandsabhängigen Zugriffskontrollansätzen wieder, von denen im Folgenden einige Vertreter vorgestellt werden.

2.4.1. Li und Wang

Ninghui Li und Qihua Wang entwickelten in [53, 54] einen algebraischen Ansatz, mit dem sogenannte *high-level* Sicherheitspolitiken spezifiziert werden können. Die Arbeit wurde von dem Prinzip der Funktionstrennung (*Separation of Duties*) geprägt. Grundlegend kann mit diesem Ansatz eine Zugriffskontrolle durchgesetzt werden, die ausdrückt, dass die Erledigung einer bestimmten Aufgabe k Benutzer des Systems benötigt, welche zusätzlich bestimmten Rollen zugewiesen sein müssen.

Ähnlich dem algebraischen Ansatz aus [84] spezifizieren Li und Wang Politiken nicht als Mengen, sondern als Abbildungen von Aufgaben (*tasks*) auf Mengen von Begünstigten. Die Aufgaben werden als Sequenzen sogenannter *steps* definiert. Benutzer des Systems können die einzelnen *steps* einer Aufgabe ausführen, wobei in der Implementierung des Zugriffskontrollsystems nachgehalten wird, welcher Begünstigte welche *steps* ausgeführt hat. In Abhängigkeit dieser protokollierten Historie einer Aufgabe wird über Zugriffswünsche der Benutzer entschieden: Bevor ein Benutzer die Erlaubnis zur Ausführung eines *steps* erhält, prüft die Zugriffskontrolle des Systems, ob nach Ausführung des *steps* die Aufgabe noch erfüllt werden kann.

2.4.2. Eckert

In [32] ist ein Ansatz zur Spezifizierung und Durchsetzung zustandsabhängiger Zugriffskontrollpolitiken vorgestellt worden. Der Begriff „zustandsabhängig“ bezieht sich in diesem Ansatz auf die aktuelle Position innerhalb einer beliebigen *Handlungsabfolge*. Zur Durchsetzung der zustandsabhängigen Zugriffskontrollpolitik ist die Festlegung von erlaubten Handlungsabfolgen für die Objekte des zugrunde liegenden Rechensystems notwendig. Die Handlungsabfolgen, sogenannte *Protokolle*, werden durch reguläre Ausdrücke spezifiziert und können zur Durchsetzung in endliche, deterministische Automaten überführt werden. In zentralen Rechensystemen erlaubt diese Modellierung die Durchsetzung der Zugriffskontrolle durch eine Realisierung des entsprechenden Automaten in einem zentralen Referenzmonitor.

Für die Durchsetzung in verteilten Rechensystemen wird in [17, 32] der Einsatz von *capabilities* vorgeschlagen. Als „verbrauchbare“ *capability* wird das Tupel (A, q) betrachtet, wobei A den Automaten und q einen Zustand in diesem beschreibt. Damit ein Subjekt eine Zugriffsoperation auf einem Objekt initiieren darf, muss das Subjekt „in Besitz“ des entsprechenden Automaten sein, der sich im Zustand q befindet. Um die „Besitzerschaft“ eines Automaten zu realisieren, wird in diesem Ansatz ein *token*-Konzept vorgeschlagen, nach dem der Automat ähnlich einem *token* unter den Subjekten weitergereicht wird.

2.4.3. Botha und Eloff

CoSAWoE ist ein zustandsabhängiges Zugriffskontrollmodell, welches von Reinhardt Botha entwickelt wurde, siehe [21]. Dieses Modell basiert auf rollenbasierter Zugriffskontrolle. Der Kontext dieses Modells sind sogenannte *Workflowsysteme*. Es geht um die zeitliche Reihenfolge und Abarbeitung verschiedener Aufgaben, der *tasks*. Ein *workflow* wird durch eine *Prozessdefinition* festgelegt, die aus mehreren, abhängigen Aufgaben besteht. Es existiert eine Rollenzuweisungsfunktion, die nach Instantiierung des Prozesses festlegt, welche Rolle welche Aufgabe ausführen darf. Das Grundkonzept legt sogenannte *sessions* zugrunde, für deren Dauer eine Zuweisung Rolle – Aufgabe erhalten bleibt. In diesem Modell wird eine Session für nur eine Aufgabe verwendet. Ein Benutzer innerhalb einer bestimmten Rolle erhält die notwendigen Rechte zur Erfüllung seiner Aufgabe nur für die Dauer der Bearbeitung dieser. Sobald ein Benutzer aufhört, an einer Aufgabe zu arbeiten, werden ihm die entsprechenden Zugriffsrechte wieder entzogen.

2.4.4. Chander

Chander et al. [25] entwerfen eine Modellierung diskretionärer Zugriffskontrolle für *capability*-basierte Systeme. Die Modellierung basiert auf sogenannten globalen *abstract system states*, welche den jeweils aktuellen Systemzustand widerspiegeln. Ein Systemzustand wird durch eine Menge von Zugriffsrechten für die Subjekte und Objekte des Systems gegeben und durch eine Menge erlaubter Delegationen, in der festgehalten wird, welches Subjekt oder Objekt welches Zugriffsrecht delegieren darf. Durch sogenannte *transitions* kann ein Systemzustand in einen anderen überführt werden. Die Zustandsübergänge werden durch eine Menge erlaubter, ebenfalls global zugänglicher *actions* definiert, mit denen die Dynamik des Systems modelliert werden kann. Eine Aktion kann beispielsweise die Vergabe eines Zugriffsrechtes oder das Hinzufügen neuer Objekte sein. Die tatsächliche Zugriffskontrolle findet durch die Anwendung zuvor spezifizierter, globaler Ableitungsregeln statt, die für den aktuellen Systemzustand ableiten lassen, ob ein von einem konkreten Subjekt angefragtes Zugriffsrecht als erlaubt angesehen wird.

Die Autoren zeigen, dass sowohl die klassische diskretionäre Zugriffskontrolle, als auch die *capability*-basierte diskretionäre Zugriffskontrolle mit diesem Rahmenwerk modelliert werden kann. Obwohl dieser Ansatz nicht explizit für zustandsabhängige Zugriffskontrolle entworfen wurde, kann die konzeptuelle Modellierung der Systemzustände hierfür ausgenutzt werden.

2.5. Zusammenfassung und Bewertung der aktuellen Literatur

Wie in den vorherigen Abschnitten dargestellt, lässt sich die (aktuelle) Literatur für Zugriffskontrollmodelle unter verschiedenen Gesichtspunkten beleuchten. In dieser Arbeit werden wir ein Rahmenwerk für eine verteilte Zugriffskontrolle entwerfen, in dem sowohl kompositionale als auch zustandsbasierte Aspekte eingebunden werden können. Vor diesem Hintergrund haben wir in diesem Kapitel die folgende Klassifizierung der betrachteten Zugriffskontrollmodelle vorgenommen:

1. Die *klassische Zugriffskontrolle*, zu deren Vertreter Varianten der diskretionären Zugriffskontrolle und mandatorischen Zugriffskontrolle vorgestellt wurden, siehe hierzu Abschnitt 2.1.

2. Die *kompositionale Zugriffskontrolle* ermöglicht in einem Rechensystem mehrere, unabhängig voneinander spezifizierte Zugriffskontrollpolitiken zu kombinieren. Einige Vertreter in der Literatur wurden in Abschnitt 2.2 vorgestellt.

   Detaillierter eingegangen wurde auf den Ansatz von Bonatti et al. [19], welcher den Ausgangspunkt für die im später folgenden Kapitel 6 entwickelte kompositionale und zustandsabhängige Zugriffskontrollpolitik ist.

3. Zwei wichtige Ansätze für *Zugriffskontrollpolitiken für verteilte Systeme* wurden in Abschnitt 2.3 vorgestellt. Unabhängig davon, ob einige der anderen Zugriffskontrollmodelle für verteilte Rechensysteme angepasst werden können, sind die hier vorgestellten Ansätze explizit für verteilte und offene Systeme konzipiert worden.

   Der Ansatz [62, 63] weist Merkmale zustandsabhängiger Zugriffskontrolle im Hinblick auf das Durchsetzen der *obligations* auf. Die eigentliche Zugriffskontrolle erfolgt zustandslos, während der Beobachtungsmechanismus zur Verpflichtungskontrolle eine zustandsähnliche Historie herausgegebener Objekte verwalten muss.

4. Schließlich wurden einige Vertreter der *zustandsabhängigen Zugriffskontrolle* in Abschnitt 2.4 vorgestellt.

Die unter 2. und 3. betrachteten Zugriffskontrollmodelle erfüllen jeweils den Aspekt der Kompositionalität und der Verteiltheit. Allerdings behandeln diese zustandslosen Zugriffskontrollmodelle (vorrangig) die Vergabe und Kontrolle von Zugriffsrechten für einzelne Zugriffsoperationen auf atomaren Objekten. Einige dieser Ansätze sind sogar für geschlossene Rechensystemen entworfen worden. Betrachten wir komplexe Anwendungen in offenen Rechensystemen, so müssen die Zugriffskontrollpolitiken

allerdings auch für komplexere, strukturierte Zugriffsrechte spezifiziert werden können. Der hier vorgestellte Ausschnitt aus der Literatur für zustandsabhängige Zugriffskontrollmodelle (4.) greift für strukturierte Zugriffsrechte ausschließlich im Hinblick auf sequenzielle Zugriffsoperationen auf atomaren Objekten.

In dieser Arbeit werden die Konzepte kompositionaler Zugriffskontrolle gemeinsam mit den Konzepten zustandsbasierter Zugriffskontrollmodelle für die Überwachung und Kontrolle strukturierter Dienste angewendet. Vor dem Hintergrund virtueller Organisationen, welche beliebig und systemübergreifend kombinierte Dienste anbieten, wird ein Rahmenwerk für Zugriffkontrolle entwickelt, welches die Kontrolle und Überwachung in solchen Szenarien ermöglicht.

Abschließend gibt die Tabelle 2.2 einen gerafften Überblick über die in diesem Kapitel vorgestellen Ansätze. Die Arbeiten werden im Hinblick darauf verglichen, ob sie eine Basis haben, kompositionale Rahmenwerke für Zugriffskontrolle in verteilten Systemen für strukturierte Dienste anbieten zu können. Als Vergleichspunkte nehmen wir die in Abschnitt 1.2 aufgestellten Anforderungen an eine Zugriffskontrolle. So wird betrachtet,

- wie der jeweilige Ansatz eine Zugriffskontrollpolitik repräsentiert (Repräsentation),
- ob der Ansatz kompositionale Eigenschaften im Hinblick auf die Kontrolle strukturierter Dienste aufweist (Komposit.),
- ob der Ansatz zustandsabhängige Konstrukte aufweist (Zustandsbas.) und
- ob der Ansatz explizit für verteilte Systeme entworfen wurde (Verteilt).

Betrachten wir einen Einsatz der Zugriffskontrolle in virtuellen Organisationen, welche nicht festgelegte Mengen von Objekten und Subjekten verwalten muss, so ist weiterhin relevant,

- ob der jeweilige Ansatz die Möglichkeiten der Paramatrisierung (Parameter) und
- ob der Ansatz die Möglichkeit der Rollennutzung (Rolle) bietet.

Klassifizierung	Ansatz	Repräsentation
kompositionale Zugriffskontrollmodelle	[19] Bonatti et al.: [84] Wijesekera/Jajodia: [86] Woo/Lam: [47] Siewe et al.:	Erlaubnismenge Abbildungen Regelmenge (DL) Regelmenge (ITL)
explizit verteilte Zugriffskontrollmodelle	[57] Minsky/Ungureanu: [62, 63] Pretschner et al.:	Koalitionspolitik Provisionen,*obligations*
zustandsabhängige Zugriffskontrollmodelle	[53] Li/Wang: [32] Eckert: [21] Botha/Eloff: [25] Chander et al.:	Abbildung reguläre Ausdrücke Rollenzuweisung Zustandsübergänge

Tabelle 2.2.: Klassifizierung und Repräsentation der Zugriffskontrollansätze.

Ansatz	Komposit.	Zustandsbas.	Verteilt	Parameter	Rollen
[19]:	ja	nein	nein	ja	nein
[84]:	ja	ja	nein	nein	nein
[86]:	nein	nein	ja	ja	nein
[47]:	ja	nein	nein	ja	nein
[57]:	nein	nein	ja	nein	implizit
[62, 63]:	nein	nein,*ja*	ja	ja	ja
[53]:	nein	ja	nein	ja	ja
[32]:	nein	ja	ja	ja	ja
[21]:	nein	ja	nein	nein	ja
[25]:	nein	ja	nein	nein	nein

Tabelle 2.3.: Vergleich der Zugriffskontrollansätze im Hinblick auf deren Einsatzmöglichkeit und Flexibilität.

Teil II.

Kompositionale Politiken

Zu Beginn dieses Teils erfolgt eine kurze Einführung in die *Public Key* Infrastruktur SPKI/SDSI, die zur Realisierung kompositionaler Zugriffskontrolle verwendet wird. Der Einsatz einer solchen Infrastruktur ermöglicht es, die Vertrauensstrukturen der Teilnehmer des Zugriffkontrollsystems zu materialisieren. Es wird ein Beispiel einer virtuellen Organisation eingeführt, welches im Folgenden dieser Arbeit durchgängig verwendet wird.

Zwei Ansätze zur Zertifikat-basierten Realisierung kompositionaler Zugriffskontrolle werden vorgestellt. Der erste Ansatz erlaubt eine zentrale Zugriffserlaubnisentscheidung anhand kompositional spezifizierter Politiken. Der zweite Ansatz erlaubt eine dezentrale und sichere Ausführung einer kompositional spezifizierten Ausführungssequenz von Webdiensten.

Kapitel 3.

Vorbereitendes und ein Beispiel

In diesem Kapitel starten wir mit einer kurzen Einführung in die *Public Key* Infrastruktur SPKI/SDSI, welche wir für die Realisierung der Ansätze zur kompositionalen Zugriffskontrolle verwenden. Bedingt durch den Einsatz einer solchen *Public Key* Infrastruktur betrachten wir die Vertrauensstrukturen der Teilnehmer in unserer Umgebung und welche Wechselwirkungen zwischen verschiedenen Ausprägungen von Vertrauen und deren Materialisierung existieren. Abschließend führen wir eine Beispielszenario ein, welches innerhalb einer verteilten Web-Umgebung realisiert werden kann. Dieses Szenario veranschaulicht den Einsatz kompositionaler Zugriffskontrolle und wird in den folgenden Kapiteln dieser Arbeit aufgegriffen.

3.1. Einführung in *Public Key* Infrastrukturen

Credentials sind ein verbreiteter Implementierungsmechanismus zur Sicherung von Zugriffskontrolle, siehe beispielsweise [22, 26]. Ein signifikanter Vorteil für den Einsatz von *Credentials* in offenen Rechensystemen ist, dass diese auf Grund ihrer kryptographisch gesicherten Beglaubigungen über ungesicherte Kanäle versendet werden können. Im Rahmen einer attributbasierten Zugriffskontrolle können *Credentials* ebenfalls zur Kodierung von Teilnehmermerkmalen genutzt werden. Für einen typischen Zugriffsentscheidungsvorgang in einem Zertifikat-basierten Zugriffskontrollsystem sendet ein Teilnehmer eine Baumstruktur von *Credentials* als Zugriffsanfrage an eine Zugriffskontrollkomponente, siehe [27]. Für eine Zugriffserlaubnis muss die Struktur von mindestens einem Teilnehmer (*Zertifizierer*) ausgehen, den die Zugriffskontrollkomponente als vertrauenswürdig einschätzt. Eine prinzipielle Frage ist, was in diesem Kontext unter „Vertrauen“ verstanden wird. Üblicherweise tritt eine vertrauenswürdige dritte Partei (*certification authority*) auf, von welcher angenommen wird, dass sie stets korrekte Informationen zertifiziert und nicht zu betrügen beabsichtigt. Prinzipiell allerdings bleibt es den Teilnehmern eines Systems überlassen, welche anderen Teilnehmer als (allgemein) vertrauenswürdig angesehen werden. Die mindesten Anforderungen hierfür sind, dass

ein vertrauenswürdiger Teilnehmer seinen öffentlichen Schlüssel in authentischer Form vorlegt und den geheimen Schlüssel adäquat schützt.

Die Teilnehmer eines SPKI/SDSI Systems, siehe [34, 35, 66], werden durch ihre öffentlichen Schlüssel repräsentiert. Häufig wird nicht zwischen einem Teilnehmer s und seinem öffentlichen Schlüssel K_s unterschieden, sondern der Begriff *Prinzipal* verwendet. SPKI/SDSI-Zertifikate binden Prinzipale an Autorisierungen, Eigenschaften oder Namen. Die Bindung geschieht entweder explizit über einen im Zertifikat benannten Prinzipal oder implizit über einen Namen, der sich zu einem Prinzipal auflösen lässt. Durch seine Entstehung bedingt, besteht SPKI/SDSI aus zwei verschiedenen Teilen: Eine Namensdefinitionsstruktur (SDSI-Teil) und eine Autorisierungsstruktur (SPKI-Teil). Diese beiden Strukturen lassen sich sowohl gemeinsam als auch getrennt voneinander benutzen. Die Namensdefinition wird durch sogenannte *Namenszertifikate* realisiert, und die Autorisierungsstruktur durch sogenannte *Autorisierungszertifikate*. Ein wesentliches Merkmal der Namensstruktur ist die Tatsache, dass jeder Teilnehmer seinen eigenen, lokalen Namensraum verwaltet, indem er autonom Zertifikate ausstellt. Dies ist die für uns ausschlaggebende Eigenschaft von SPKI/SDSI, da andere verfügbare *Public Key* Infrastrukturen, wie beispielsweise das *KeyNote* System [18] oder X.509 v3 [40], dies nicht in dieser Form ermöglichen.

Definition 3.1 (Autorisierungszertifikat). *Ein SPKI Autorisierungszertifikat ist ein signiertes 5-Tupel der Form*

$$(Aussteller, Subjekt, Autorisierung, Delegation, Gültigkeit)_{AusstellerSignatur}.$$

Der *Aussteller* eines Autorisierungszertifikats ist der Prinzipal, der die Bindung des angegebenen *Subjekts* an die angegebene *Autorisierung* zertifiziert. *Delegation* gibt an, ob das begünstigte Subjekt die Autorisierung (oder Teile davon) weitergeben darf. Unter *Gültigkeit* wird die Gültigkeit des Zertifikats definiert. Im einfachsten Fall gibt die Gültigkeit eine Zeitspanne an; es gibt aber mehrere Varianten der Gültigkeitsbestimmung, von denen eine im Kapitel 4 näher besprochen wird.

Definition 3.2 (Namenszertifikat). *Ein SDSI Namenszertifikat ist ein signiertes 4-Tupel der Form*

$$(Aussteller, Bezeichner, Subjekt, Gültigkeit)_{AusstellerSignatur}.$$

Der *Aussteller* eines Namenszertifikats bildet zusammen mit dem von ihm frei wählbaren *Bezeichner* einen *lokalen Namen*. Dem lokalen Namen wird als Wert das angegebene *Subjekt* zugewiesen. Das Subjekt kann durch einen Prinzipal oder durch einen weiteren lokalen Namen angegeben sein. Der letztere Fall bedeutet, dass der Aussteller seinen

lokalen Namen an den Namensraum des Prinzipals bindet, der durch den weiteren lokalen Namen benannt wird. Für die *Gültigkeit* gelten dieselben Bedingungen wie bei einem Autorisierungszertifikat.

Im Hinblick auf eine Erlaubnisentscheidung müssen lokale Namen zu Prinzipalen aufgelöst werden können. Dieser Vorgang wird in der Literatur *Namensauflösung* genannt. Für die Definition der Semantik von lokalen Namen existieren verschiedene Ansätze in der Literatur, von denen die mengentheoretische Semantik die wohl am weitesten verbreitete ist, siehe hierzu [27]. Sei $\mathcal{C}$ eine gegebene Menge von Namenszertifikaten und sei $\mathcal{V}_{\mathcal{C}}(T)$ eine Evaluierungsfunktion, die auf einem Term T angewendet wird, wobei T ein Schlüssel K oder ein lokaler Name $K\ A$ oder ein erweiterter Name $K\ A_1 \ldots A_n$ ist. A und A_i sind beliebige, von einem Aussteller gewählte Bezeichner. *Subjekt* bezeichnet das Subjekt des Namenszertifikats, das den lokalen Namen $K\ A$ definiert. Nach [27] kann dann die Semantik von lokalen Namen als die kleinste Menge von Schlüsseln definiert werden, die die folgenden drei Eigenschaften erfüllt:

1. $\mathcal{V}_{\mathcal{C}}(K) = \{K\}$,
2. $\mathcal{V}_{\mathcal{C}}(K\ A) \supseteq \mathcal{V}_{\mathcal{C}}(\mathit{Subjekt})$,
3. $\mathcal{V}_{\mathcal{C}}(K\ A_1 A_2 \ldots A_n) = \bigcup_{K' \in \mathcal{V}_{\mathcal{C}}(K\ A_1)} \mathcal{V}_{\mathcal{C}}(K'\ A_2 \ldots A_n)$.

Die konkrete Nutzung der SPKI/SDSI Zertifikate zur Erlaubnisentscheidung kann in [27] nachgelesen werden. Ein anfragender Prinzipal muss zwei verschiedene Nachweise erbringen, um Zugriff auf eine geschützte Ressource zu erlangen: Einen Authentizitätsbeweis (*proof of authenticity*) und einen Autorisierungsbeweis (*proof of authorization*). Für den ersten Nachweis muss ein anfragender Prinzipal K_s zeigen, dass er tatsächlich Teilnehmer s ist; beziehungsweise über den zugehörigen privaten Schlüssel PK_s verfügt. Für den zweiten Nachweis muss ein anfragender Teilnehmer eine Menge von Zertifikaten vorweisen, die ihn zur gewünschten Nutzung der Ressource autorisieren. Die Kommunikation der Teilnehmer findet durch das Austauschen von signierten Zertifikatsequenzen statt. Die Erlaubnisentscheidung wird von einem verifizierenden Teilnehmer, typischerweise dem Ressourcenhalter, anhand einer Zertifikatsequenz durchgeführt. Die in der Sequenz enthaltenen Zertifikate werden miteinander so *reduziert*, dass ein resultierendes (temporäres), nicht signiertes Autorisierungszertifikat erzeugt wird. Dieser Reduktionsalgorithmus wird im Rahmen des sogenannten *certificate chain discovery* verwendet, siehe [27]. Autorisiert das resultierende Zertifikat den anfragenden Prinzipal bezüglich des gewünschten Ressourcenzugriffs, so wird dieser erlaubt. Für einen verifizierenden Prinzipal K_v, und einen anfragenden Prinzipal K_s, der Zugriff auf eine Ressource R erhalten möchte, kann das resultierende Zertifikat der beispielhaft gegebenen Form $(K_v, K_s, R, \mathit{nein}, \mathit{jetzt})$ sein.

3.2. Schutzdomänen unter dem Blickwinkel der Vertrauensstrukturen

Beide in den nachfolgenden Kapiteln 4 und 5 vorgestellten Realisierungsmöglichkeiten kompositionaler Zugriffskontrolle basieren auf dem Einsatz der *Public Key* Infrastruktur SPKI/SDSI. Mittels der Mechanismen der *Public Key* Infrastruktur wird es ermöglicht, Vertrauen der Teilnehmer untereinander auszudrücken. Dies begründet eine Grundlage für eine Sicherheitsanalyse. Die in Kapitel 1 entworfene Umgebung für kompositionale und zustandsbasierte Zugriffskontrolle wird in diesem Abschnitt dahingehend untersucht, wie die Vertrauenstrukturen der Teilnehmer untereinander in dieser Umgebung mittels der SPKI/SDSI abgebildet werden können.

Die Teilnehmer an dem Szenario „Mehrere Schutzdomänen – freie Mediatoren“, welches in Abschnitt 1.2.3 konzeptionell vorgestellt wurde, lassen sich wie folgt charakterisieren. Auf konzeptueller Ebene betrachten wir die Vertrauensstrukturen der jeweiligen Teilnehmer. Für eine Zertifikat-basierte Realisierung diskutieren wir mögliche Materialisierungsmechanismen der Vertrauensstrukturen, die SPKI/SDSI bietet.

Der Verwalter

Auf konzeptueller Ebene legt der Verwalter einer Schutzdomäne in seiner Zugriffskontrollpolitik $\mathcal{P}$ durch einen kompositionalen, algebraischen Ausdruck fest, welche Teilnehmer welche Funktionalitäten welcher Dienste in Anspruch nehmen dürfen.

Im Hinblick auf eine Realisierung wird durch $\mathcal{P}$ festgelegt, welche (freien oder gebundenen) Merkmale eines anfragenden Teilnehmers vorliegen müssen, damit diesem ein Zugriffswunsch gewährt wird. Im expliziten Fall agiert der Verwalter als *Aussteller* eines entsprechenden Autorisierungszertifikats. Im impliziten Fall wird festgehalten, welchen Mediatoren vertraut wird, die geforderten freien Eigenschaften durch das Ausstellen von Namenszertifikaten zu bestätigen. Diese fungieren dann ebenfalls in der Rolle eines Ausstellers.

Ein Mediator

Auf konzeptueller Ebene fungiert ein Mediator als Beauftragter eines Verwalters, indem er eingehende Zugriffswünsche anhand der vom Verwalter spezifizierten Zugriffskontrollpolitik $\mathcal{P}$ überprüft und den eigentlichen Zugriffswunsch an die entsprechende Schutzdomäne weiterleitet.

Im Hinblick auf die Realisierung kann ein Mediator zusätzlich die Aufgaben eines Ausstellers übernehmen. Er kann dann freie Merkmale anderer Teilnehmer zu bestätigen. Hierfür stellt er entsprechende Namenszertifikate aus. Durch das Ausstellen solcher Zertifikate spricht er das Vertrauen in die Begünstigten seiner ausgestellten Zertifikate aus. Dass einige ihrer Begünstigten mithilfe der Namenszertifikate bei anderen Verwaltern Zugriffsrechte erlangen, muss den Ausstellern nicht bekannt sein.

Der Dienstnutzer

Der Vertrauensaspekt im Hinblick auf anfragende Teilnehmer als Dienstnutzer kommt nur unter dem Blickwinkel der Realisierung zum tragen. Der anfragende Dienstnutzer drückt sein Vertrauen in den Verwalter einer Schutzdomäne durch die Vorlage der ihm ausgestellten Zertifikate aus.

Die Abbildung 3.1 visualisiert die allgemeinen Vertrauensbeziehung unter den agierenden Teilnehmern sowie die Materialisierung des jeweiligen Vertrauens. Die Vertrauensbeziehungen auf der konzeptuellen Ebene werden durch Pfeile K.1 und K.2 dargestellt. Ein Verwalter einer Schutzdomäne drückt durch eine entsprechende Spezifikation seiner kompositionalen Zugriffskontrollpolitik sein Vertrauen in die anderen Teilnehmer aus. Die Pfeile R.1 bis R.4 visualisieren das Vertrauen auf der Realisierungsebene wie folgt:

R.1: Ein Verwalter stellt einem (potenziellen) Dienstnutzer Autorisierungszertifikate aus.

R.2: Ein Mediator, welcher bereits auf konzeptioneller Ebene das Vertrauen eines Verwalters ausgesprochen bekommen hat (K.1), stellt einem (potenziellen) Dienstnutzer Namenszertifikate aus.

R.3: Ein Dienstnutzer legt ihm ausgestellte Zertifikate vor, um einen Zugriffswunsch zu äußern.

R.4: Ein Dienstnutzer legt ihm ausgestellte Zertifikate vor, um ein (weiteres) Namenszertifikat ausgestellt zu bekommen.

Aufgrund der Dynamik der Vertrauensstrukturen müssen zwangsläufig auch Überzeugung, Misstrauen oder Zweifel von Teilnehmern berücksichtigt werden. Diese „Varianten von Vertrauen“ können verschiedene Auswirkungen auf die Zugriffsentscheidung eines Verwalters haben. Gehen wir davon aus, dass ein Verwalter seine Zugriffskontrollpolitik kompositional spezifiziert, so kann er sein Vertrauen in andere Teilnehmer oder sein Misstrauen anderen Teilnehmern gegenüber durch eine entsprechende Verwendung von Kompositionsoperatoren ausdrücken. Beispielsweise kann

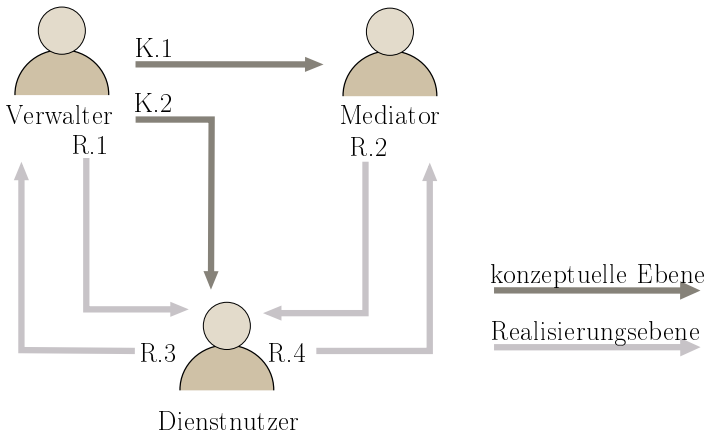


Abbildung 3.1.: Vertrauensstrukturen im Szenario kompositionaler Zugriffskontrolle.

die Verwendung des Differenzoperators zum Ausdruck des Misstrauens verwendet werden. In einer Zertifikat-basierten Operationalisierung werden die verschiedenen Varianten von Vertrauen mit entsprechenden Zertifikattypen materialisiert, wie sie in Abbildung 3.2 skizziert werden. Die Abbildung 3.2 veranschaulicht, dass ein Besitzer sein Vertrauen in einen Dienstnutzer durch das Ausstellen eines Autorisierungszertifikats ausdrückt. Stellt ein Mediator einem Dienstnutzer ein Namenszertifikat aus, so bindet er diesen Teilnehmer an einen lokalen Namen. Die spätere Verwendung dieses Namenszertifikat ist zum Zeitpunkt der Ausstellung nicht festgelegt. Der Mediator ist der Überzeugung, dass der Dienstnutzer ein Merkmal aufweist, das er durch das Namenszertifikat bestätigt. Da die Vertrauensstrukturen dynamisch sind, können Zweifel der ausstellenden Teilnehmer an anderen Teilnehmern auftreten. In diesen Fällen bietet die *Public Key* Infrastruktur die Möglichkeit eines Rückrufs von bereits ausgestellten Zertifikaten. Ein generelles Misstrauen eines Verwalters gegenüber eines potenziellen Dienstnutzers führt zum Ausstellen sogenannter *Verbotzertifikate*. Diese Aspekte sind nicht direkt für die vorliegende Arbeit relevant und werden deshalb hier nicht behandelt. Sie wurden detailliert in den Veröffentlichungen [3, 5, 6] und in der Arbeit [43] diskutiert.

3.3. Beispiel: Forschungsdatenbank ViBib

In diesem Abschnitt wird ein Beispiel für eine virtuelle Organisation vorgestellt, welche sich für ihre Zugriffskontrolle auf die Komposition der Zugriffskontrollpolitiken der einzelnen Organisationseinheiten stützt.

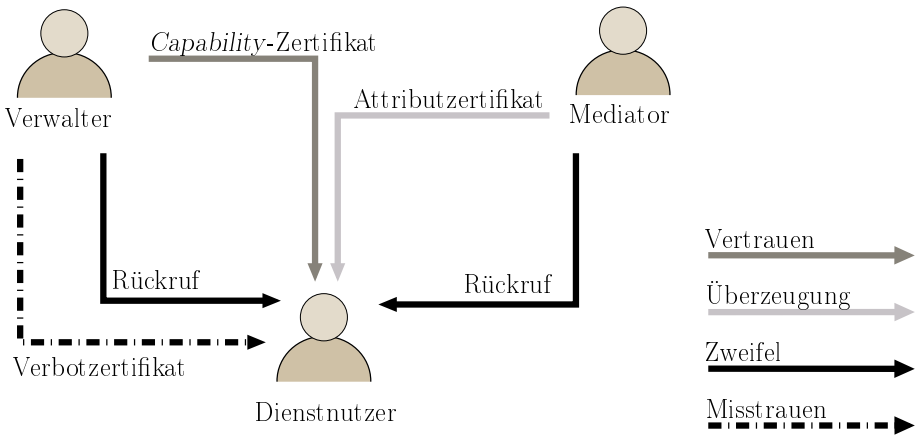


Abbildung 3.2.: Materialisierung der Vertrauensstrukturen im Szenario kompositionaler Zugriffskontrolle.

Wir betrachten eine virtuelle Forschungsdatenbank ViBib, die ihren Nutzern einen Zugang zu verteilt gespeicherten Dokumenten liefert. Die Organisationseinheiten, die zur virtuellen Forschungsdatenbank gehören, setzen sich aus verschiedenen Universitäts- und Landesbibliotheken zusammen. ViBib bietet seinen Nutzern die Möglichkeit einer Verfügbarkeitsrecherche nach Dokumenten der eingebundenen Organisationseinheiten und stellt hierfür einen einheitlichen Zugangspunkt. Die Nutzer erfahren, ob ein Dokument in einer der Bibliotheken vorrätig ist und ob ein elektronischer Volltext erhältlich ist. Da der Dienst der Forschungsdatenbank kostenlos gestellt wird, soll er nur zu Forschungszwecken in Anspruch genommen werden dürfen. Zur Vereinfachung des Szenarios gehen wir davon aus, dass die Hochschulbibliothek der TU Dortmund, TUDoBib, die Stadtbibliothek Dortmund, CityBib, und die Hochschulbibliothek der Ruhr-Universität Bochum, RUBib, eingebunden sind. Nutzer, welche die Dokumentensuche und den Dokumentenzugriff über ViBib verwenden, unterliegen der Zugriffskontrollpolitik der virtuellen Bibliothek (P_{ViBib}). In diesem Kontext verwenden wir Rollenkonzepte, so dass wir Erlaubnisse nicht für identifizierte Subjekte spezifizieren, sondern die Subjekte bestimmten Rollen zuordnen. Die konkrete Zuordnung eines Subjekts zu seiner Rolle wird in diesem Beispiel allerdings nicht thematisiert und wird in Abschnitt 8.2 besprochen. Die einzelnen Bibliotheken verfügen ebenfalls über elektronische Zugriffspunkte, die ausschließlich von den lokal spezifizierten Zugriffskontrollpolitiken kontrolliert und überwacht werden. Für die lokal angebotenen Dienste der einzelnen Bibliotheken wird von einem anfragenden Nutzer eine entsprechende Mitgliedschaft inklusive einer persönlich gewählten Kennnummer verlangt. Die lokalen Dienste sind in unserem Szenario auf die Suche und das Anzeigen von elektronischen Dokumenten beschränkt. Des Weiteren gehen

wir zunächst davon aus, dass alle hier betrachteten Dokumente einheitlich bezüglich der Zugriffskontrolle behandelt werden.

Unter Berücksichtigung der lokalen Zugriffskontrollpolitiken der eingebundenen Organisationseinheiten ($P_{TUDoBib}$, $P_{CityBib}$ und P_{RUBib}) wird die Politik P_{ViBib} von dem Verwalter der Forschungsdatenbank kompositional spezifiziert. Folgende Annahmen liegen der Spezifizierung zugrunde:

- Der Verwalter geht davon aus, dass Mitglieder der Hochschulbibliotheken die Dokumente ausschließlich für Forschungszwecke verwenden. Daher wird die Vereinigung der Komponentenpolitiken $P_{TUDoBib}$ und $P_{CityBib}$ gebildet: $P_{TUDoBib} + P_{RUBib}$.

- Die Mitglieder der Stadtbibliothek haben primär keinen Bezug zur Forschung, insofern sieht der Verwalter der Forschungsdatenbank vor, dass nur die Nutzer der Stadtbibliothek Zugriff auf die Dokumente erhalten sollen, die einen Forschungsbezug nachweisen können: $P_{CityBib}\ ^\wedge\ [s = \textit{Forschungsbezug}]$.

- Die Nutzung der Forschungsdatenbank soll nicht solchen Nutzern zur Verfügung stehen, die bereits offene Mahngebühren bei einer der involvierten Bibliothek haben. Zur Vereinfachung nehmen wir an, dass die jeweiligen Organisationseinheiten i zu diesen Zwecken eine „negative“ Politik $\overline{P}_i$ verwalten.

Die Zugriffskontrollpolitik der virtuellen Forschungsdatenbank wird nach dem Formalismus vorgestellt in Abschnitt 2.2.1 wie folgt kompositional spezifiziert:

$$P_{ViBib} = (P_{TUDoBib} - \overline{P}_{TUDoBib}) + (P_{RUBib} - \overline{P}_{RUBib}) + (P_{CityBib} - \overline{P}_{CityBib})^\wedge\ [s = \textit{Forschungsbezug}]$$

Die Abbildung 3.3 veranschaulicht die virtuelle Organisation mitsamt ihrer eingebundenen Organisationseinheiten und deren Schutzdomänen. Beachte, dass die eingebundenen Organisationseinheiten nicht notwendigerweise alle Objekte ihrer Schutzdomäne der virtuellen Organisation zur Verfügung stellen. Die Abbildung visualisiert dies beispielhaft durch die farbige Hervorhebung der eingebundenen Objekte. Des Weiteren können Nutzer die Objekte der virtuellen Organisation über verschiedene Zugriffspunkte zugreifen. Zur Verfügung stehen hier die Zugriffspunkte der einzelnen Schutzdomänen. Abhängig vom gewählten Zugriffspunkt kann über den Zugriffswunsch unterschiedlich entschieden werden.

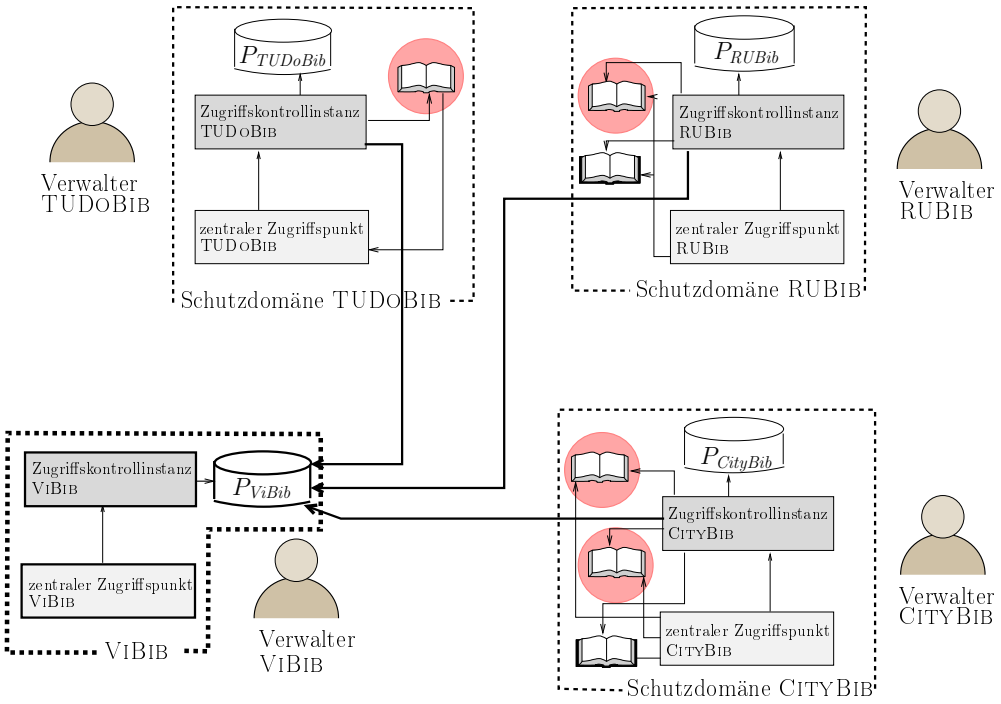


Abbildung 3.3.: Beispielszenario der Schutzdomänen der virtuellen Forschungsdatenbank.

Kapitel 4.

Realisierung kompositionaler Zugriffskontrolle

In Kapitel 2.2.1 wurde ein zustandsloser Ansatz kompositionaler Zugriffskontrolle vorgestellt und in Kapitel 3.3 wurde anhand dieses Ansatzes ein Beispielszenario entworfen. Die Originalarbeit [19] schlägt keine Realisierung der kompositional spezifizierten Zugriffskontrollpolitiken vor. In unseren Arbeiten [1, 4] stellen wir eine Realisierung mittels der SPKI/SDSI Infrastruktur vor. Die Elemente einer kompositionalen Zugriffskontrollpolitik, wie sie in [20] definiert wurde, sind Grundautorisierungsterme der Form $<s, o, a>$. Wie bereits in Abschnitt 2.2.1 beschrieben, kann ein solcher Grundautorierungsterm als subjektbezogene *capability* interpretiert werden. Diese Interpretation ermöglicht, dass ein Grundautorisierungsterm als Autorisierungszertifikat implementiert werden kann. Das Subjekt s des Grundautorisierungsterms tritt dann als Subjekt eines entsprechenden Autorisierungszertifikats auf, in dem die *capability* $<o, a>$ dem Subjekt durch eine entsprechende Autorisierung im Zertifikat zugesprochen wird. Das Zusprechen einer Autorisierung, beziehungsweise die Bindung eines Begünstigten an eine *capability*, geschieht durch das Ausstellen von entsprechenden Zertifikaten.

Unserer Realisierung liegt das grundlegende Konzept zugrunde, dass Subjekte eines Autorisierungszertifikats mittels algebraischer Ausdrücke spezifiziert werden, wie sie in Abschnitt 2.2.1 beziehungsweise Kapitel 3.3 vorgestellt wurden. Hierfür wurde in [4] eine entsprechende Erweiterung der Autorisierungszertifikate vorgeschlagen. Unsere Realisierung stützt sich auf zwei, aufeinander bezogene Zertifikatsklassen, wie sie für das *hybride Modell* in [16, 49] identifiziert wurden:

1. Sogenannte *Attributzertifikate* sprechen einem Teilnehmer Merkmale zu, deren spätere Verwendung noch nicht festgelegt ist. Diese Merkmale werden *freie Merkmale* genannt.

   Auf der Realisierungsebene werden die freien Merkmale durch das Ausstellen von Namenszertifikaten an Prinzipale gebunden. Somit werden die freien Merkmale

durch lokale Namen realisiert. Da jeder Teilnehmer seinen eigenen, lokalen Namensraum verwaltet, kann jeder Teilnehmer unter von ihm für sinnvoll befundene lokale Namen weitere Prinzipale oder Gruppen definieren. Dies führt allerdings dazu, dass von Teilnehmer zu Teilnehmer syntaktisch gleiche Namen eine unterschiedliche Semantik erhalten. Unter der Annahme, dass öffentliche Schlüssel systemweit eindeutig sind, können allerdings durch den Bezug auf den öffentlichen Schlüssel eines definierenden Teilnehmers lokale Namen global eindeutig benutzt werden.

2. Sogenannte *Capability-Zertifikate* sprechen einem Teilnehmer Zugriffsrechte für einen konkreten Dienst zu. Diese Berechtigungen werden *gebundene Merkmale* genannt, da deren Verwendung im Hinblick auf den zugehörigen Dienst festgelegt ist.

   Auf der Realisierungsebene werden gebundene Merkmale durch das Ausstellen von *Capability*-Zertifikaten an Prinzipale gebunden.

Betrachten wir das Szenario der Schutzdomänen, wie es in Abschnitt 1.2 eingeführt wurde, so treten Verwalter von Schutzdomänen als Spezifizierer von kompositionalen Zugriffskontrollpolitiken auf, siehe hierzu auch das Beispiel in Kapitel 3.3. In einer Zertifikat-basierten Realisierung treten diese Verwalter als vertrauenswürdige Zertifizierer auf. Um eine kompositional spezifizierte Zugriffskontrollpolitik zu realisieren, wurden zwei Möglichkeiten identifiziert. Ein Verwalter kann jedem Subjekt, welches durch eine Zugriffskontrollpolitik autorisiert ist, *explizit* ein entsprechendes *Capability*-Zertifikat ausstellen. Oder ein Verwalter kann die von der Zugriffskontrollpolitik autorisierten Subjekte *implizit* mit Hilfe lokaler Namen begünstigen. In diesem letzteren Fall werden weitere, vertrauenswürdige Zertifizierer benötigt, die Attributzertifikate für die entsprechenden lokalen Namen ausstellen. In unserem Szenario wird diese Aufgabe von den Mediatoren übernommen, die in Abschnitt 1.2.3 eingeführt worden sind. Ein Verwalter benennt vertrauenswürdige Mediatoren und erkennt von diesen ausgestellte Attributzertifikate an.

Zur Entscheidung über seinen Zugriffswunsch legt ein anfragender Teilnehmer eine Zertifikatsequenz bei dem Verwalter der Schutzdomäne vor, aus der sich die notwendige Berechtigung zur Inanspruchnahme der gewünschten Funktionalität des Dienstes ableiten lassen soll. Im expliziten Fall kann der anfragende Teilnehmer direkt das entsprechende *Capability*-Zertifikat vorlegen, welches ihn explizit als Subjekt autorisiert. Im impliziten Fall muss der Teilnehmer durch eine Menge ihn begünstigende Attributzertifikate nachweisen, dass er impliziter Begünstigter eines entsprechenden *Capability*-Zertifikats für diesen Dienst ist.

Definition 4.1 (algebraisches Subjekt). *Ein algebraisches Subjekt eines Capability-Zertifikats ist durch die folgende Grammatik gegeben:*

$$E \quad ::= \quad \boldsymbol{sbj} \mid E + E \mid E\&E \mid E - E \mid E^{\wedge} cons$$

Der Bezeichner ***sbj*** *steht für einen Prinzipal oder einen lokalen Namen. Der Bezeichner cons ist Element einer eingeschränkten Sprache $\mathcal{L}_{cons}$, die nur Einschränkungen auf Prinzipale erlaubt.*

Die Semantik eines algebraischen Subjekts wird durch eine Menge von Schlüsseln gegeben. Sie basiert auf der mengentheoretischen Semantik lokaler Namen, wie in Abschnitt 3.1 eingeführt, so dass für eine gegebene Menge von Namenszertifikaten $\mathcal{C}$ und ein algebraisches Subjekt ζ die Semantik gegeben ist durch $\mathcal{V}_{\mathcal{C}}(\zeta)$, sofern $\zeta = K$ oder $\zeta = K\ A$. Für zusammengesetzte algebraische Subjektausdrücke ζ_1 und ζ_2 gilt:

$$\begin{aligned} \mathcal{V}_{\mathcal{C}}(\zeta_1 + \zeta_2) &= \mathcal{V}_{\mathcal{C}}(\zeta_1) \cup \mathcal{V}_{\mathcal{C}}(\zeta_2), \\ \mathcal{V}_{\mathcal{C}}(\zeta_1 \,\&\, \zeta_2) &= \mathcal{V}_{\mathcal{C}}(\zeta_1) \cap \mathcal{V}_{\mathcal{C}}(\zeta_2), \\ \mathcal{V}_{\mathcal{C}}(\zeta_1 - \zeta_2) &= \mathcal{V}_{\mathcal{C}}(\zeta_1) \setminus \mathcal{V}_{\mathcal{C}}(\zeta_2). \end{aligned}$$

Da wir nur den eingeschränkten Fall der Selektion betrachten, kann die Auswertung eines entsprechenden algebraischen Subjekts auf die Auswertung des Durchschnitts reduziert werden.

Durch die Definition 4.1 eines algebraischen Subjekts, gemeinsam mit der mengentheoretischen Semantik lokaler Namen, lässt sich Folgendes festhalten.

Theorem 4.2. *Sei P eine kompositionale Politik aus Abschnitt 2.2.1, die, für eine feste capability $<o, a>$, durch die Kompositionsoperatoren Addition, Durchschnitt, Subtraktion, Selektion definiert ist. Seien K und K_s Prinzipale, sei $\mathcal{NC}_s$ eine Menge Attributzertifikate, die dem Prinzipal K_s vorliegen, sei $\zeta = P$ ein algebraisches Subjekt nach Definition 4.1, und sei $\mathcal{C}$ die Vereinigung der Attributzertifikate $\mathcal{NC}_s$ und dem von K ausgestelltem Capability-Zertifikat: $\mathcal{C} = \mathcal{NC}_s \cup \{(K, \zeta, <o, a>, Delegation, true)\}$. Dann gilt*

$$<s, o, a> \in [[P]]_e \text{ genau dann, wenn } K_s \in \mathcal{V}_{\mathcal{C}}(\zeta).$$

Im Hinblick auf Zugriffsentscheidungen für *Capability*-Zertifikate mit algebraischen Subjekten wird die durch einen anfragenden Prinzipal vorgelegte Zertifikatsequenz mit einem von uns erweiterten Reduktionsalgorithmus, siehe Algorithmus 1, ausgewertet. Die Kernidee der Auswertung basiert darauf, dass das algebraische Subjekt als Syntaxbaum dargestellt wird, dessen Knoten durch die verwendeten Operatoren dargestellt werden, und dessen Blätter durch die verwendeten Prinzipale oder lokale Namen dargestellt werden. Dieser Syntaxbaum wird dann rekursiv ausgewertet. Für jedes Vorkommen eines Operators werden zur Auswertung temporäre Zertifikate generiert, wie im Folgenden erklärt.

Addition:

Die Additionsoperation wird dadurch realisiert, dass für jeden Operanden ein lokaler Name generiert wird. Der Verwalter begünstigt diese lokalen Namen in einem *Capability*-Zertifikat. Von dem Verwalter benannte Mediatoren stellen Attributzertifikate für die lokalen Namen aus. Hierdurch werden begünstigte Gruppen von Subjekten gebildet, auf denen die mengentheoretische Vereinigung angewendet wird.

Durchschnitt:

Die Durchschnittsoperation wird ebenfalls dadurch realisiert, dass für jeden Operand ein lokaler Name generiert wird. Zudem wird auf ein spezielles Konzept der SPKI/SDSI Infrastruktur zurückgegriffen, dem *Threshold*-Subjekt der Form „k-of-n“. Im Fall eines *Threshold*-Subjekts wird die Autorisierung auf n nicht notwendigerweise verschiedene Subjekte aufgeteilt. *Threshold*-Subjekte lassen sich zu Prinzipalen auflösen, welche mindestens in der geforderten Anzahl k von den n Subjekten enthalten sind.

Die *Threshold*-Subjekte werden in diesem Kontext dazu verwendet, einen anfragenden Teilnehmer dazu zu zwingen, mehrere Gruppenzugehörigkeiten nachzuweisen, die ihn bezüglich der angefragten *capability* begünstigen. Der Verwalter stellt ein *Capability*-Zertifikat aus, das als *Subjekt* ein *Threshold*-Subjekt enthält, welches genau die generierten lokalen Namen benennt. Ein anfragender Teilnehmer muss dann nachweisen, dass er zu jeder durch die benannten lokalen Namen gebildeten Gruppe gehört, um die angefragte *capability* in Anspruch nehmen zu können.

Beispiel 4.3. Der Verwalter, repräsentiert durch K_v, stellt ein *Capability*-Zertifikat der Form $(K_v, \mathit{Subjekt}, <o, a>, \mathit{nein}, \mathit{true})_{PK_v}$ aus, für das *Subjekt* durch den Ausdruck $K_{m_1}\mathit{grantees}$ & $K_{m_2}\mathit{grantees}$ gegeben ist. In diesem Fall muss ein anfragender Teilnehmer nachweisen, dass er sowohl Begünstigter des durch K_{m_1} repräsentierten Mediator m_1 als auch Begünstigter des durch K_{m_2} repräsentierten Mediator m_2 ist, um die *capability* $<o, a>$ in Anspruch nehmen zu können. Das in der Realisierung verwendete SPKI/SDSI *Threshold*-Subjekt hat dann die Form 2-of-2 $(K_{m_1}\mathit{grantees}, K_{m_2}\mathit{grantees})$. ◊

Subtraktion:

Die Realisierung des Subtraktionsoperators war allein mit den vorhandenen Konzepten der SPKI/SDSI Infrastruktur nicht möglich. Es wurde eine abgewandelte Form des *Threshold*-Subjekts eingesetzt, mit dem ein lokaler Name für die „zu subtrahierende“ Politik benannt werden kann. Für ein algebraisches Subjekt der

Form $K_{m_1}grantees - K_{m_2}grantees$ hat ein solches, spezielles *Threshold*-Subjekt die Form 1st-of-2 $(K_{m_1}grantees, K_{m_2}grantees)$.

Andererseits wurde eine Erweiterung der Gültigkeitsangaben von *Capability*-Zertifikaten entworfen und umgesetzt, mit der eine Art *online*-Test (*location*) ermöglicht wird, der die Durchsetzung der Subtraktion erlaubt. Die *location* gibt die Gültigkeit in Form eines lokalen Namen an, der (nach Auswertung) die Prinzipale enthält, welche *keine* Zugriffserlaubnis erhalten sollen.

Beispiel 4.4. Der Verwalter, repräsentiert durch K_v, stellt ein *Capability*-Zertifikat der Form $(K_v, Subjekt, <o, a>, nein, K_{m_2}grantees)_{PK_v}$ aus, für das *Subjekt* durch den Ausdruck $K_{m_1}grantees - K_{m_2}grantees$ gegeben ist. In diesem Fall muss ein anfragender Teilnehmer nachweisen, dass er Begünstigter des durch K_{m_1} repräsentierten Mediator m_1 ist, aber *kein* Begünstigter des durch K_{m_2} repräsentierten Mediators m_2, um die *capability* $<o, a>$ in Anspruch nehmen zu können. ◊

Zur korrekten Durchsetzung der Subtraktion während einer Erlaubnisentscheidung wird davon ausgegangen, dass der Mediator m_2 dem verifizierenden Prinzipal K_v eine Aufzählung der Prinzipale liefert, die durch den lokalen Namen $K_{m_2}grantees$ benannt werden.

Selektion:

Wir betrachteten zunächst nur den eingeschränkten Fall einer Selektion auf Subjekten. Insofern konnte die Realisierung der Selektion auf die Realisierung des Durchschnitts übertragen werden.

Die Zertifikat-basierte Realisierung der Kompositionsoperatoren der zugrunde liegenden Algebra wird jeweils durch ein *Capability*-Zertifikat mit einem algebraischen Subjekt und zugehörige Attributzertifikate erreicht, welche durch vertrauenswürdige Mediatoren ausgestellt werden. Der Reduktionsalgorithmus der SPKI/SDSI Infrastruktur, siehe [27], der eine Zugriffsentscheidung aufgrund einer vorgelegten Zertifikatsequenz trifft, wurde entsprechend angepasst, um algebraische Subjekte auswerten zu können. Der von uns erweiterte Auswertalgorithmus wendet für ein algebraisches Subjekt den originären Algorithmus auf den Operanden des Ausdrucks (Prinzipale oder lokale Namen) an, und wertet die so entstehenden Mengen von begünstigten Subjektgruppen durch die entsprechenden mengentheoretischen Operationen aus. Hierfür wird zunächst das algebraische Subjekt als Syntaxbaum dargestellt. Die notwendigen Eingabeparameter des Algorithmus 1 „boolean *evaluate*“ sind:

- Der verifizierende Prinzipal K_v,

- der anfragende Prinzipal K_s,
- eine Menge von Attributzertifikaten $\mathcal{NC}_s$, die der anfragende Prinzipal mitliefert,
- das *Capability*-Zertifikat *acl* mit dem algebraischen Subjekt, und
- die Wurzel *node* des Syntaxbaums, der das algebraische Subjekt des *Capability*-Zertifikats darstellt.

Der Algorithmus ruft die Funktionen *change_left*(*acl*) und *change_right*(*acl*) auf, welche das Subjekt des übergebenen Zertifikats *acl* so überschreiben, dass *Subjekt* nur noch den linken beziehungsweise den rechten Teilbaum des ursprünglichen Syntaxbaums darstellt. Die Funktionen *replace_left*(K,*acl*) und *replace_right*(K,*acl*) ersetzen das Subjekt des übergebenen Zertifikats *acl* durch den übergebenen Schlüssel K. Die Variablen $left_{child}$ und $right_{child}$ sind der linke beziehungsweise rechte Kindknoten der Wurzel des Syntaxbaumes, der das aktuelle *Subjekt* des Zertifikats *acl* darstellt. Des Weiteren verwendet der Algorithmus 1 als lokale Variablen my_{acl} und *auth* vom Typ *Capability*-Zertifikat, $attr_1$ und $attr_2$ vom Typ Attributzertifikat, und $\mathcal{C}$ vom Typ Zertifikatmenge.

Zur korrekten Ausführung des Algorithmus 1 liegen die folgenden Annahmen zugrunde.

Annahme 4.5. *Jeder Aussteller und jeder Begünstigte eines Zertifikats verfügt über eine lokale Kopie des Zertifikats.*

Annahme 4.6. *Sofern das algebraische Subjekt eines capability-Zertifikats den Subtraktionsoperator verwendet, ist dieser Ausdruck in Normalform. Das bedeutet, dass die Subtraktion ausschließlich auf solchen Operanden angewendet wird, die Prinzipale oder lokale Namen sind.*

Die Annahme 4.6 bedeutet für einen ausstellenden Teilnehmer keine Einschränkung in der Spezifizierung des kompositionalen Ausdrucks, da jeder kompositionale Ausdruck, der aus Prinzipalen und lokalen Namen und den Operatoren Addition, Durchschnitt und Subtraktion spezifiziert ist, in Normalform umgewandelt werden kann.

Lemma 4.7. *Jeder kompositional spezifizierte Ausdruck aus Prinzipalen und lokalen Namen und den Operatoren Addition, Durchschnitt und Subtraktion kann in Normalform umgewandelt werden.*

Beweis. Seien P_1, P_2 und P_3 entsprechende Ausdrücke, dann können die folgenden Umwandlungen angewendet werden.

$$
\begin{array}{lcl}
P_1 - (P_2 \,\&\, P_3) & = & (P_1 - P_2) + (P_1 - P_3) \\
(P_1 \,\&\, P_2) - P_3 & = & (P_1 - P_3) \,\&\, (P_2 - P_3) \\
P_1 - (P_2 + P_3) & = & (P_1 - P_2) \,\&\, (P_1 - P_3) \\
(P_1 + P_2) - P_3 & = & (P_1 - P_3) + (P_2 - P_3) \\
(P_1 - P_2) - P_3 & = & (P_1 - P_2) \,\&\, (P_1 - P_3) \\
P_1 - (P_2 - P_3) & = & (P_1 - P_2) + (P_1 \,\&\, P_3)
\end{array}
$$

□

In unserem erweiterten Reduktionsalgorithmus, Algorithmus 1, werden für die Behandlung der drei Operatoren *Addition*, *Durchschnitt* und *Subtraktion* die drei Auswertalgorithmen *algorithm*(K_s,$\mathcal{C}$), *threshold_ algorithm*(K_s,$\mathcal{C}$) und *1st-of-2_ algorithm*(K_s,$\mathcal{C}$) aufgerufen, die im Folgenden beschrieben werden.

boolean *algorithm*(K_s,$\mathcal{C}$)

1. Von der übergebenen Zertifikatmenge $\mathcal{C}$ werden zunächst alle Zertifikate entfernt, welche nicht gültig sind, und welche als Autorisierung nicht die von K_s gewünschte *capability* $<o, a>$ enthalten.

2. Der in [27] angegebene Algorithmus zur Namensauflösung wird auf der Zertifikatmenge $\mathcal{C}$ ausgeführt, und $\mathcal{C}'$ sei die resultierende Zertifikatmenge.

3. Von der resultierenden Zertifikatmenge $\mathcal{C}'$ werden die *capability*-Zertifikate entfernt, deren Subjekte *nicht* durch einen einzelnen Schlüssel K_i gegeben sind.

4. Es wird ein Graph G erzeugt, dessen Knoten durch die in $\mathcal{C}'$ enthaltenen Schlüssel K_i gegeben sind. Für jedes *capability*-Zertifikat $(K_i, K_j, <o, a>, \mathit{Delegation}, \mathit{true}) \in \mathcal{C}'$ wird in G eine entsprechende Kante von K_i zu K_j erzeugt.

Der Aufruf dieses Algorithmus liefert *true*, genau dann, wenn in G ein Pfad existiert, dessen Anfangsknoten K_v und dessen Endknoten K_s ist.

boolean *threshold_ algorithm*(K_s,$\mathcal{C}$)

1. Das in $\mathcal{C}$ enthaltene *capability*-Zertifikat *auth* hat ein *Threshold*-Subjekt der Form *Subjekt* = 2-of-2(K_i*grantees*, K_j*grantees*). Die in Subjekt angegebenen Prinzipale K_i und K_j werden durch temporär erzeugte Schlüssel $K_{t'}$ und $K_{t''}$ ersetzt, so dass *Subjekt* = 2-of-2($K_{t'}$*grantees*, $K_{t''}$*grantees*).

2. Um die ursprüngliche Semantik des *capability*-Zertifikats zu wahren, werden

Algorithmus 1 boolean *evaluate*(K_v,K_s,$\mathcal{NC}_s$,*acl*,*node*)

my_{acl},*auth* *Capability*-Zertifikate; $attr_1$,$attr_2$ Attributzertifikate; $\mathcal{C}$ Zertifikatmenge;

if $left_{child}$ ist eine Operation **then**
  my_{acl} = *change_left*(*acl*)
  if *evaluate*(K_v,K_s,$\mathcal{NC}_s$,my_{acl},$left_{child}$) **then**
    replace_left(K_s,*acl*)
  else
    replace_left(K_v,*acl*)
  end if
end if

if $right_{child}$ ist eine Operation **then**
  my_{acl} = *change_right*(*acl*)
  if *evaluate*(K_v,K_s,$\mathcal{NC}_s$,my_{acl},$right_{child}$) **then**
    replace_right(K_s,*acl*)
  else
    replace_right(K_v,*acl*)
  end if
end if

switch(*node*)
case *node* ist ein öffentlicher Schlüssel K:
    return(K== K_s);
case *node* ist ein lokaler Name K N:
    Setze $\mathcal{C}$ = $\mathcal{NC}_s$;
    return true genau dann, wenn $K_s \in \mathcal{V}_C(K\ N)$;
{In den folgenden Fallunterscheidungen ist *Subjekt* von *acl* ein kompositionaler Ausdruck der Form $\mathcal{P}_1$ op $\mathcal{P}_2$, für den gilt $\mathcal{P}_1 = K_i grantees$ und $\mathcal{P}_2 = K_j grantees$.}
case *node* ist der Operator *Addition* {*Subjekt* = $\mathcal{P}_1$ + $\mathcal{P}_2$}:
    Setze *auth* = *acl*, wobei *Subjekt* = K_v *supergroup*;
    Setze $attr_1 = (K_v, supergroup, K_i\ grantees, true)$;
    Setze $attr_2 = (K_v, supergroup, K_j\ grantees, true)$;
    Setze $\mathcal{C}$ = $\mathcal{NC}_s \cup \{auth, attr_1, attr_2\}$;
    return(***algorithm***(K_s,$\mathcal{C}$));
case *node* ist der Operator *Durchschnitt* {*Subjekt* = $\mathcal{P}_1$ & $\mathcal{P}_2$}:
    Setze *auth* = *acl*, wobei *Subjekt* = 2-of-2($K_i grantees$, $K_j grantees$);
    Setze $\mathcal{C}$ = $\mathcal{NC}_s \cup \{auth\}$;
    return(***threshold_algorithm***($K_s, \mathcal{C}$));
case *node* ist der Operator *Subtraktion* {*Subjekt* = $\mathcal{P}_1$ − $\mathcal{P}_2$}:
    Setze *auth* = *acl*, wobei *Subjekt* = 1st-of-2($K_i grantees$, $K_j grantees$) und
    $Gültigkeit = K_j grantees$;
    Setze $\mathcal{C}$ = $\mathcal{NC}_s \cup \{auth\}$;
    return(***1st-of-2_algorithm***($K_s, \mathcal{C}$));

der Zertifikatmenge $\mathcal{C}$ zwei temporäre, nicht signierte *capability*-Zertifikate $(K_{t'}, K_i, <o, a>, Delegation, true)$ und $(K_{t''}, K_j, <o, a>, Delegation, true)$ hinzugefügt.

3. Zur Ausführung der Namensauflösung auf $\mathcal{C}$ wird das modifizierte *capability*-Zertifikat *auth* entfernt; ebenso alle Zertifikate, welche nicht gültig sind, und welche als Autorisierung nicht die von K_s gewünschte *capability* $<o, a>$ enthalten.

4. Der in [27] angegebene Algorithmus zur Namensauflösung wird auf der Zertifikatmenge $\mathcal{C}$ ausgeführt, und $\mathcal{C}'$ sei die resultierende Zertifikatmenge.

5. Von der resultierenden Zertifikatmenge $\mathcal{C}'$ werden die *capability*-Zertifikate entfernt, deren *Subjekte nicht* durch einen einzelnen Schlüssel K_i gegeben sind.

6. Das modifizierte *capability*-Zertifikat *auth* wird der resultierenden Zertifikatmenge $\mathcal{C}'$ hinzugefügt.

7. Es wird ein Graph G erzeugt, dessen Knoten durch die in $\mathcal{C}'$ enthaltenen Schlüssel K_i gegeben sind. Für jedes *capability*-Zertifikat $(K_i, K_j, <o, a>, Delegation, true) \in \mathcal{C}'$ wird in G eine entsprechende Kante von K_i zu K_j erzeugt.

Der Aufruf dieses Algorithmus liefert *true*, genau dann, wenn in G zwei Pfade p' und p'' mit den folgenden Eigenschaften existieren. Im Pfad p' ist K_v der Anfangsknoten, $K_{t'}$ ist ein innerer Knoten, und K_s ist der Endknoten. Im Pfad p'' ist K_v der Anfangsknoten, $K_{t''}$ ist ein innerer Knoten, und K_s ist der Endknoten.

boolean *1st-of-2_ algorithm*(K_s,$\mathcal{C}$)

1. Die in dem *capability*-Zertifikat angegebene Gültigkeits-*location* K_j*grantees* muss ausgewertet werden. Wir gehen davon aus, dass der verifizierende Prinzipal K_v von dem für ihn vertrauenswürdigen Mediator, repräsentiert durch K_j, eine Kopie aller Attributzertifikate erhält, welche den lokalen Namen K_j*grantees* definieren. Diese Attributzertifikate werden der Zertifikatmenge $\mathcal{C}$ hinzugefügt.

2. Das in $\mathcal{C}$ enthaltene *capability*-Zertifikat *auth* hat ein *Threshold*-Subjekt der Form *Subjekt* = 1st-of-2(K_i*grantees*, K_j*grantees*). Die in *Subjekt* angegebenen Prinzipale K_i und K_j werden durch temporär erzeugte Schlüssel $K_{t'}$ und $K_{t''}$ ersetzt, so dass *Subjekt* = 1st-of-2($K_{t'}$*grantees*, $K_{t''}$*grantees*).

3. Um die ursprüngliche Semantik des *capability*-Zertifikats zu wahren, werden der Zertifikatmenge $\mathcal{C}$ zwei temporäre, nicht signierte *capability*-Zertifikate $(K_{t'}, K_i, <o, a>, Delegation, true)$ und $(K_{t''}, K_j, <o, a>, Delegation, true)$ hinzuge-

fügt.

4. Zur Ausführung der Namensauflösung auf $\mathcal{C}$ wird das modifizierte *capability*-Zertifikat *auth* entfernt; ebenso alle Zertifikate, welche nicht gültig sind, und welche als Autorisierung nicht die von K_s gewünschte *capability* $<o, a>$ enthalten.

5. Der in [27] angegebene Algorithmus zur Namensauflösung wird auf der Zertifikatmenge $\mathcal{C}$ ausgeführt, und $\mathcal{C}'$ sei die resultierende Zertifikatmenge.

6. Von der resultierenden Zertifikatmenge $\mathcal{C}'$ werden die *capability*-Zertifikate entfernt, deren Subjekte *nicht* durch einen einzelnen Schlüssel K_i gegeben sind.

7. Das modifizierte *capability*-Zertifikat *auth* wird der resultierenden Zertifikatmenge $\mathcal{C}'$ hinzugefügt.

8. Es wird ein Graph G erzeugt, dessen Knoten durch die in $\mathcal{C}'$ enthaltenen Schlüssel K_i gegeben sind. Für jedes *capability*-Zertifikat $(K_i, K_j, <o, a>, \mathit{Delegation}, \mathit{true}) \in \mathcal{C}'$ wird in G eine entsprechende Kante von K_i zu K_j erzeugt.

Der Aufruf dieses Algorithmus liefert *true*, genau dann, wenn in G ein Pfad p' existiert, in dem K_v Anfangsknoten, $K_{t'}$ ein innerer Knoten und K_s Endknoten ist, *und* wenn *kein* Pfad p'' existiert, für den K_v der Anfangsknoten, $K_{t''}$ ein innerer Knoten, und K_s der Endknoten ist.

Die Korrektheit des Algorithmus 1 wird durch folgendes Theorem gezeigt.

Theorem 4.9. *Sei P eine kompositionale Politik aus Abschnitt 2.2.1, die, für eine feste capability $<o, a>$, durch die Kompositionsoperatoren Addition, Durchschnitt, Subtraktion, Selektion definiert ist. Sei K_v ein verifizierender Prinzipal, sei K_s ein anfragender Prinzipal, sei $\zeta = P$ ein algebraisches Subjekt, sei node die Wurzel des Syntaxbaums, der ζ darstellt, und sei $acl = (K_v, \zeta, <o, a>, \mathit{Delegation}, \mathit{true})$. Der erweiterte Auswertalgorithmus boolean evaluate$(K_s, \mathcal{NC}_s, acl, node)$ liefert true genau dann, wenn $<s, o, a> \in [[P]]_e$.*

Beweis. Aufgrund Theorem 4.2 ist $<s, o, a> \in [[P]]_e$ genau dann, wenn $K_s \in \mathcal{V}_{\mathcal{C}}(\zeta)$ gilt. Daher zeigen wir, dass boolean *evaluate*$(K_v, K_s, \mathcal{NC}_s, acl, node)$ *true* liefert, genau dann, wenn $K_s \in \mathcal{V}_{\mathcal{C}}(\zeta)$ gilt. Der Beweis geht per Induktion über die Tiefe n des Syntaxbaumes, der ζ repräsentiert.

1. Sei $n = 0$.

   „$\Rightarrow$“ (boolean *evaluate*$(K_v, K_s, \mathcal{NC}_s, acl, node)$ liefert *true*):
   In diesem Fall hat das übergebene *Capability*-Zertifikat *auth* die Form $(K_v, \zeta, <o, a>, \mathit{Delegation}, \mathit{true})$, wobei $\zeta = K_s$, oder $\zeta = K\ A$. Im ersten Fall ist

$\mathcal{V}_{\mathcal{C}}(\zeta) = \{K_s\}$. Im zweiten Fall existiert ein Attributzertifikat $(K, A, K_s, \mathit{true}) \in \mathcal{C}$, so dass $\mathcal{V}_{\mathcal{C}}(\zeta) = \{K_s\}$.

„$\Leftarrow$“ $K_s \in \mathcal{V}_{\mathcal{C}}(\zeta)$:
Entweder ist $\mathcal{V}_{\mathcal{C}}(\zeta) = \{K_s\}$, oder es existiert ein Attributzertifikat $(K, A, K_s, \mathit{true}) \in \mathcal{C}$, so dass ebenfalls $\mathcal{V}_{\mathcal{C}}(\zeta) = \{K_s\}$. Im ersten Fall hat das übergebene *Capability*-Zertifikat *auth* die Form $(K_v, K_s, <o, a>, \mathit{Delegation}, \mathit{true})$. Im zweiten Fall hat das übergebene *Capability*-Zertifikat *auth* die Form $(K_v, K\ A, <o, a>, \mathit{Delegation}, \mathit{true})$. In beiden Fällen liefert der Algorithmus boolean $\mathit{evaluate}(K_v, K_s, \mathcal{NC}_s, \mathit{acl}, \mathit{node})$ *true*.

2. Induktionsanfang. Sei $n = 1$.

   Sei *node* die Wurzel, B_1 und B_2 das linke beziehungsweise rechte Blatt. B_1 und B_2 sind durch Prinzipale oder lokale Namen gegeben. Im Folgenden sei B_1 gegeben durch $K\ A$, und sei B_2 gegeben durch $K'\ A'$. Die Argumentation für $B_1 = K$ und $B_2 = K'$ verläuft analog. Die Wurzel *node* ist gegeben durch eine der Operatoren Addition, Durchschnitt und Differenz.

   a) Sei $\mathit{node} = +$:
   In diesem Fall wählt der Algorithmus boolean *evaluate* die dritte **case**-Anweisung der **switch**-Umgebung aus. Es werden die folgenden drei Zertifikate erzeugt. Das *capability*-Zertifikat *auth* der Form $(K_v, K_v\mathit{supergroup}, <o, a>, \mathit{Delegation}, \mathit{true})$. Die Attributzertifikate attr_1 und attr_2 der Form $(K_v, \mathit{supergroup}, KA, \mathit{true})$ beziehungsweise $(K_v, \mathit{supergroup}, K'A', \mathit{true})$. Die Zertifikatmenge $\mathcal{C}$ wird definiert durch $\mathcal{C} = \mathcal{NC}_s \cup \{\mathit{auth}, \mathit{attr}_1, \mathit{attr}_2\}$.
   „$\Leftrightarrow$“:
   Es gilt $K_s \in \mathcal{V}_{\mathcal{C}}(K\ A) \cup \mathcal{V}_{\mathcal{C}}(K'\ A')$. Ist $K_s \in \mathcal{V}_{\mathcal{C}}(K\ A)$, dann existiert ein Attributzertifikat $(K, A, K_s, \mathit{true}) \in \mathcal{NC}_s$, und somit gilt auch $(K, A, K_s, \mathit{true}) \in \mathcal{C}$. Ist $K_s \in \mathcal{V}_{\mathcal{C}}(K'\ A')$, dann existiert ein Attributzertifikat $(K', A', K_s, \mathit{true}) \in \mathcal{NC}_s$, und somit gilt auch $(K', A', K_s, \mathit{true}) \in \mathcal{C}$.

   Die **case**-Anweisung ruft den Auswertalgorithmus $\boldsymbol{\mathit{algorithm}}(K_s, \mathcal{C})$ auf, der in Schritt 2. den originären Algorithmus zur Namensauflösung auf der Zertifikatmenge $\mathcal{C}$ anwendet. Dies bedeutet, dass die resultierende Zertifikatmenge $\mathcal{C}'$ mindestens ein *Capability*-Zertifikat der Form $(K_v, K_s, <o, a>, \mathit{Delegation}, \mathit{true})$ enthält. Der in Schritt 4. erzeugte Graph G enthält also den gesuchten Pfad.

   b) Sei $\mathit{node} = \&$:
   In diesem Fall wählt der Algorithmus boolean *evaluate* die vierte **case**-Anweisung der **switch**-Umgebung aus. Es wird das *Capability*-Zertifikat

auth der Form $(K_v, \text{2-of-2}(KA, K'A'), <o, a>, \mathit{Delegation}, \mathit{true})$ erzeugt. Die Zertifikatmenge $\mathcal{C}$ wird definiert durch $\mathcal{C} = \mathcal{NC}_s \cup \{auth\}$.
„⇔“:
Es gilt $K_s \in \mathcal{V}_\mathcal{C}(K\ A) \cap \mathcal{V}_\mathcal{C}(K'\ A')$. Ist $K_s \in \mathcal{V}_\mathcal{C}(K\ A)$, dann existiert ein Attributzertifikat $(K, A, K_s, \mathit{true}) \in \mathcal{NC}_s$, und somit gilt auch $(K, A, K_s, \mathit{true}) \in \mathcal{C}$. Ist $K_s \in \mathcal{V}_\mathcal{C}(K'\ A')$, dann existiert ein Attributzertifikat $(K', A', K_s, \mathit{true}) \in \mathcal{NC}_s$, und somit gilt auch $(K', A', K_s, \mathit{true}) \in \mathcal{C}$.

Die **case**-Anweisung ruft den Algorithmus ***threshold_algorithm***$(K_s, \mathcal{C})$ auf, welcher in Schritt 1. das *Capability*-Zertifikat *auth* zu $(K_v, \text{2-of-2}(K_{t'}A, K_{t''}A'), <o, a>, \mathit{Delegation}, \mathit{true})$ überschreibt. In Schritt 2. werden die beiden temporären *Capability*-Zertifikate $(K_{t'}, K, <o, a>, \mathit{Delegation}, \mathit{true})$ und $(K_{t''}, K', <o, a>, \mathit{Delegation}, \mathit{true})$ erzeugt und der Zertifikatmenge $\mathcal{C}$ hinzugefügt. Nachdem in Schritt 4. der originäre Algorithmus zur Namensauflösung auf der Zertifikatmenge $\mathcal{C}$ angewendet wurde, sind beide temporären *Capability*-Zertifikate in der resultierenden Zertifikatmenge $\mathcal{C}'$. Dies bedeutet, dass sowohl Pfad p' als auch Pfad p'' in dem in Schritt 7. erzeugten Graph G gefunden wird.

c) Sei $node = -$:
In diesem Fall wählt der Algorithmus boolean *evaluate* die fünfte **case**-Anweisung der **switch**-Umgebung aus. Es wird das *Capability*-Zertifikat *auth* der Form $(K_v, \text{1st-of-2}(KA, K'A'), <o, a>, \mathit{Delegation}, \mathit{true})$ erzeugt. Die Zertifikatmenge $\mathcal{C}$ wird definiert durch $\mathcal{C} = \mathcal{NC}_s \cup \{auth\}$.
„⇔“:
Es gilt $K_s \in \mathcal{V}_\mathcal{C}(K\ A) \setminus \mathcal{V}_\mathcal{C}(K'\ A')$ genau dann, wenn $K_s \in \mathcal{V}_\mathcal{C}(K\ A)$ und $K_s \notin \mathcal{V}_\mathcal{C}(K'\ A')$. Ist $K_s \in \mathcal{V}_\mathcal{C}(K\ A)$, dann existiert ein Attributzertifikat $(K, A, K_s, \mathit{true}) \in \mathcal{NC}_s$, und somit gilt auch $(K, A, K_s, \mathit{true}) \in \mathcal{C}$. Da $K_s \notin \mathcal{V}_\mathcal{C}(K\ A)$, existiert *kein* Attributzertifikat $(K, A, K_s, \mathit{true})$.

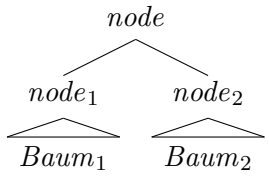
Die **case**-Anweisung ruft den Algorithmus ***1st-of-2_algorithm***$(K_s, \mathcal{C})$ auf, welcher in Schritt 2. das *Capability*-Zertifikat *auth* zu $(K_v, \text{1st-of-2}(K_{t'}A, K_{t''}A'), <o, a>, \mathit{Delegation}, \mathit{true})$ überschreibt. In Schritt 3. werden die beiden temporären *Capability*-Zertifikate $(K_{t'}, K, <o, a>, \mathit{Delegation}, \mathit{true})$ und $(K_{t''}, K', <o, a>, \mathit{Delegation}, \mathit{true})$ erzeugt und der Zertifikatmenge $\mathcal{C}$ hinzugefügt. Nachdem in Schritt 5. der originäre Algorithmus zur Namensauflösung auf der Zertifikatmenge $\mathcal{C}$ angewendet wurde, ist nur das temporäre *Capability*-Zertifikat $(K_{t'}, K, <o, a>, \mathit{Delegation}, \mathit{true})$ in der resultierenden Zertifikatmenge $\mathcal{C}'$. Dies bedeutet, dass nur Pfad p' in dem in Schritt 8. erzeugten Graph G gefunden wird.

3. Induktionsannahme. Für alle Syntaxbäume der Tiefe $1 \leq n \leq k$ gilt das

Theorem 4.9.

4. Induktionsschritt. Wir betrachten einen Syntaxbaum der Tiefe $n = k + 1$.

   Der Syntaxbaum ist der Form:



   Der Teilbaum $Baum_1$ der Tiefe $n \leq k$ ist laut Induktionsannahme korrekt ausgewertet worden. Dies bedeutet, dass $node_1$ durch einen Schlüssel ersetzt wird. Im Fall einer für den Anfragenden erfolgreichen Auswertung durch K_s, oder im Fall einer für den Anfragenden misslungenen Auswertung durch K_v. Die gleiche Argumentation gilt für den Teilbaum $Baum_2$. Somit erhalten wir den noch auszuwertenden Syntaxbaum:

   Dieser Syntaxbaum hat die Tiefe 1 und kann, wie im Induktionsanfang beschrieben ausgewertet werden. □

Kapitel 5.

Realisierung explizit verteilter Zugriffskontrolle

Der in der Veröffentlichung [2] vorgestellte Ansatz wurde ursprünglich entwickelt, um die Ausführung kompositionaler Webdienste abzusichern. Es wurden kryptographische Mechanismen eingesetzt, um den *sicheren Ablauf* einer zuvor in der Beschreibungssprache BPEL spezifizierten Ausführungsreihenfolge zu garantieren. Die Anforderungen an einen sicheren Ablauf, welche durch die eingesetzten Mechanismen erfüllt werden, sind:

Anforderung 5.1. *Die kompositional spezifizierte Ausführungsreihenfolge der Webdienste wird korrekt eingehalten.*

Anforderung 5.2. *Eingebundene Webdienste führen nur die in der Spezifikation angegebene Funktionalität aus.*

Anforderung 5.3. *Eingebundene Webdienste dürfen nur über solche Informationen verfügen, die zur Ausführung der in der Spezifikation angegebenen Funktionalität notwendig sind.*

Zur Durchsetzung der Spezifikation wird weitestgehend auf eine zentrale Kontrollinstanz verzichtet. Auszuführende Sicherheitsüberprüfungen werden an die eingebundenen Webdienste übertragen. Die zentrale Idee, um die Ausführung dezentral zu kontrollieren, besteht in der Konstruktion einer neuen Datenstruktur, die als *Container* bezeichnet wird. Die vorliegende Spezifikation wird instantiiert, indem anhand dieser Spezifikation ein entsprechender Container erzeugt wird, der die Informationen über den kompositionalen Ablauf trägt. Während der Ausführung einer initiierten Spezifikation wird der Container zwischen den eingebundenen Webdiensten verschickt, damit diese auf die notwendigen Informationen wie vorgesehen zugreifen und lokale Sicherheitsüberprüfungen durchführen können. Grob umrissen, muss ein eingebundener Webdienst hierfür die folgenden vier Schritte durchführen:

1. Der Webdienst extrahiert aus dem Prozess-Schema seinen Prozessausschnitt und erforderliche Eingabeparameter.

2. Der Webdienst ruft die in seinem Prozess-Schema benannte, gewünschte Funktionalität auf.

3. Der Webdienst aktualisiert den Container[1] und

4. er versendet den Container an die unmittelbar nachfolgenden Webdienste.

Der Container besteht aus drei Schichten, die nachfolgend vorgestellt werden. Für die Beschreibung der Containererzeugung wird der Bezeichner „Teilnehmer“ synonym mit „eingebundener Webdienst“ verwendet. Dem System liegt die *Public Key* Infrastruktur SPKI/SDSI zugrunde, so dass die Teilnehmer über Schlüsselpaare verfügen, symmetrische und asymmetrische Verschlüsselung verwendet werden kann, und Attribut- und *Capability*-Zertifikate im Umlauf sind. Diese Zertifikate können als notwendige Eingabe für auszuführende Funktionalitäten verwendet werden.

5.1. Die Zertifikat-Schicht

Die Zertifikat-Schicht ist verantwortlich für die Verwaltung der Teilnehmerzertifikate. Bevor die Spezifikation initiiert und gestartet wird, müssen dem initiierenden Teilnehmer alle Zertifikate vorliegen, die zur Durchführung benötigt werden. Ein initiierender Teilnehmer erzeugt für jedes Zertifikat i einen geheimen, symmetrischen Zertifikatschlüssel syk_i und verschlüsselt damit die Zertifikate. Ein so symmetrisch verschlüsselten Zertifikat $Enc_{syk_i}(i)$ wird dann unter einem eindeutigen Bezeichner l_i in der Zertifikat-Schicht des Containers gespeichert, damit sie später von den Teilnehmern leicht identifiziert werden können. Um zu verhindern, dass die Zertifikat-Schicht nachträglich von unbefugten Dritten unbemerkt manipuliert wird, signiert sie der initiierende Teilnehmer.

5.2. Die Ausgabe-Schicht

In der Ausgabe-Schicht werden (eventuelle) Rückgaben der eingebundenen Webdienste gespeichert. Auf diese Art können Rückgaben anderen Teilnehmern zur Verfügung gestellt werden. Ähnlich wie in der Zertifikat-Schicht dürfen auch hier nur die Teilnehmer auf die Rückgabewerte zugreifen, welche als Eingabewerte zur Ausführung der jeweilig gewünschten Funktionalität unbedingt notwendig sind. Aus diesem Grund werden die Rückgabewerte unter Anwendung spezieller, symmetrischer Ausgabe-Schlüssel vom

[1] Dies bedeutet primär, dass der Webdienst eventuelle Rückgaben hinzufügt.

Erzeuger der Ausgabe verschlüsselt und ebenfalls mit einem eindeutigen Bezeichner versehen.

5.3. Die Nutzlast-Schicht

Die Nutzlast-Schicht ist die zentrale Komponente des Containers. Im Wesentlichen ist sie für die Bereitstellung von Kontrollinformationen und die Verteilung symmetrischer Schlüssel an die eingebundenen Teilnehmer zuständig. Für jeden eingebundenen Webdienst enthält die Nutzlast-Schicht ein sogenanntes *parcel*. Jedes *parcel* beinhaltet den Prozessausschnitt für den jeweilig zugehörigen Webdienst, sowie einige zusätzliche Kontrollvariablen zur (Weiter-)Verarbeitung des Containers. Im Einzelnen sind dies:

- *functionality*: Die Signatur der auszuführenden Funktionalität.
- *input*: Die Bezeichner und symmetrische Zertifikat- beziehungsweise Ausgabe-Schlüssel zur Entschlüsselung erforderlicher Eingabe-Zertifikate.
- *output*: Bezeichner und symmetrische Ausgabe-Schlüssel zur Verschlüsselung der Rückgaben.
- *activity*: Die Kontrollflussstruktur, in die der Webdienst eingebettet ist, sowie Bedingungen, die ausgewertet werden müssen, um die Integrität der Ausführungsreihenfolge gewährleisten zu können. Beispielsweise muss bei einer bedingten Verzweigung entschieden werden, an welchen nachfolgenden Webdienst der Container verschickt werden muss.
- *partnerlink*: Die Referenz(en) auf nachfolgende Webdienste.

Diese Kontrollvariablen werden unter dem Begriff *Kontrollausschnitt* (*control view*) zusammengefasst. Um die Informationen eines *parcels* ausschließlich dem vorgesehenen Webdienst zur Verfügung zu stellen, erzeugt der initiierende Teilnehmer für jedes *parcel* eines Webdienstes einen symmetrischen *parcel*-Schlüssel. Jedes *parcel* wird daraufhin vom initiierenden Teilnehmer mit dem entsprechenden Parcel-Schlüssel symmetrisch verschlüsselt und das Ergebnis in der Nutzlast-Schicht abgelegt.

Um die Integrität der Ausführungsreihenfolge zu gewährleisten, wird zunächst der *parcel*-Schlüssel mit dem öffentlichen Schlüssel des jeweilig zugehörigen Webdienstes asymmetrisch verschlüsselt. Hierdurch wird gewährleistet, dass nur der vorgesehene Webdienst den benötigten *parcel*-Schlüssel kennen kann. Um weiterhin sicherzustellen, dass ein eingebundener Webdienst erst dann Zugriff auf ein *parcel* hat, wenn die zugehörige Funktionalität laut Spezifikation aufgerufen werden soll, wird der

verschlüsselte *parcel*-Schlüssel in dem *parcel* des unmittelbaren Vorgängers gespeichert. Ein Webdienst ist demnach erst dann in der Lage, sein *parcel* und somit die darin enthaltenen Kontrollinformationen zu entschlüsseln, nachdem sein unmittelbarer Vorgänger die geforderte Funktionalität ausgeführt und ihm anschließend der Container mitsamt verschlüsseltem *parcel*-Schlüssel übermittelt hat. Wie im Fall der Zertifikat- und Ausgabe-Schicht, wird jedem *parcel* ein eindeutiger Bezeichner zugeordnet, damit ein Webdienst sein *parcel* in der Nutzlast-Schicht auffinden kann. Zudem existiert in der Nutzlast-Schicht keine Reihenfolge der *parcels*, damit nicht hierüber Schlüsse über die Ausführungsreihenfolge gezogen werden können. Weiterhin signiert der initiierende Teilnehmer die Nutzlast-Schicht.

Der allgemeine Aufbau eines Containers ist wie folgt definiert.

Definition 5.4 (Container). *Für einen initiierenden Teilnehmer t und n eingebundene Teilnehmer ist der 3-Schichtenaufbau eines Containers durch*

$$((\textit{Zertifikat-Schicht}_{PK_t}, \textit{Nutzlast-Schicht}_{PK_t}), \textit{Ausgabe-Schicht})$$

gegeben. Der Aufbau der Nutzlast-Schicht durch entsprechend verschlüsselte parcels und zugehörige Bezeichner ist gegeben durch:

$$((l_1, Enc_{syk_1}(parcel_1)), (l_2, Enc_{syk_2}(parcel_2)), \ldots, (l_n, Enc_{syk_n}(parcel_n))).$$

Je nach Spezifikation der Ausführungsreihenfolge unterscheidet sich der innere Aufbau der *parcels* der Nutzlast-Schicht. Im Folgenden geben wir zwei Beispiele für eine sequenzielle und eine parallele Ausführungsreihenfolge von Webdiensten.

Beispiel 5.5 (sequenzielle Ausführung Webdiensten). Für eine Sequenz von n teilnehmenden Webdiensten $WD_1\, WD_2 \ldots WD_n$ und einem initiierenden Teilnehmer t, haben die *parcels* der Nutzlast-Schicht den Aufbau:

$$\begin{array}{lcl} parcel_i & = & (\textit{Kontrollausschnitt}_i, Enc_{K_{i+1}}(l_{i+1}, syk_{i+1})), \\ parcel_n & = & (\textit{Kontrollausschnitt}_n, t^2). \end{array}$$ ◊

Beispiel 5.6 (parallele Ausführung von Webdiensten). Für eine Menge von n teilnehmenden Webdiensten $\{WD_1, WD_2, \ldots, WD_n\}$ und einem initiierenden Teilnehmer t sollen die Webdienste WD_2 bis WD_{n-1} parallel ausgeführt werden. Wie in der Semantik

[2] In $parcel_n$ gibt der Bezeichner t anstelle eines symmetrischen Ausgabe-Schlüssels an, dass kein weiterer Webdienst eingebunden ist. Der Webdienst WD_n schickt somit den Container nach Ausführung der Funktionalität an den initiierenden Teilnehmer zurück.

einer BPEL Spezifikation zur parallelen Ausführung von Webdiensten vorgesehen, fungiert der Webdienst WD_1 als „Aufspaltungsstelle“ und der Webdienst WD_n als „Synchronisationsstelle“: $WD_1(\,WD_2 \| \ldots \| \,WD_{n-1})\,WD_n$.

Die *parcels* der Nutzlast-Schicht eines realisierenden Containers haben den Aufbau:

$$\begin{array}{lcl} parcel_1 & = & (Kontrollausschnitt_1, Enc_{K_2}(l_2, syk_2), \ldots, Enc_{K_{n-1}}(l_{n-1}, syk_{n-1})), \\ parcel_i & = & (Kontrollausschnitt_i, Enc_{K_n}(l_n, s_{i-1})), \text{ mit } 1 < i < n, \\ parcel_n & = & (Kontrollausschnitt_n, t). \end{array}$$

In $parcel_i$ bezeichnet s_{i-1} ein *Teilgeheimnis* des symmetrischen *parcel*-Schlüssels syk_n des Webdienstes WD_n. Zur Konstruktion eines realisierenden Containers erzeugt der initiierende Teilnehmer n Teilgeheimnisse für den Schlüssel syk_n. Nur unter Kenntnis aller n Teilgeheimnisse kann hieraus der *parcel*-Schlüssel syk_n konstruiert werden. Eingesetzt werden können dazu Verfahren der sogenannten *Geheimnisteilung* (*secret sharing*), siehe hierzu beispielsweise [73]. Sogenannte (n, n)-Schwellwert-Schemata sind für die Realisierung einer parallelen Struktur ausreichend. ◊

Beispiel 5.7 (bedingte Verzweigung von Webdiensten). Für eine Menge von n teilnehmenden Webdiensten $\{WD_1, WD_2, \ldots, WD_{n-1}, WD_n\}$ und einem initiierenden Teilnehmer t soll aus der Menge der Webdienste $\{WD_2, \ldots, WD_{n-1}\}$ nur einer zur Ausführung seiner Funktionalität kommen. Wie in der Semantik einer BPEL Spezifikation zur bedingten Verzweigung von Webdiensten vorgesehen, fungiert der Webdienst WD_1 als „Aufspaltungsstelle“ und der Webdienst WD_n als „Synchronisationsstelle“: $WD_1(\,WD_2 \oplus \ldots \oplus \,WD_{n-1})\,WD_n$.

Die *parcels* der Nutzlast-Schicht eines realisierenden Containers haben den Aufbau:

$$\begin{array}{lcl} parcel_1 & = & (Kontrollausschnitt_1, Enc_{K_2}(l_2, syk_2), \ldots, Enc_{K_{n-1}}(l_{n-1}, syk_{n-1})), \\ parcel_i & = & (Kontrollausschnitt_i, Enc_{K_n}(l_n, syk_n)), \\ parcel_n & = & (Kontrollausschnitt_n, t). \end{array}$$

◊

Die Abbildung 5.1 veranschaulicht einen Containeraufbau mit den drei Schichten. Der detailliert beschriebene Ansatz kann in der Veröffentlichung [2] nachgelesen werden. Relevant für den in dieser Arbeit vorgestellten Ansatz ist die vorgeschlagene Datenstruktur, der Container, die in Abschnitt 9.2 Grundlage für eine echt dezentrale Realisierung zustandsdynamischer Politiken sein wird. In dem dortigen Abschnitt wird auch detailliert auf den konkreten Einsatz eines Containers eingegangen.

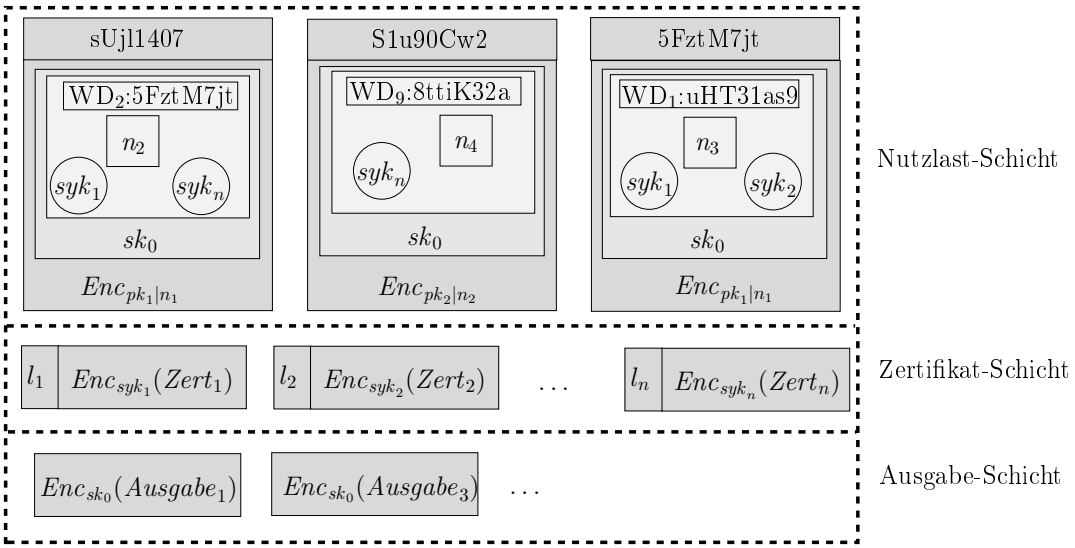


Abbildung 5.1.: 3-Schichtenaufbau eines Containers.

Teil III.

Zustandsdynamische Politiken

In diesem Teil wird ein Rahmenwerk für eine Zugriffskontrolle entworfen, die auf den Konzepten kompositionaler und zustandsbasierter Zugriffskontrollmodelle aufbaut, und diese zu einer dynamischen Zugriffskontrollpolitik vereint.

Kapitel 6.

Domänenkonfigurationen

Im vorherigen Kapitel 3.3 haben wir beispielhaft gezeigt, wie eine kompositionale Zugriffskontrollpolitik für virtuelle Organisationen spezifiziert werden kann, sofern die in den involvierten Organisationseinheiten angebotenen Dienste bzw. Objekte atomar sind. Betrachten wir allerdings komplexe Folgen der Funktionalitäten angebotener Dienste, so muss eine geeignete Zugriffskontrolle auch diese kontrollieren und überwachen können. In diesem Kapitel spezifizieren wir eine Zugriffskontrolle, welche es Verwaltern von Schutzdomänen erlaubt, einfache zustandsabhängige Zugriffskontrolle explizit kompositional zu spezifizieren und durchzusetzen.

Basierend auf diskretionärer Zugriffskontrolle gehen wir von einem Tupel $\mathcal{E} = (S, O, A)$ aus, für das S die Menge aller Subjekte, O die Menge aller Objekte und A die Menge der atomaren Aktionen ist, welche durch die Subjekte auf den Objekten ausgeführt werden können.

Definition 6.1 (mögliche Ausführungssequenz). *Die Menge* $\mathcal{T} = (S \times O \times A)^\star$ *bestimmt die möglichen Ausführungssequenzen.*

Ein Verwalter kann von der Menge $\mathcal{T}$ der möglichen Ausführungssequenzen bestimmte auswählen, die er innerhalb seiner Schutzdomäne als *erlaubt* ansieht. Die Nutzung der in einer *erlaubte Ausführungssequenz* vorkommenden elementaren Aktivitäten soll nur in der angegebenen Reihenfolge gewährt werden. Bezüglich seiner Schutzdomäne spezifiziert der Verwalter eine Menge erlaubter Ausführungssequenzen, die *Auswahlmenge*. Einem Dienstnutzer wird die *schrittweise* erfolgende Auswahl genau einer erlaubten Ausführungssequenz aus der Auswahlmenge erlaubt. Hierbei werden konzeptionell alle anderen erlaubten Ausführungssequenzen der Nutzbarkeit entzogen.

Die möglichen Ausführungssequenzen bilden die elementaren Einheiten eines zugrunde liegenden *Rechensystems*.

Definition 6.2 (Rechensystem). *Das Rechensystem* $\mathcal{S} \subset \mathcal{T}$ *wird gegeben durch eine Teilmenge der möglichen Ausführungssequenzen.*

Betrachten wir zunächst lokal autonome Organisationseinheiten des Rechensystems. Jede Organisationseinheit unterliegt als Schutzdomäne der Kontrolle ihres Verwalters. In einem ersten Schritt geben wir dem Verwalter die Möglichkeit, von ihm innerhalb seiner Organisationseinheit im Sinne der Zugriffskontrolle als *erlaubt* angesehene Ausführungssequenzen, eine sogenannte *Domänenkonfiguration* zu definieren. Wir weisen hier darauf hin, dass wir im Folgenden nicht unterscheiden, ob die für eine Schutzdomäne definierte(n) erlaubte(n) Ausführungssequenz(en) nur vom Verwalter dieser oder von entsprechend delegierten Beauftragten aufgestellt wurde(n). Sobald diese Unterscheidung im Kontext unseres Ansatzes relevant wird, werden wir diese explizit einführen.

Definition 6.3 (Domänenkonfiguration). *Eine Domänenkonfiguration $\mathcal{C} \subset \mathcal{T}$ bestimmt die erlaubten Ausführungssequenzen und wird gegeben durch eine Teilmenge der möglichen Ausführungssequenzen.*

Offensichtlich ist eine Domänenkonfiguration nur in solchen Fällen sinnvoll, wenn sie vom zugrunde liegenden Rechensystem angebotene Ausführungssequenzen bestimmt.

Definition 6.4 (reale Domänenkonfiguration). *Eine Domänenkonfiguration $\mathcal{C}$ ist für ein Rechensystem S real, wenn $\mathcal{C} \cap \mathcal{S} \neq \emptyset$ gilt.*

Sofern wir nicht explizit darauf eingehen, sprechen wir im Folgenden ausschließlich über reale Domänenkonfigurationen. Des Weiteren verwenden wir die Kurzversion „Konfiguration“ synonym mit „Domänenkonfiguration“, sofern der Kontext ersichtlich ist.

Beispiel 6.5 (Forschungsdatenbank). In Kapitel 3.3 wurde eine Forschungsdatenbank als Beispiel einer virtuellen Organisation eingeführt. Die dort kombinierte gesamte Zugriffskontrollpolitik ist durch

$$P_{ViBib} = (P_{TUDoBib} - \overline{P}_{TUDoBib}) + (P_{RUBib} - \overline{P}_{RUBib}) + (P_{CityBib} - \overline{P}_{CityBib})^{\wedge} [s = \mathit{Forschungsbezug}]$$

gegeben. Zur Vereinfachung des Szenarios berücksichtigt die Zugriffskontrollpolitik P_{ViBib} nur atomare Lesezugriffe auf den Objekten, welche der virtuellen Organisation angehören. Die Komposition wird auf Mengen solcher atomaren Erlaubnisse angewendet, so dass eine Instanz der kombinierten Zugriffspolitik durch eine Menge atomarer Erlaubnisse bezüglich Lesezugriff gegeben ist. Typischerweise ist es in einer Bibliothek möglich, gefundene Dokumente auszuleihen oder zu erwerben. Da sowohl der Vorgang der Ausleihe als auch der Vorgang eines Kaufes nicht mehr atomare Zugriffsoperationen sondern Abläufe von solchen sind, kann eine geeignete kombinierte Zugriffskontrollpolitik nicht mehr in der in Abschnitt 2.2.1 vorgestellten Spezifikation erfolgen. In diesem Beispiel veranschaulichen wir, wie eine mögliche Domänenkonfiguration für die

Organisationseinheit TUDoBib spezifiziert werden kann. Auch in diesem Beispiel nutzen wir das Rollenkonzept für Subjekte, ohne auf die konkrete Umsetzung einzugehen: Wir bezeichnen einen registrierten Nutzer der Hochschulbibliothek im Folgenden als *mitglied*. Des Weiteren gehen wir von einer Klasse von Objekten beziehungsweise Dokumenten aus, ohne zu differenzieren, dass bestimmte Dokumente unterschiedliche Zugriffsoperationen anbieten und unterschiedlichen Zugriffskontrollpolitiken unterliegen können. Wir legen die folgenden, möglichen Zugriffe zugrunde:

$$
\begin{aligned}
\alpha &= (mitglied, dokument, lesen),\\
\beta &= (mitglied, dokument, warenkorb\_legen),\\
\gamma &= (mitglied, dokument, kauf\_rechnung),\\
\delta &= (mitglied, dokument, kauf\_kreditkarte),\\
\varepsilon &= (mitglied, dokument, bestellen),\\
\zeta &= (mitglied, dokument, warenkorb\_entfernen).
\end{aligned}
$$

Mit den obigen möglichen Zugriffen kann ein einfacher Ablauf spezifiziert werden, in dem ein Nutzer ein gewünschtes Dokument zunächst liest, sich dann dazu entschließt dieses zu kaufen und nach Festlegung der Bezahlmodalität es schließlich verbindlich bestellt. Mögliche Domänenkonfigurationen können wie folgt spezifiziert werden:

$$Kauf_R = \{\alpha\beta\gamma\varepsilon\},\ Kauf_K = \{\alpha\beta\delta\varepsilon\}.$$

Die Domänenkonfiguration $Kauf_R$ legt einen Ablauf fest, in dem ein Nutzer auf Rechnung bezahlt, die Domänenkonfiguration $Kauf_K$ legt einen Ablauf fest, in dem ein Nutzer mit Kreditkarte bezahlt. ◊

6.1. Spezifizierung

Sei $\varepsilon_i = (s_i, o_i, a_i)$ eine elementare Erlaubnis und seien $t, \lambda \in \mathcal{T}$, wobei wir mit λ die *leere Ausführungssequenz* kennzeichnen. Wir vereinbaren die folgenden Funktionen.

$sbj(\varepsilon)$	liefert s_i;
$obj(\varepsilon)$	liefert o_i;
$akt(\varepsilon)$	liefert a_i;
$length(t)$	liefert die Anzahl der elementaren Erlaubnisse in t;
$kopf(t)$	liefert das erste Element der übergebenen Sequenz;
$kopf(\lambda)$	liefert λ;
$kopf^i$	sei die Notation für die i-fache Anwendung von $kopf$;
$rumpf(t)$	liefert die übergebene Sequenz ohne das erste Element;
$rumpf(t_1)$	liefert λ für $length(t_1) = 1$;
$rumpf(\lambda)$	liefert λ;
$rumpf^i$	sei die Notation für die i-fache Anwendung von $rumpf$;
$cond(t)$	liefert die Auswertung der Bedingung $cond$ im Hinblick darauf, ob die Ausführungssequenz t sie erfüllt oder nicht;

Die folgenden Operatoren dienen dem Spezifizieren von Domänenkonfigurationen.

$\cup$	Der Vereinigungsoperator liefert die *mengentheoretische Vereinigung* zweier Konfigurationen.
$\cap$	Der Durchschnittsoperator liefert den *mengentheoretischen Durchschnitt* zweier Konfigurationen.
$\setminus$	Der Differenzoperator liefert die *mengentheoretische Differenz* zweier Konfigurationen.
$\bullet$	Der Konkatenationsoperator liefert die *Verknüpfung* zweier Konfigurationen zu einer neuen, welche alle Verknüpfungen einer Ausführungssequenz der ersten Konfiguration mit einer Ausführungssequenz der zweiten enthält.
$\|\|\|$	Der Verschränkungsoperator liefert die *Verschränkung* zweier Konfigurationen zu einer neuen, welche alle Ausführungssequenzen enthält, die eine sequentielle Verzahnung einer Ausführungssequenz der ersten Konfiguration mit einer Ausführungssequenz der zweiten enthält.
$*$	Der unäre *Kleene-Stern* liefert den *Kleeneschen Abschluss* einer übergebenen Konfiguration.
σ_{cond}	Der unäre Auswahloperator liefert diejenigen Ausführungssequenzen einer übergebenen Konfiguration, welche die angegebene Bedingung $cond$ erfüllen.
$rumpf$	Der unäre Schnittoperator wendet die rumpf-Funktion auf jeder Ausführungssequenz einer übergebenen Konfiguration an.

Beispiel 6.6 (Forschungsdatenbank). Falls ein Nutzer der Hochschulbibliothek TU-DoBib ein gefundenes Dokument nur lesen, nicht aber kaufen möchte, muss hierfür ebenfalls die Möglichkeit bestehen. Ebenso muss die Hochschulbibliothek es ermöglichen, einen Kauf abzubrechen. Wir spezifizieren hierfür die Domänenkonfiguration

$$Abbruch \quad = \quad \{\alpha\beta\zeta\},$$

welche aus nur einer Ausführungssequenz besteht. Eine kombinierte Domänenkonfiguration für die Recherche eines Nutzers kann wie folgt spezifiziert werden:

$$Recherche \quad = \quad (Kauf_R \cup Kauf_K \cup Abbruch)^*.$$

Die Domänenkonfiguration *Recherche* erlaubt eine beliebige Abfolge der verwendeten Domänenkonfigurationen $Kauf_R$, $Kauf_K$ und *Abbruch*. ◊

Beispiel 6.7 (diskretionäre Zugriffskontrolle). In klassischen diskretionären Zugriffskontrollmodellen, wie in Abschnitt 2.1.1 vorgestellt, werden atomare Erlaubnisse der Form (s, o, a) vergeben. Übertragen wir dieses Modell auf unseren Ansatz, so lässt sich diese Art der Zugriffskontrolle durch eine Domänenkonfiguration $\{(s, o, a)\}^*$ spezifizieren. Diese Domänenkonfiguration entspricht in ihrer Semantik der eines Elementes der Zugriffsrelation *ZR*. Das Subjekt s ist autorisiert die *capability* (o, a) ohne einschränkende Bedingungen und unabhängig von anderen Erlaubnissen der Zugriffsrelation zu nutzen. Um die gesamte Zugriffsrelation abzudecken, spezifizieren wir für jeden Eintrag $\varepsilon_i = (s, o, a)_i \in ZR$ eine entsprechende Domänenkonfiguration $\{\varepsilon_i\}^*$. Die Domänenkonfigurationen werden dann zu

$$\{\varepsilon_1\}^* ||| \ldots ||| \{\varepsilon_m\}^*$$

kombiniert, um das Zugriffskontrollmodell abzubilden. ◊

Beispiel 6.8 (RBAC). Da unser Ansatz Zugriffskontrolle für verteilte Organisationen bieten soll, kann realistischerweise nicht davon ausgegangen werden, dass die möglichen Nutzer identifizierte Subjekte sind. Bei einer großen Anzahl potentieller Nutzer bietet sich das Rollenkonzept für die Spezifizierung und Verwaltung der entsprechenden Zugriffsrechte an. In einem solchen Modell abstrahieren wir für die Spezifikation atomarer Erlaubnisse von identifizierten Subjekten und verwenden stattdessen *freie Merkmale*, wie sie in Kapitel 4 eingeführt wurden. Die freien Merkmals fungieren dann als Variablen, die vor einer tatsächlichen Zugriffsentscheidung an identifizierte Subjekte gebunden werden. Den Formalismus betreffend betrachten wir weiterhin identifizierte Subjekte, werden aber in Kapitel 8.2 die Einsetzbarkeit der Rollenkonzepte in unserem Ansatz zu zeigen. ◊

Beispiel 6.9 (Chinese-Wall Ansatz). In Abschnitt 2.1.2 wurde der Chinese-Wall Ansatz, siehe [23], grob umrissen. Wir geben hier ein Beispiel, mit dem verdeutlicht wird, dass dieser zustandsbasierte Ansatz systembestimmter Zugriffskontrolle ebenfalls mittels Domänenkonfigurationen ausgedrückt werden kann. Wir betrachten einen Berater

s (Subjekt), ein Objekt o_1, welches zum Unternehmen U_1 gehört und ein Objekt o_2, welches zum Unternehmen U_2 gehört. Die Unternehmen gehören der gleichen CoI-Klasse an, so dass o_1 in einem Interessenkonflikt zu dem Objekt o_2 steht. Die Domänenkonfiguration

$$\begin{aligned} \mathit{Chinese\_Wall} = \quad & \{(s, o_1, \mathit{lesen})\} \bullet \{(s, o_1, \mathit{lesen}), (s, o_1, \mathit{schreiben})\}^* \cup \\ & \{(s, o_2, \mathit{lesen})\} \bullet \{(s, o_2, \mathit{lesen}), (s, o_2, \mathit{schreiben})\}^* \end{aligned}$$

erreicht eine Zugriffskontrolle gemäß des Chinese-Wall Ansatzes. Zu Beginn kann ein Berater s frei wählen, ob er einen Lesezugriff auf Objekt o_1 oder auf Objekt o_2 ausführen möchte. Sobald allerdings ein Zugriff ausgeführt wurde, kann der Berater s nicht mehr auf das andere Objekt zugreifen, welches zu dem anderen Unternehmen der gleichen CoI-Klasse gehört. ◊

6.2. Deklarative Zugriffskontrolle

Mit dem Entwurf der Domänenkonfigurationen haben wir ein Instrument entwickelt, mit dem ein Verwalter einer Domäne sowohl traditionelle Zugriffskontrolle im Sinne von atomaren Zugriffsrechten als auch (einfache) zustandsbasierte Zugriffskontrolle einsetzen kann. Betrachten wir den Aufbau einer Schutzdomäne, wie in Abbildung 1.2 dargestellt, so gehen wir davon aus, dass die Zugriffe auf die zu schützenden Objekte durch eine Zugriffskontrollinstanz geschützt wird. Jeder Teilnehmer des Rechensystems kann eine Anfragesequenz t an den Zugriffspunkt einer Schutzdomäne stellen. Alle eingehenden Anfragesequenzen werden elementweise bearbeitet. Eine elementare Aktivität wird als *erlaubt* bezeichnet, wenn die Zugriffskontrollinstanz die Ausführung der entsprechenden Aktion durch das entsprechende Subjekt auf dem entsprechenden Objekt erlaubt. Eine Teilsequenz einer Anfragesequenz wird als erlaubt bezeichnet, wenn alle enthaltenen Aktivitäten erlaubt sind.

Mit der Spezifikation einer Domänenkonfiguration legt der Verwalter einer Schutzdomäne implizit ein gewünschtes Verhalten der Zugriffskontrollkomponenten im Hinblick auf die Abarbeitung eingehender Anfragesequenzen fest.

Annahme 6.10. *Für jede Anfragesequenz* $(s_1, o_1, a_1) \ldots (s_i, o_i, a_i) \ldots (s_n, o_n, a_n)$ *sind genau die Teilsequenzen erlaubt, die Präfix mindestens einer erlaubten Ausführungssequenz der Domänenkonfiguration sind. Alle übrigen Anfrageteilsequenzen werden abgelehnt.*

Um das gewünschte Verhalten einer Zugriffskontrolle zu operationalisieren, sind im Wesentlichen zwei Kernpunkte wichtig. Zum einen muss die Zugriffskontrollinstanz

sequenziell für jede gewünschte Aktivität einer vorliegenden Anfragesequenz aufgrund der zugrunde liegenden Domänenkonfiguration entscheiden, ob die gewünschte(n) Aktivität(en) erlaubt sind. Zum anderen wird nach jeder Entscheidung über eine gewünschte Aktivität der Anfragesequenz vor erneuter Entscheidung die vorliegende Domänenkonfiguration entsprechend aktualisiert.

Für die folgenden Definitionen sei $\mathcal{C}$ eine Domänenkonfiguration und $\varepsilon = (s, o, a)$ eine Anfrage.

Definition 6.11 (entscheide-Prädikat). *Das entscheide-Prädikat wird gegeben durch*

$$\mathit{entscheide}(\mathcal{C}, \varepsilon) \equiv (\exists t)\ [t \in \mathcal{C}, \mathit{kopf}(t) = \varepsilon].$$

Definition 6.12 (Domänenkonfigurationsupdate). *Das Aktualisieren einer Domänenkonfiguration wird gegeben durch*

$$\mathit{aktualisiere}(\mathcal{C}, \varepsilon) = \{\mathit{rumpf}(t) \mid t \in \mathcal{C}\ \mathit{and}\ \mathit{kopf}(t) = \varepsilon\}.$$

Das entscheide-Prädikat zusammen mit dem Domänenkonfigurationsupdate beschreibt das Verhalten einer Zugriffskontrolle wie folgt.

Definition 6.13 (Verhalten der Zugriffskontrolle).

Eingabe: $\mathcal{C}$ *[vorliegende] Konfiguration*
ε *Anfrage*

Ausgabe: d *Boolean [Zugriffsentscheidung]*
$\mathcal{D}$ *[aktualisierte] Konfiguration*

Methode: $d := \mathit{entscheide}(\mathcal{C}, \varepsilon)$;
$\mathcal{D} :=$ `if` d `then` $\mathit{aktualisiere}(\mathcal{C}, \varepsilon)$ `else` $\mathcal{C}$;

Ein Verwalter, der eine Domänenkonfiguration spezifiziert, möchte sicher gehen, dass seine in Annahme 6.10 getroffene Annahme über das entsprechende Verhalten der Zugriffskontrollkomponenten seiner Schutzdomäne auch tatsächlich erfüllt wird.

Theorem 6.14. *Das Verhalten der Zugriffskontrollkomponente laut Definition 6.13 entspricht dem von einem Verwalter implizit angenommenen Verhalten, siehe Annahme 6.10.*

Beweis. Wir gehen davon aus, dass das Aktualisieren einer vorliegenden Konfiguration nicht durch das Abtrennen der entsprechenden Sequenzköpfe, sondern durch

deren Markieren stattfindet. Hierfür vereinbaren wir zwei neue Funktionen $kopf_{nm}$ und $kopf_{mark}$, und wir ersetzen das bisherige Domänenkonfigurationsupdate (siehe Definition 6.12) *aktualisiere* durch eine Vorschrift $aktualisiere_{mark}$:

$kopf_{nm}(t)$	liefert das erste nicht markierte Element der übergebenen Ausführungssequenz t;
$kopf_{mark}(t)$	liefert die Ausführungssequenz t so zurück, dass das erste nicht markierte Element der Sequenz t zusätzlich markiert ist;
$aktualisiere_{mark}(\mathcal{C}, \varepsilon) =$	$\{kopf_{mark}(t) \mid t \in \mathcal{C}$ und $kopf_{nm}(t) = \varepsilon\}$.

Laut Definition 6.11 ist das entscheide-Prädikat genau dann wahr, wenn die Anfrage erstes Element mindestens einer erlaubten Ausführungssequenz der vorliegenden Konfiguration ist. Im Fall eines wahren entscheide-Prädikats markiert „aktualisiere$_{mark}$“ die entsprechenden Sequenzköpfe der vorliegenden Ausführungssequenz:

$$\texttt{if } (entscheide(\mathcal{C}, \varepsilon)) \texttt{ then } aktualisiere_{mark}(\mathcal{C}, \varepsilon).$$

Wir zeigen per Induktion über die Länge n einer Anfragesequenz $t = \varepsilon_1 \ldots \varepsilon_i \ldots \varepsilon_n$, dass die von der Zugriffskontrolle erlaubten Teilsequenzen der Anfragesequenz der in der vorliegenden Konfiguration markierten Ausführungssequenz(en) entspricht (entsprechen).

1. Induktionsanfang. Für eine Anfragesequenz der Länge 1, das bedeutet $t = \varepsilon_1$, gilt das Theorem:

   a) Das entscheide-Prädikat ist wahr.
   Es existiert mindestens eine erlaubte Ausführungssequenz $t_i \in \mathcal{C}$ in der vorliegenden Konfiguration, so dass $kopf_{nm}(t_i) = \varepsilon_1$. Laut Definition 6.13 wird in diesem Fall das Aktualisieren der Domänenkonfiguration durchgeführt. Das Aktualisieren wurde durch die Markierung $aktualisiere_{mark}$ ersetzt, welche die entsprechende(n) Ausführungssequenz(en) der vorliegenden Konfiguration korrekt markiert.

   b) Das entscheide-Prädikat ist falsch.
   Es existiert keine erlaubte Ausführungssequenz $t_i \in \mathcal{C}$ in der vorliegenden Konfiguration, so dass $kopf_{nm}(t_i) = \varepsilon_1$ gilt. Laut Definition 6.13 wird in diesem Fall die vorliegende Konfiguration unverändert beibehalten. Es findet also keine Markierung von Ausführungssequenzen statt.

2. Induktionsannahme. Für alle Anfragesequenzen der Länge n gilt das Theorem 6.14.

3. Induktionsschritt. Wir betrachten eine Anfragesequenz t der Länge $n+1$, das bedeutet $t = \varepsilon_1 \ldots \varepsilon_i \ldots \varepsilon_n \ \varepsilon_{n+1}$. Auf Grund der Induktionsannahme 2. existiert mindestens eine markierte, erlaubte Ausführungssequenz $t_i \in \mathcal{C}$ in der vorliegenden Konfiguration, deren markierte Teilsequenz genau der bisher erlaubten Teilanfragesequenz der Länge n entspricht, $t_i = \varepsilon_1^{mark} \ldots \ \varepsilon_i^{mark} \ldots \ \varepsilon_n^{mark} \ \varepsilon_{n+1} \ \ldots$. Sei $(s_{n+1}, o_{n+1}, a_{n+1})$ der nächste angefragte Zugriff. Die Argumentation verläuft analog zum Induktionsanfang 1.:

   a) Das entscheide-Prädikat ist wahr.
   Die erste nicht markierte elementare Erlaubnis in t_i entspricht dem angefragten Zugriff, das bedeutet $\varepsilon_{n+1} = (s_{n+1}, o_{n+1}, a_{n+1})$. Laut Definition 6.13 wird die vorliegende Konfiguration aktualisiert; hier markiert $aktualisiere_{mark}$ die elementare Erlaubnis ε_{n+1}, so dass $t_i = \varepsilon_1^{mark} \ldots \ \varepsilon_i^{mark} \ \ldots \ \varepsilon_n^{mark} \ \varepsilon_{n+1}^{mark} \ \ldots$.

   b) Das entscheide-Prädikat ist falsch.
   In diesem Fall ist $\varepsilon_{n+1} \neq (s_{n+1}, o_{n+1}, a_{n+1})$, da die erste nicht markierte elementare Erlaubnis in t_i nicht dem angefragten Zugriff entspricht. Laut Definition 6.13 wird die vorliegende Konfiguration unverändert beibehalten und es findet keine Markierung statt.

□

6.3. Eine Operationalisierung der Zugriffskontrolle

Um eine Zugriffskontrolle mittels Domänenkonfigurationen effizient und berechenbar zu operationalisieren, müssen Spezifizierungen von und Berechnungen auf Konfigurationen algorithmisiert werden können. Ein Ansatz hierfür ist, die Automatentheorie zugrunde zu legen. Um eine Möglichkeit der Operationalisierung vorzustellen, betrachten wir das Spezifizieren von Konfigurationen gemäß *regulärer Ausdrücke*. Für diesen eingeschränkten Fall lassen sich Zugriffsentscheidungen auf entsprechende *akzeptierende Automaten* beziehungsweise auf entsprechende Konstruktionen von diesen abbilden. Die Umsetzung kann dann mit den üblichen Verfahren zur Konstruktion akzeptierender Automaten für eine von einem regulären Ausdruck erzeugten Sprache geschehen, siehe beispielsweise [68].

Zur Vorbereitung wird jeder mögliche Zugriff der Form (s, o, a) als ein Buchstabe eines Alphabets einer formalen Sprache verwendet. Somit wird jede Domänenkonfiguration $\mathcal{C}$ durch einen regulären Ausdruck G mit dem Alphabet $\mathcal{E}$ erzeugt, so dass gilt $\mathcal{C} = Lan(G)$. Diese modifizierte Spezifikation wird in einen deterministischen

endlichen Automaten (*deterministic finite automaton*, DFA) $(Q, \Sigma, \delta, q_0, E)$ überführt, dessen akzeptierte Sprache $Lan(DFA) = Lan(G)$ ist. Ein solcher DFA lässt sich wie folgt angeben.

- Q ist die endliche Menge der Zustände des Automaten.
- $\Sigma \subset \mathcal{E}$ ist das Eingabealphabet.
- $\delta : Q \times \Sigma \to Q$ ist die Übergangsfunktion. Für q_i, q_j, $q_F \in Q$ und eine Eingabe $\varepsilon \in \Sigma$ gilt:
  $\delta(q_i, \varepsilon) = q_j$, falls der Zustandsübergang aufgrund des Ausdrucks G durchgeführt werden kann,
  $\delta(q_i, \varepsilon) = q_F$ sonst.
  Für den Fangzustand q_F gilt $\delta(q_F, \varepsilon) = q_F$ für alle $\varepsilon \in \Sigma$.
- $q_0 \in Q$ ist der Startzustand.
- $E \subseteq Q$ ist die Menge der akzeptierenden Endzustände.

Für die Operationalisierung der Zugriffskontrolle muss für eine Anfragesequenz t und eine gegebene Konfiguration $\mathcal{C}$ schrittweise entschieden werden, ob t als erlaubt angesehen wird (im Sinne von Annahme 6.10 und Theorem 6.14). Hierfür werden das Akzeptanzverhalten im Hinblick auf die Zugriffskontrolle und die Eingaben, welche den Automaten in den Fangzustand q_F überführen, gesondert betrachtet:

Wir definieren statt des Fangzustandes q_F eine endliche Menge Q_F von Fangzuständen des Automaten, so dass zu jedem Zustand $q_i \in Q$ des Automaten ein Fangzustand $q_{iF} \in Q_F$ existiert. Die Übergangsfunktion des Automaten wird dann durch $\delta : (Q \cup Q_F) \times \Sigma \to (Q \cup Q_F)$ folgendermaßen redefiniert:

- Für q_i, $q_j \in Q$, $q_{iF} \in Q_F$ und eine Eingabe $\varepsilon \in \Sigma$ gilt:
  $\delta(q_i, \varepsilon) = q_j$, falls der Zustandsübergang aufgrund des Ausdrucks G durchgeführt werden kann,
  $\delta(q_i, \varepsilon) = q_{iF}$ sonst.
  Für einen Fangzustand q_{iF} gilt $\delta(q_{iF}, \varepsilon) = \delta(q_i, \varepsilon)$ für alle $\varepsilon \in \Sigma$.

Die Einführung der Fangzustände erlaubt dann die Operationalisierung der Zugriffskontrolle wie folgt. Betrachten wir eine Anfragesequenz t, eine gegebene Konfiguration $\mathcal{C}$ und den akzeptierenden Automaten. Führt eine Eingabe ε aus t zu einem Zustandsübergang $\delta(q_i, \varepsilon) = q_j$ und ist $q_j \in Q$, so wird die durch ε definierte elementare Aktivität als erlaubt angesehen und ausgeführt. Führt eine Eingabe ε aus t zu einem Zustandsübergang $\delta(q_i, \varepsilon) = q_{jF}$ und ist $q_{jF} \in Q_F$, so wird die durch ε definierte elementare Aktivität nicht als erlaubt angesehen.

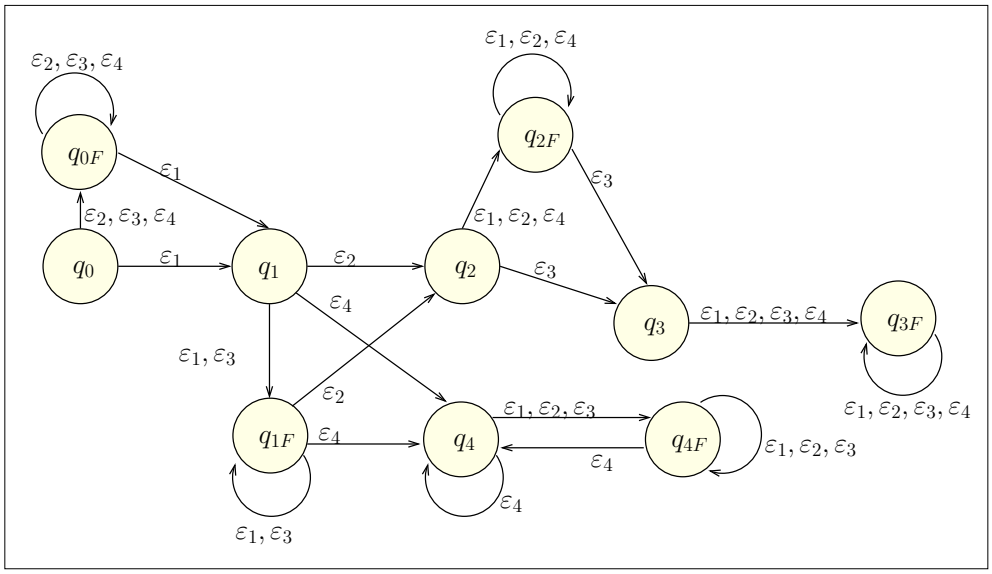


Abbildung 6.1.: Operationalisierung der Zugriffskontrolle durch einen Automaten mit Fangzuständen.

Ist also die Anfragesequenz t Präfix mindestens einer erlaubten Ausführungssequenz der Konfiguration $\mathcal{C}$, wird dieses von dem Automaten verifiziert.

Beispiel 6.16 (Akzeptor einer Zugriffskontrolle). Die Abbildung 6.1 zeigt die Darstellung eines Automaten, der zur Operationalisierung der Zugriffskontrolle für die wie folgt definierte Domänenkonfiguration C dient:

$$\begin{array}{lll} C & = & \{\varepsilon_1\} \bullet (\{\varepsilon_2\varepsilon_3\} \cup \{\varepsilon_4\}^*) \\ & \text{mit} & \mathcal{E} = \{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4\} \ . \end{array}$$ ◊

Für die bessere Übersichtlichkeit der grafischen Darstellung der Automaten wird im Folgenden dieser Arbeit in den Abbildungen auf die Darstellung der Fangzustände verzichtet. Die Konstruktion der einfacheren Darstellung erfolgt durch das Ausblenden der Zustandsübergänge, deren Eingabe im Sinne der Zugriffskontrolle nicht zu einer erlaubten Aktivität führt. Die Abbildung 6.2 visualisiert diese Darstellung für den Beispielautomaten aus Abbildung 6.1.

Um eine berechenbare und effiziente Zugriffskontrolle für Domänenkonfigurationen zu gewährleisten, betrachten wir nun die Aspekte *Berechenbarkeit* und *Effizienz*.

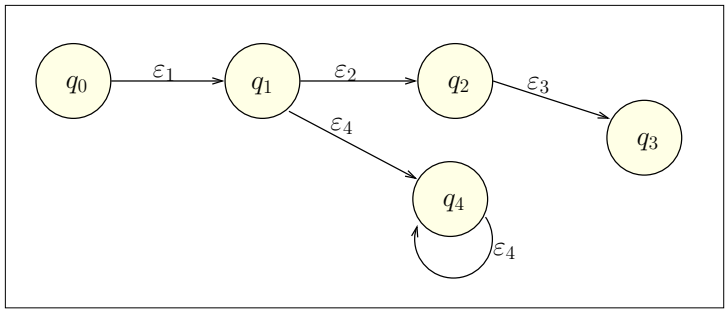


Abbildung 6.2.: Automat ohne Fangzustände.

Berechenbarkeit

Anforderung 6.17 (Berechenbarkeit). *Das für eine Zugriffskontrolle für Domänenkonfigurationen notwendige „entscheide-Prädikat“ (siehe Definition 6.11) und das „Domänenkonfigurationsupdate“ (siehe Definition 6.12) müssen für entsprechende akzeptierende Automaten algorithmisiert werden können.*

Wie schon in den zuvor gegebenen Beispielen erkennbar, muss die Zugriffskontrollinstanz die Zugriffsentscheidungen für solche endliche Automaten treffen, die durch die Spezifizierung einer Konfigurationen gegebenen sind.

Es wird sequenziell für jeden Zugriffswunsch, der einer Eingabe für den Automaten entspricht, eine *Zustandskontrolle* durchgeführt. Diese Zustandskontrolle entscheidet, ob die Ausführung der Eingabe im aktuellen Zustand des Automaten erlaubt ist. Eine Eingabe ε in einem aktuellen Zustand q_i wird dann als erlaubt angesehen, wenn die Zustandsüberführung nicht in einen Fangzustand des Automaten führt, das heißt $\delta(q_i, \varepsilon) \in Q$. Andernfalls führt die Zustandsüberführung in einen Fangzustand, $\delta(q_i, \varepsilon) \in Q_F$, und die Eingabe wird nicht als erlaubt angesehen. Gemäß dieser Vorgehensweise wird die für den deklarativen Teil definierte Zugriffskontrolle, siehe Abschnitt 6.2, im Folgenden für die Operationalisierung angepasst. Sei $\mathcal{A}$ ein Automat, q_i der aktuelle Zustand des Automaten, δ die Übergangsfunktion und $\varepsilon = (s, o, a)$ eine Eingabe.

Definition 6.18 (Zustandskontrolle). *Die Zustandskontrolle wird gegeben durch*

$$\mathit{entscheide}(\mathcal{A}, q_i, \varepsilon) \equiv [\delta(q_i, \varepsilon) \in Q].$$

Mit Hilfe der Zustandskontrolle kann dann das Verhalten der Zugriffskontrolle für

akzeptierende Automaten analog zum Zugriffskontrollverhalten auf deklarativer Ebene, siehe Definition 6.13, definiert werden.

Definition 6.19 (Verhalten der Zugriffskontrolle für akzeptierende Automaten).

Eingabe:	$\mathcal{A}$	*[vorliegender] Automat*
	q_i	*aktueller Zustand des vorliegenden Automaten*
	ε	*Anfrage*
Ausgabe:	d	*Boolean [Zugriffsentscheidung]*
	$q_i{}'$	*[aktualisierter] aktueller Zustand des vorliegenden Automaten*
Methode:	$d := entscheide(\mathcal{A}, q_i, \varepsilon);$	
	$q_i{}' := \delta(q_i, \varepsilon)$	

Effizienz der Operationalisierung

Die unter Abschnitt 6.1 angegebenen Operatoren zur Spezifizierung von Domänenkonfigurationen müssen ebenfalls auf der Ebene der Operationalisierung betrachtet werden.

Anforderung 6.20 (Effizienz). *Alle Operationen, die der Spezifizierung von Domänenkonfigurationen dienen, haben eine Entsprechung auf Ebene der regulären Ausdrücke. Dies bedeutet, dass für eine Konfigurationsoperation op eine entsprechende* Ausdrucksoperation op_A *existiert, wobei*

$$Lan(RA_1\ op_A\ RA_2) = Lan(RA_1)\ op\ Lan(RA_2)$$

gelten soll.

Betrachten wir Domänenkonfigurationen, die gemäß regulärer Ausdrücke spezifiziert werden, so können wir auf die Abschlusseigenschaften regulärer Sprachen zurückgreifen. Betrachten wir zunächst den Vereinigungsoperator. Seien $\mathcal{A}_1$ und $\mathcal{A}_2$ Automaten, die $Lan_1 = Lan(\mathcal{A}_1)$ beziehungsweise $Lan_2 = Lan(\mathcal{A}_2)$ akzeptieren. Hierfür wird ein Vereinigungsautomat konstruiert, der die Sprache $Lan_1 \cup Lan_2$ akzeptiert. Da die regulären Sprachen unter Vereinigung, Durchschnitt, Differenz, Konkatenation und dem Kleeneschen Abschluss abgeschlossen sind, können zum Beweis hierfür jeweilige Konstruktionen über die Automaten angewendet werden, siehe beispielsweise [68].

Kapitel 7.

Zustandsdynamische Zugriffskontrollpolitiken

Bisher betrachteten wir zu einem Zeitpunkt jeweils eine Schutzdomäne, welche durch genau eine Zugriffskontrollkomponente geschützt wird. Die zugehörige Domänenkonfiguration definiert die in dieser Schutzdomäne als erlaubt angesehenen Ausführungssequenzen, von denen genau eine in Anspruch genommen werden kann. Der Fokus dieser Arbeit liegt auf der Überwachung und Kontrolle verteilter Rechensysteme. Ein mögliches Szenario, in das wir das hier entwickelte Rahmenwerk zur Zugriffskontrolle einbetten wollen, sind virtuelle Organisationen, siehe Kapitel 1.1. Eine Zugriffskontrolle für Domänenkonfigurationen, wie sie im vorherigen Kapitel 6 entworfen wurde, ist zu einschränkend für solche komplexen Anwendungsszenarien. Wir gehen davon aus, dass eine virtuelle Organisation verschiedene Dienste unterschiedlicher Schutzdomänen umfasst. Für jede der eingebundenen Schutzdomänen sollen zugehörige Verwalter beliebige Domänenkonfigurationen definieren und durchsetzen können. Ein weiterer wichtiger Aspekt ist, dass für eine virtuellen Organisation viele anfragende Subjekte auftreten können, die möglicherweise parallel Zugriffswünsche an die Zugriffskontrolle richten.

Die Zugriffskontrolle einer virtuellen Organisation muss also sowohl mit mehreren, möglicherweise kombinierten Domänenkonfigurationen involvierter Schutzdomänen, als auch mit mehreren anfragenden Subjekten umgehen können. Im Hinblick auf die Zustandsabhängigkeit ist es wichtig, dass mehrere Instanzen der jeweiligen Domänenkonfigurationen parallel behandelt werden können.

Um diesen Anforderungen an eine adäquate Zugriffskontrolle gerecht zu werden, erweitern wir die im vorherigen Kapitel 6 entworfenen Domänenkonfigurationen um die folgenden beiden Konzepte:

1. Domänenkonfigurationen erhalten systemweit gültige *Identifikatoren*.
2. Eine *zustandsdynamische Zugriffskontrollpolitik* wird definiert als eine Menge identifizierter Domänenkonfigurationen.

Hierfür definieren wir $\mathcal{I}$ als die Menge systemweiter Identifikatoren.

Definition 7.1 (identifizierte Domänenkonfiguration). *$\mathcal{C}^{id}$ ist die Menge der mittels $id \in \mathcal{I}$ identifizierten Domänenkonfigurationen.*

Identifizierte Domänenkonfigurationen können zu einer systemweiten, zustandsdynamischen Zugriffskontrollpolitik zusammengefasst werden.

Definition 7.2 (zustandsdynamische Zugriffskontrollpolitik). *$\mathcal{P} \subset \wp(\mathcal{C}^{id})$ ist die Menge der zustandsdynamischen Zugriffskontrollpolitiken.*

7.1. Spezifizierung

Um eine zustandsdynamische Zugriffskontrollpolitik kompositional spezifizieren zu können, definieren wir auf dieser Ebene ebenfalls geeignete Kompositionsoperatoren. Die Operatoren entsprechen denen vorgestellt in Kapitel 6, allerdings angepasst für Potenzmengenkonstrukte. Hierfür muss beachtet werden, dass Kompositionsoperationen, welche die Konfigurationsstrukturen verändern, entsprechend neue Identifikatoren erzeugen.

- $\cup$ Der Vereinigungsoperator liefert die *mengentheoretische Vereinigung* zweier zustandsdynamischer Zugriffskontrollpolitiken.
- $\cap$ Der Durchschnittsoperator liefert den *mengentheoretischen Durchschnitt* zweier zustandsdynamischer Zugriffskontrollpolitiken.
- $\setminus$ Der Differenzoperator liefert die *mengentheoretische Differenz* zweier zustandsdynamischer Zugriffskontrollpolitiken.
- $\bullet$ Der *Konkatenationsoperator* verknüpft zwei zustandsdynamische Zugriffskontrollpolitiken, indem die Konkatenation auf den identifizierten Domänenkonfigurationen ausgeführt wird:

$$\mathcal{P}_1 \bullet \mathcal{P}_2 = \{\mathcal{C}^{con(id_1,id_2)} \mid \exists\ \mathcal{C}_1^{id_1} \in \mathcal{P}_1 \text{ und } \mathcal{C}_2^{id_2} \in \mathcal{P}_2, \text{ so dass } \mathcal{C} = \mathcal{C}_1 \bullet \mathcal{C}_2\}.$$

- $|||$ Der *Verschränkungsoperator* liefert die sequenzielle Verzahnung zweier zustandsdynamischen Zugriffskontrollpolitiken, indem die Verschränkung auf den identifizierten Konfigurationen ausgeführt wird:

$$\mathcal{P}_1|||\mathcal{P}_2 = \{\mathcal{C}^{int(id_1,id_2)} \mid \exists\ \mathcal{C}_1^{id_1} \in \mathcal{P}_1 \text{ und } \mathcal{C}_2^{id_2} \in \mathcal{P}_2, \text{ so dass } \mathcal{C} = \mathcal{C}_1|||\mathcal{C}_2\}.$$

- $*$ Der unäre *Kleene*-Stern liefert den *Kleeneschen Abschluss* einer übergebenen zustandsdynamischen Zugriffskontrollpolitik:

$$\mathcal{P}^* = \{\mathcal{C}^{kleene(id)} \mid \exists\ \mathcal{C}_1^{id} \in \mathcal{P}, \text{ so dass } \mathcal{C} = \mathcal{C}_1^*\}.$$

σ_{cond} Der unäre *Auswahloperator* liefert diejenigen identifizierten Konfigurationen einer übergebenen zustandsdynamischen Zugriffskontrollpolitik, auf denen der entsprechende Auswahloperator ausgeführt wurde:

$$\sigma_{cond}(\mathcal{P}) = \{\mathcal{C}^{cond(id)} \mid \exists\ \mathcal{C}_1^{id} \in \mathcal{P}, \text{ so dass } \mathcal{C} = \sigma_{cond}(\mathcal{C}_1)\}.$$

Während die oben definierten Kompositionsoperatoren angelehnt an Kompositionsoperatoren für Domänenkonfigurationen sind, benötigen wir weiterhin exklusive Operatoren für zustandsdynamische Zugriffskontrollpolitiken.

σ_{Cond} Die *Konfigurationsauswahl* liefert diejenigen identifizierten Konfigurationen einer übergebenen zustandsdynamischen Zugriffskontrollpolitik, die das Prädikat *Cond* erfüllen:

$$\sigma_{Cond}(\mathcal{P}) = \{\mathcal{C}^{id} \mid \mathcal{C}^{id} \in \mathcal{P} \text{ und } \mathcal{C} \text{ erfüllt } Cond\}.$$

Die Konfigurationsauswahl kann als *Duplikatfilter* eingesetzt werden. In diesem speziellen Fall wird *Cond* entsprechend definiert, so dass der Operator nur diejenigen identifizierten Konfigurationen zurückliefert, die mit jeweils keiner anderen in der zustandsdynamischen Zugriffskontrollpolitik enthaltenen identifizierten Konfigurationen übereinstimmen.

δ^n Der unäre *Duplikaterzeuger* liefert n Kopien der identifizierten Konfigurationen einer übergebenen zustandsdynamischen Zugriffskontrollpolitik:

$$\delta^n(\mathcal{P}) = \{\mathcal{C}^{\delta(id_1)}, \ldots, \mathcal{C}^{\delta(id_n)} \mid \mathcal{C}^{id} \in \mathcal{P}\}.$$

δ^* Der unäre *unbeschränkte Duplikaterzeuger* liefert unbeschränkt viele Kopien der identifizierten Konfigurationen einer übergebenen zustandsdynamischen Zugriffskontrollpolitik:

$$\delta^*(\mathcal{P}) = \{\mathcal{C}^{\delta^*(id)} \mid \mathcal{C}^{id} \in \mathcal{P}\}.$$

Beispiel 7.3 (Forschungsdatenbank). Betrachten wir die virtuelle Organisation ViBib, so bietet die Datenbank ihren Nutzern die Recherche in den Dokumentenbeständen, die die eingebundenen Organisationseinheiten hierfür zur Verfügung stellen. Wir gehen davon aus, dass jede der Organisationseinheiten eine lokale Domänenkonfiguration spezifiziert hat, die die Recherche in den Beständen kontrolliert, siehe hierzu das

Beispiel 6.6. Zur Identifizierung der lokalen Domänenkonfigurationen wird der Name der jeweiligen Organisationseinheit genutzt. Eine einfache zustandsdynamische Zugriffskontrollpolitik der Forschungsdatenbank kann dann als

$$\mathcal{P} \quad = \quad \{Recherche^{TUDoBib},\ Recherche^{RUBib},\ Recherche^{CityBib}\}$$

spezifiziert werden. ◊

Beispiel 7.4 (Domänenkonfiguration). Im Allgemeinen kann jede Domänenkonfiguration $\mathcal{C}$, wie sie in Kapitel 6 definiert wurde, durch einen Identifikator id ausgezeichnet werden, siehe Definition 7.1. Dann können eine Domänenkonfiguration $\mathcal{C}$ und eine zustandsdynamische Politik $\mathcal{P} = \{\mathcal{C}^{id}\}$ als äquivalent betrachtet werden. Dies bedeutet auch, dass die Mächtigkeit der Ausdrucksmöglichkeiten der Domänenkonfigurationen im Hinblick auf traditionelle Zugriffskontrollmodelle, vergleiche die hierzu die Beispiele „diskretionäre Zugriffskontrolle“ (6.7), „RBAC“ (6.8) und „Chinese-Wall Ansatz“ (6.9), auf zustandsdynamische Politiken übertragen werden kann. ◊

7.2. Deklarative Zugriffskontrolle

Als mögliches Anwendungsszenario der zustandsdynamischen Zugriffskontrolle betrachten wir ein verteiltes System, welches sich aus mehreren Schutzdomänen zusammensetzt. Wir gehen davon aus, dass für jede Schutzdomäne eine Domänenkonfiguration vorliegt, welche von einem Verwalter spezifiziert wurde. Eine Politik $\mathcal{P}$ fasst die identifizierten Domänenkonfigurationen zur systemweiten, zustandsdynamischen Zugriffskontrollpolitik zusammen. Zugriffswünsche von anfragenden Subjekten sollen bestimmten identifizierten Domänenkonfigurationen zugeordnet werden können, wobei die anfragenden Subjekte diese Zuordnung festlegen. Hierfür ermöglichen wir die *Verknüpfung* einer Anfrage mit einer identifizierten Domänenkonfiguration mit Hilfe entsprechender Identifikatoren.

Annahme 7.5. *Jede Anfrage (s, o, a) bezüglich einer zustandsdynamischen Zugriffskontrollpolitik $\mathcal{P}$ ist über eine Indizierung der Art $(id : s, o, a)$ mit einer identifizierten Domänenkonfiguration $\mathcal{C}_i^{id} \in \mathcal{P}$ verknüpft.*

Der Identifikator einer Anfrage wird dann als sogenannter *Sicherheitskontext* interpretiert. Dies bedeutet, dass eine identifizierte Anfrage aufgrund des Indizes einer identifizierten Domänenkonfiguration und somit der zugehörigen Schutzdomäne zugeordnet wird. Die Abbildung 7.1 veranschaulicht die Zuordnung einer Anfrage im Beispielszenario der Forschungsdatenbank. Mit der Spezifizierung einer globalen zustandsdynamischen Zugriffskontrollpolitik legt ein Verwalter implizit ein gewünschtes

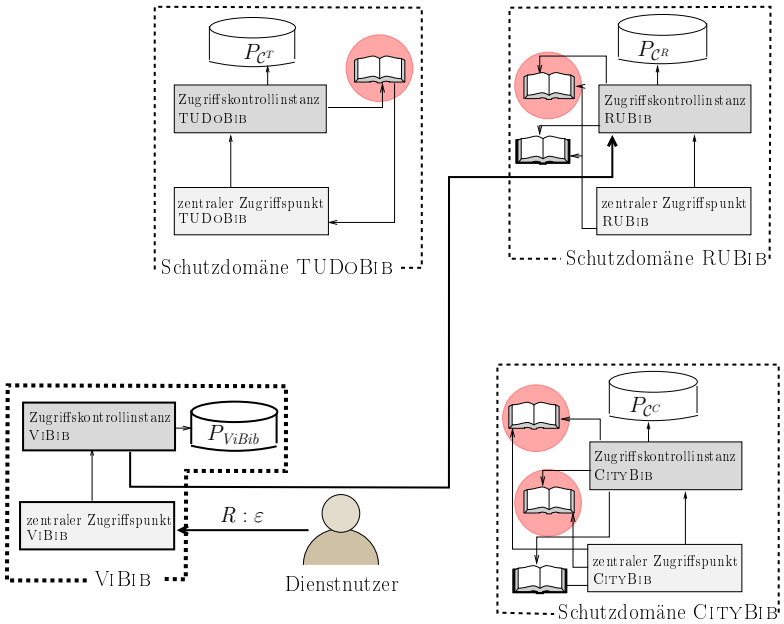


Abbildung 7.1.: Beispielszenario der Schutzdomänen unter Berücksichtigung des Sicherheitskontextes einer Anfrage.

Verhalten der (globalen) Zugriffskontrolle im Hinblick auf die Abarbeitung eingehender Anfragesequenzen fest.

Definition 7.6 (gewünschtes Verhalten der Zugriffskontrolle). *Ein Verwalter, der eine zustandsdynamische Zugriffskontrollpolitik $\mathcal{P}$, bestehend aus einer Menge identifizierter Domänenkonfigurationen $\mathcal{C}_i^{id_i}$ spezifiziert, legt das folgende Verhalten der globalen Zugriffskontrolle fest: Für jede identifizierte Anfragesequenz $(id_1 : s_1, o_1, a_1) \ldots (id_i : s_i, o_i, a_i) \ldots (id_n : s_n, o_n, a_n)$, für jeden Index id mit $\mathcal{C}^{id} \in \mathcal{P}$, betrachten wir die Teilsequenzen aller identifizierten Anfragen, die an den Identifikator id gebunden sind. Die erlaubten Teilsequenzen dieser identifizierten Anfrageteilsequenzen sind jeweils Präfix mindestens einer erlaubten Ausführungssequenz der Domänenkonfiguration $\mathcal{C}^{id}$. Alle übrigen Anfrageteilsequenzen werden abgelehnt.*

Basierend auf dem gewünschten Verhalten der Zugriffskontrolle einer Domänenkonfiguration, siehe Abschnitt 6.2, erweitern wir hier die Zugriffskontrolle entsprechend. Aufgrund der Identifizierung der Domänenkonfigurationen und Anfragen benötigt ein Zugriffskontrollmechanismus die Möglichkeit, sich auf den jeweiligen Sicherheitskontext einzustellen. Jede Domänenkonfiguration wird innerhalb ihres Sicherheitskontextes kontrolliert und überwacht. Für die Zuordnung einer identifizierten Anfrage zu dem jeweiligen Sicherheitskontext wird eine globale Kontrollkomponente benötigt, die alle eingehenden identifizierten Anfragen abfängt und entsprechend weiterleitet.

Definition 7.7 (Verhalten der Zugriffskontrolle).

Eingabe: $\mathcal{P}$ *[vorliegende] zustandsdynamische Politik*
$(id : \varepsilon)$ *identifizierte Anfrage*

Ausgabe: d *Boolean [Zugriffsentscheidung]*
$\mathcal{Q}$ *[aktualisierte] zustandsdynamische Politik*

Methode:
if id Sicherheitskontext in $\mathcal{P}$
then
$d :=$ *entscheide*$(\mathcal{C}^{id}, \varepsilon)$;
$\mathcal{D}^{id} :=$ *if* d **then** *aktualisiere*$(\mathcal{C}^{id}, \varepsilon)$
else $\mathcal{C}^{id}$;
$\mathcal{Q} :=$ *ersetze*$(\mathcal{C}^{id}$ *durch* $\mathcal{D}^{id}$ *in* $\mathcal{P})$
else
$d := false$;
$\mathcal{Q} := \mathcal{P}$;

Aus Theorem 6.14 ergibt sich das folgende Korollar.

Korollar 7.8. *Wird eine identifizierte Anfrage gemäß Definition 7.7 behandelt, so wird das gewünschte Verhalten der Zugriffskontrolle gemäß Definition 7.6 gewährleistet.*

7.3. Eine Operationalisierung der Zugriffskontrolle

Wie schon in Abschnitt 1.2 besprochen, muss eine Zugriffskontrolle effizient und berechenbar operationalisiert werden können. Hierfür müssen die Spezifizierungen von Zugriffskontrollpolitiken und Berechnungen auf diesen geeignet algorithmisiert werden. In Abschnitt 6.3 haben wir einen Vorschlag der Operationalisierung erarbeitet, der auf Domänenkonfigurationen basiert, die gemäß regulärer Ausdrücke spezifiziert werden. In diesem Fall lassen sich zugehörige Zugriffsentscheidungen auf entsprechende akzeptierende Automaten beziehungsweise auf entsprechende Konstruktionen von diesen abbilden.

Die Spezifikation zustandsdynamischer Politiken basiert grundlegend auf der Spezifikation von Domänenkonfigurationen. Aufgrund dessen lässt sich die Operationalisierung der Domänenkonfigurationen analog auf eine mögliche Operationalisierung der zustandsdynamischen Politiken übertragen. Die hier in Abschnitt 7.1 vorgestellten Kompositionsoperatoren für zustandsdynamische Zugriffskontrollpolitiken sind zum größten Teil analog erweiterte Kompositionsoperatoren für Domänenkonfigurationen, wie sie in Abschnitt 6.1 eingeführt wurden. Aus diesem Grund können sie ebenso analog operationalisiert werden, wie wir im Folgenden als Beispiel anführen.

Beispiel 7.9 (Forschungsdatenbank). Wir betrachten die möglichen Zugriffe aus dem Beispiel 6.5 als und definieren die Domänenkonfiguration

$$Kauf \quad = \quad \{\alpha\}^* \bullet (\{\beta\gamma\varepsilon\} \cup \{\beta\delta\varepsilon\}).$$

Zur Identifizierung der Domänenkonfigurationen benutzen wir die Anfangsbuchstaben der Namen der eingebundenen Organisationseinheiten TUDoBib und CityBib, so dass wir mit den Konfigurationen $Kauf^T$ und $Kauf^C$ eine einfache zustandsdynamische Politik spezifizieren können:

$$\mathcal{P} \quad = \quad \{Kauf^T, Kauf^C\}. \qquad \Diamond$$

Da die Identifizierung von Domänenkonfigurationen nur auf syntaktischer Ebene stattfindet, können die den Konfigurationen zugehörigen, akzeptierenden Automaten $\mathcal{A}_{Kauf^T}$ und $\mathcal{A}_{Kauf^C}$ über dem Alphabet $\Sigma = \{\alpha, \beta, \gamma, \delta, \varepsilon\}$ analog zu einem akzeptierenden Automaten $\mathcal{A}_{Kauf}$ beschrieben werden.

Ein Automat $\mathcal{A}_P$, der die Politik P durchsetzt, ist ein Vereinigungsautomat von $\mathcal{A}_{Kauf^T}$ und $\mathcal{A}_{Kauf^C}$. Die Abbildung 7.2 visualisiert einen solchen. Zur Zuordnung der Zustände der jeweiligen Automaten wurden in der Abbildung die Identifikatoren der zugehörigen

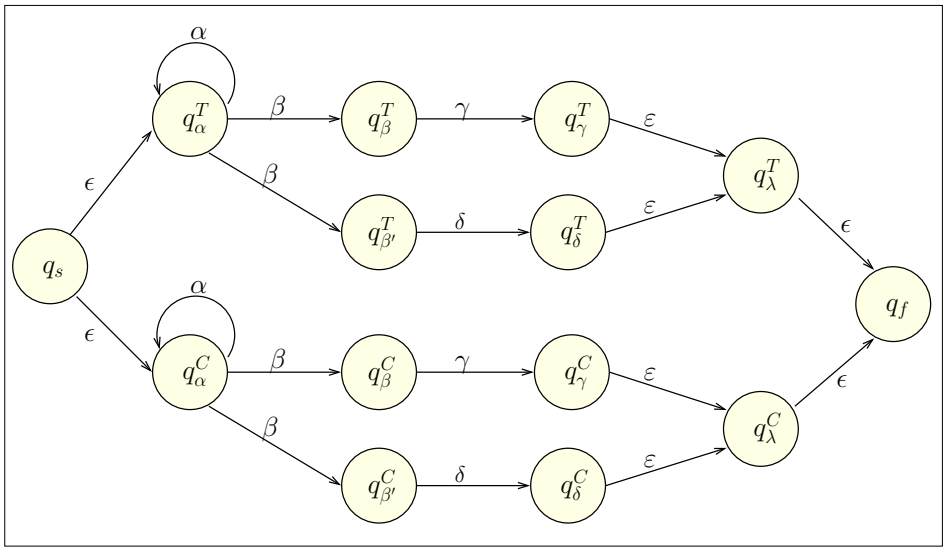


Abbildung 7.2.: Beispielautomat $\mathcal{A}_P$

Konfigurationen verwendet. Die Zustände q_s und q_f sind neue, aus der Vereinigung resultierende Zustände, die gemäß der Konstruktion von Vereinigungsautomaten, siehe bspw. [68], mit entsprechenden ϵ-Übergängen eingebunden werden.

Neben den Kompositionsoperatoren, die sich von der Operationalisierung der Domänenkonfigurationen generisch auf die Operationalisierung der zustandsdynamischen Politiken übertragen lassen, müssen noch die Operatoren betrachtet werden, welche neue Identifikatoren erzeugen, Duplikatfilter und Duplikaterzeuger.

- Laut Definition der Erzeugung von Identifikatoren, findet diese nur auf der syntaktischen Ebene der Namensgebung neuer Domänenkonfigurationen statt. Deswegen beeinflusst dieser Operator nicht die Operationalisierung der Zugriffskontrollpolitiken mittels regulärer Sprachen.

- Der Einsatz des Duplikatfilters bedeutet, dass zwei Domänenkonfigurationen auf Gleichheit überprüft werden können müssen. In Betrachtung der Operationalisierung von Domänenkonfiguration bedeutet dies, dass für die entsprechenden Grammatiken entschieden werden muss, ob diese dieselbe Sprache erzeugen. Da wir uns (bisher) im Bereich der regulären Sprachen bewegen, ist dieses Problem entscheidbar.

- Die Duplikaterzeuger werden im Hinblick auf eine Operationalisierung analog zu dem Operator *Kleene*-Stern behandelt.

Basierend auf der Annahme, dass Domänenkonfigurationen durch reguläre Sprachen operationalisiert werden, ist die zustandsdynamische Zugriffskontrollpolitik ebenfalls effizient berechenbar.

Theorem 7.10 (Berechenbarkeit zustandsdynamischer Politiken). *Es ist vorausgesetzt, dass alle durch in diesem Kapitel 7 vorgestellten Kompositionsoperatoren spezifizierten zustandsdynamischen Zugriffskontrollpolitiken beziehungsweise durch in Kapitel 6 vorgestellten Kompositionsoperatoren spezifizierten Domänenkonfigurationen durch reguläre Sprachen repräsentiert werden und alle Prädikate auf Domänenkonfigurationen entscheidbar sind. Dann sind die so spezifizierten Zugriffskontrollpolitiken, ebenso wie beinhaltete Domänenkonfigurationen algorithmisch berechenbar, indem das Verhalten einer Zugriffskontrolle nach Definition 7.6 durch die Bearbeitung der entsprechenden Anfragen nach Definition 7.7 sicher gestellt wird.*

Beweis. Die Klasse der regulären Sprachen ist abgeschlossen unter allen Kompositionsoperatoren, welche zur Spezifizierung zustandsabhängiger Zugriffskontrollpolitiken beziehungsweise zur Spezifizierung von Domänenkonfigurationen verwendet werden. Zusammen mit Korollar 7.8 beweist dies das obige Theorem. □

Teil IV.

Durchsetzung

In diesem Teil werden Zertifikat-basierte Realisierungsmöglichkeiten zustandsdynamischer Politiken erarbeitet. Ausgehend von einer zentralen Lösung werden zwei Ansätze dezentraler Realisierungen vorgeschlagen, welche die Konzepte der in Teil II vorgestellten aufgreifen und für zustandsdynamische Politiken anpassen.

Kapitel 8.

Architekturentwürfe für zustandsdynamische Zugriffskontrollpolitiken

In diesem Kapitel entwerfen wir eine zentrale Architektur für zustandsdynamische Zugriffskontrollpolitiken. Ausgehend von dieser erarbeiten wir dann mehrere Varianten dezentraler Architekturen. Danach analysieren wir deren Stärken und Schwachstellen. Wie sich herausstellen wird, ist eine durchgängig dezentrale Lösung nur unter Einbußen hinsichtlich verschiedener Sicherheitsinteressen möglich, so dass eine Abwägung zwischen Dezentralisierung und Erfüllung von Sicherheitsinteressen von Nöten ist.

8.1. Zentrale Realisierung zustandsdynamischer Politiken

Die Konstruktion der Spezifizierung der zustandsdynamischen Zugriffskontrollpolitiken ist bewusst auf einen regulären Kern beschränkt. Die Grundidee der Durchsetzung und Kontrolle der zustandsdynamischen Zugriffskontrollpolitiken ist somit, dass die Spezifikationen durch geeignete Automaten überwacht werden können. Im einfachen, zentralen Szenario mit nur einer Domänenkonfiguration kann ein zentraler Referenzmonitor diese Überwachung übernehmen. Dieser Referenzmonitor fängt alle Zugriffswünsche ab und behandelt diese wie in Abbildung 8.1 visualisiert. Die in der Abbildung gegebene Konfiguration C wird dann als entsprechender Automat repräsentiert. Wie in Abschnitt 2.3 für verteilte Systeme beschrieben, wird das Referenzmonitorprinzip in diesem zentralen Architekturentwurf durch eine Zugriffskontroll-Entscheidungskomponente und eine Zugriffskontroll-Durchsetzungskomponente realisiert. Die Durchsetzungskomponente fängt eingehende Zugriffswünsche ab und leitet diese an die Entscheidungskomponente weiter. Diese implementiert zum einen das in Definition 6.11 definierte *entscheide*-Prädikat und zum anderen die in Definition 6.12 definierte Aktualisierung einer Domänenkonfiguration. Für

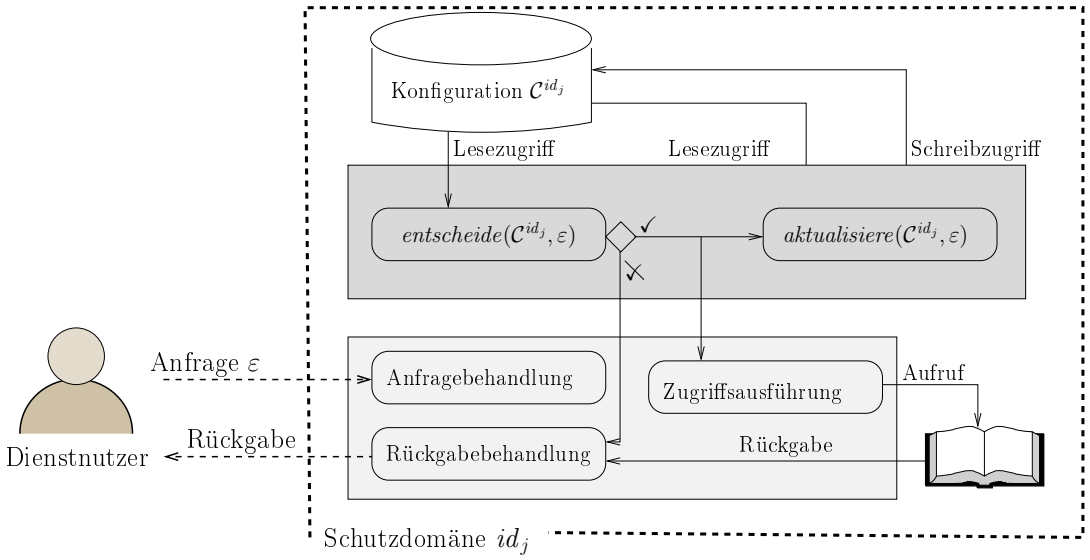


Abbildung 8.1.: Eine zentrale Architektur für die Realisierung zustandsdynamischer Politiken.

beide Aktivitäten wird der Lesezugriff auf die aktuelle Repräsentation der Domänenkonfiguration benötigt. Die Erlaubnis des Zugriffswunsches ε eines anfragenden Dienstnutzers wird in der Realisierung dadurch gegeben, dass die Ausführung der durch ε bezeichneten elementaren Erlaubnis im aktuellen Zustand des die Domänenkonfiguration repräsentierenden Automaten ein erlaubter Zustandsübergang ist. Die Entscheidungskomponente überführt dann den die Domänenkonfiguration repräsentierenden Automaten in den entsprechenden Folgezustand. Hierfür benötigt die Komponente einen Schreibzugriff auf die Repräsentation der Domänenkonfiguration. Bei einem erlaubten Zugriffswunsch wird die gewünschte Funktionalität des Dienstes ausgeführt. Eventuelle Rückgaben der Ausführung werden über die Durchsetzungskomponente an den anfragenden Dienstnutzer weitergeleitet.

Beispiel 8.1 (Forschungsdatenbank). Wir betrachten als Beispiel die Domänenkonfigurationen

$$C \quad = \quad \{\alpha\}^* \bullet \{\beta\} \bullet (\{\gamma\varepsilon\} \cup \{\delta\varepsilon\}).$$

Ein akzeptierender Automat $\mathcal{A}_C$ über $\Sigma_C = \{\alpha, \beta, \gamma, \delta, \varepsilon\}$ kann beschrieben werden durch

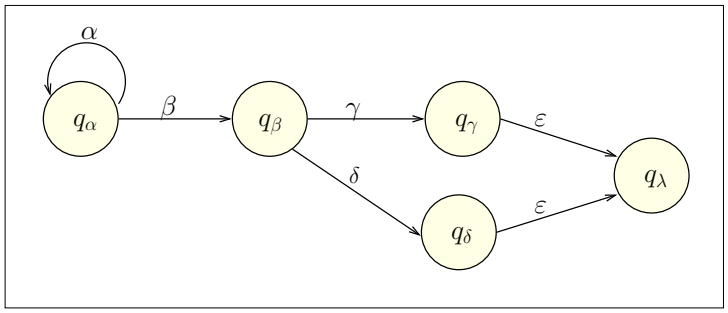


Abbildung 8.2.: Beispielautomat $\mathcal{A}_C$

$$
\begin{array}{lcl}
Z & = & \{q_\alpha, q_\beta, q_\gamma, q_\delta, q_\varepsilon, q_\lambda\}, \\
\delta & = & \{(q_\alpha, \alpha, q_\alpha), (q_\alpha, \beta, q_\beta), (q_\beta, \gamma, q_\gamma), (q_\beta, \delta, q_\delta), (q_\gamma, \varepsilon, q_\lambda), (q_\delta, \varepsilon, q_\lambda)\}, \\
s & = & \{q_\alpha\}, \\
E & = & \{q_\lambda\}.
\end{array}
$$

Die Abbildung 8.2 zeigt eine Darstellung des Automaten $\mathcal{A}_C$, der zur Operationalisierung der Zugriffskontrolle der in Beispiel 8.1 gegebenen Konfiguration C dient.

Wir betrachten eine Anfragesequenz $\alpha\beta\alpha$ und sei $\mathcal{C}$ die Domänenkonfiguration aus Beispiel 8.1. Im Folgenden spielen wir das Verhalten der Zugriffskontrolle für die Domänenkonfiguration durch, wie es in Definition 6.13 festgelegt wurde. Im ersten Schritt fängt die Anfragebehandlung die Anfrage α ab und übermittelt diese der Entscheidungskomponente. Die Auswertung von $\mathit{entscheide}(\mathcal{C}, \alpha)$ liefert *true*, da eine Ausführungssequenz $t \in \mathcal{C}$ existiert, für die $\mathit{kopf}(t) = \alpha$ gilt. Diese positive Überprüfung stößt das Domänenkonfigurationsupdate $\mathit{aktualisiere}(\mathcal{C}, \alpha)$ an, welches die aktualisierte Domänenspezifikation

$$\mathcal{C}' \quad = \quad \{\alpha\}^* \bullet \{\beta\} \bullet (\{\gamma\varepsilon\} \cup \{\delta\varepsilon\})$$

erzeugt. Analog wird im zweiten Schritt die Anfrage β bearbeitet. Die Entscheidungskomponente $\mathit{entscheide}(\mathcal{C}', \beta)$ liefert *true*, da eine Anfragesequenz $t \in \mathcal{C}'$ existiert, für die $\mathit{kopf}(t) = \beta$ gilt. Das anschließende Konfigurationsupdate erzeugt die Konfigurationsspezifikation

$$\mathcal{C}'' \quad = \quad \{\gamma\varepsilon\} \cup \{\delta\varepsilon\}.$$

Der dritte Schritt bearbeitet die Anfrage α, welche aufgrund der aktuellen

Spezifikation der Domänenkonfiguration abgelehnt wird, da die Entscheidungskomponente $entscheide(\mathcal{C}'', \alpha)$ keine Ausführungssequenz $t \in \mathcal{C}''$ findet, für die $kopf(t) = \alpha$ liefert.

Übertragen auf die Realisierung mittels des repräsentierenden Automaten bedeutet dies, dass die Entscheidungskomponente eine Zustandskontrolle übernehmen muss. Sie entscheidet für eine Anfrage ausgehend vom aktuellen Zustand des Automaten, ob die Überführungsfunktion in einen korrekten Anschlusszustand überführt. Für den positiven Fall wird durch das Konfigurationsupdate der entsprechende Zustandsübergang durchgeführt. ◊

Die Abbildung 8.3 visualisiert die drei Schritte des Beispiels. Zur anschaulichen Darstellung wird in der Visualisierung die Information über den jeweilig aktuellen Zustand q_c durch eine Kreismarkierung und eine entsprechende Benennung der Zustände dargestellt. Zu beachten ist, dass die Zustände des Automaten tatsächlich nicht umbenannt werden.

8.2. Einbindung von Rollenkonzepten

In Abschnitt 1.2 wurde das Szenario festgelegt und Anforderungen an eine adäquate Zugriffskontrollpolitik gestellt. Mit Anforderung 3 forderten wir die Einsatzmöglichkeit von Rollenkonzepten, um die große Anzahl unbekannter Dienstnutzer verwalten zu können. Die Realisierung der kompositionalen Zugriffskontrolle, wie sie in Kapitel 4 skizziert wurde, ermöglicht die Benennung von Dienstnutzern durch ihre freien Merkmale. Der für die Realisierung der Attributzertifikate eingesetzte SDSI-Teil der SPKI/SDSI Infrastruktur ermöglicht eine rollenbasierte Zugriffskontrolle. Dieser Aspekt lässt sich auf den in dieser Arbeit vorgestellten Ansatz zustandsdynamischer Zugriffskontrolle insofern übertragen, als das die in elementaren Erlaubnissen vorkommenden Subjekte durch freie Merkmale anstatt durch identifizierte Subjekte benannt werden. Diese freien Merkmale fungieren dann als *Variablen*, die vor einer tatsächlichen Zugriffsentscheidung an anfragende Teilnehmer gebunden werden müssen. Diese Bindung geschieht jeweils spontan im Rahmen einer Bearbeitung eines Zugriffswunsches, indem der jeweilig anfragende Teilnehmer ihm ausgestellte Attributzertifikate vorlegt.

Der anfragende Teilnehmer ist in dieser Realisierung selbst für die Vorlage der entsprechenden Attributzertifikate verantwortlich. Die Überprüfung, ob vorgelegte Attributzertifikate dem anfragenden Teilnehmer das geforderte freie Merkmal zertifizieren, ist für die SPKI/SDSI Infrastruktur als Namensauflösung bekannt. Die von uns entwickelte Auswertung für algebraische Subjekte, ist in Kapitel 4 vorgestellt worden, siehe hierzu auch [4]. Gehen wir davon aus, dass eine elementare Erlaubnis

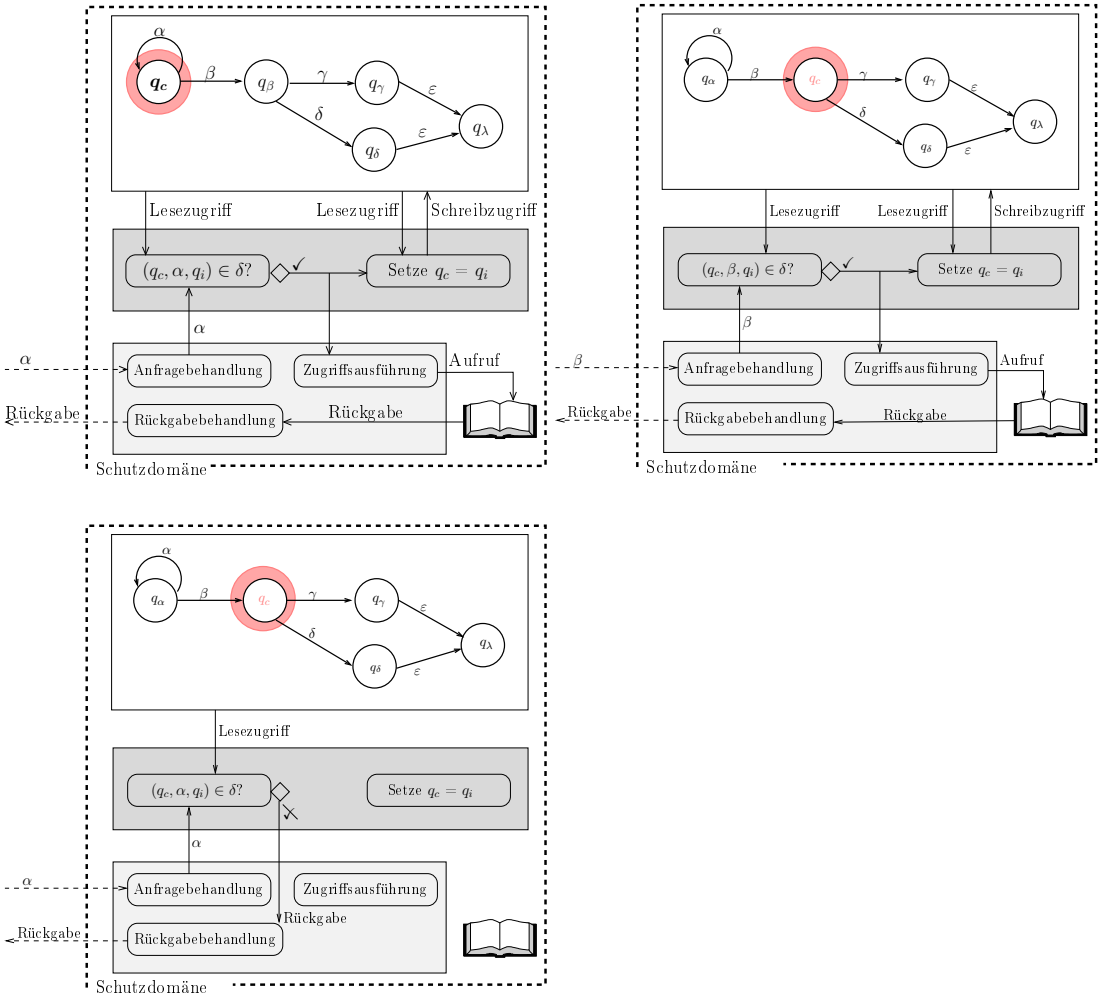


Abbildung 8.3.: Ablauf der Zugriffskontrolle für das Beispiel 8.1.

(ζ, o, a) so spezifiziert wird, dass das algebraische Subjekt ζ, im Sinne von [4, 66], als ein freies Merkmal fungiert. Dann muss die Entscheidungskomponente der Zugriffskontrolle die entsprechende Namensauflösung durchführen, um zu verifizieren, ob der anfragende Dienstnutzer das geforderte freie Merkmal ζ hat. Hierfür sammelt jeder Teilnehmer t die ihm ausgestellten Attributszertifikate $\mathcal{NC}_t$. Analog zur in Kapitel 4 vorgestellten Auswertung algebraischer Subjekte, kann durch die Auswertung $\mathcal{V}_{\mathcal{NC}_t}(\zeta)$ ermittelt werden, ob der Teilnehmer t beziehungsweise der durch seinen öffentlichen Schlüssel $\mathcal{K}_t$ repräsentierte Teilnehmer das geforderte freie Merkmal s besitzt, also ob $\mathcal{K}_t \in \mathcal{V}_{\mathcal{NC}_t}(\zeta)$. Die Operationalisierung des Ansatzes zustandsdynamischer Politiken bleibt hierdurch

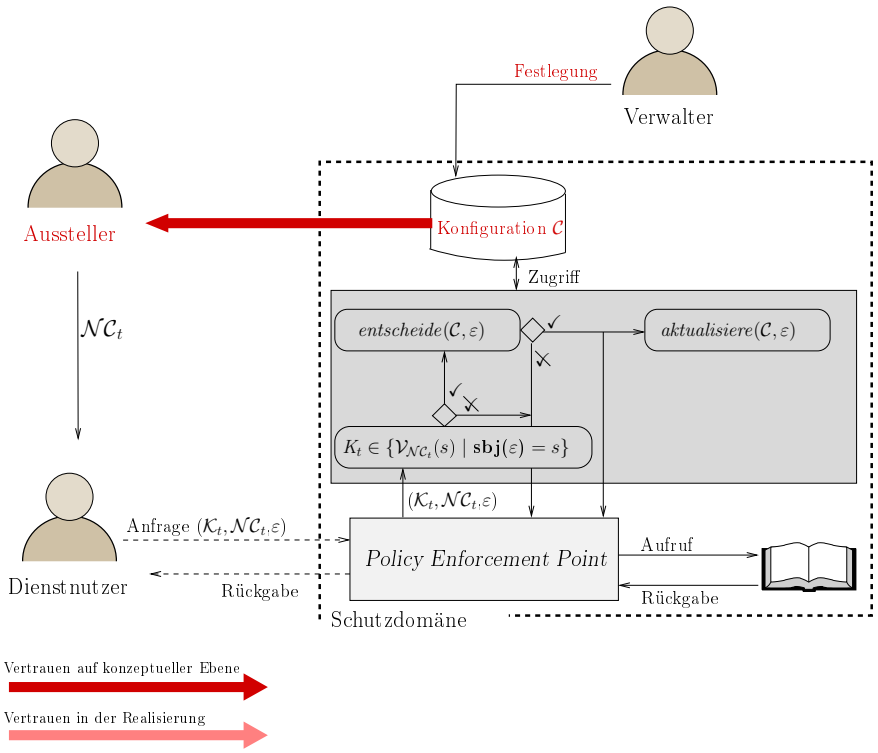


Abbildung 8.4.: Vertrauensstrukturen im zentralen Architekturmodell für die Realisierung von Rollenkonzepten.

unverändert.

In der Abbildung 8.4 wird veranschaulicht, wie der Teil der freien Merkmale des hybriden Modells in der Realisierung unseres Ansatzes eingesetzt werden kann und wie somit die Vertrauensstrukturen materialisiert werden.

8.3. Vom zentralen Entwurf zu dezentralen Architekturentwürfen

Wird der in Abschnitt 8.1 vorgestellte Ansatz einer zentral durchgesetzten Zugriffskontrolle so realisiert, so ergeben sich für den Einsatz in einer virtuellen Organisation zwei Hauptschwierigkeiten:

1. Die teilnehmenden Organisationseinheiten wollen autonom und lokal ihre Zugriffskontrollpolitiken spezifizieren. Wird ein zentrales *Repositorium* zur Speicherung der kombinierten Zugriffskontrollpolitik verwendet, so können die teilnehmenden Organisationseinheiten ihre lokalen Zugriffskontrollpolitiken nicht ohne weiteres dynamisch halten. Jede Änderung an einer lokalen Politik zieht eine Änderung an der globalen, kombinierten Politik nach sich. In einem solchen Fall muss es also Mechanismen geben, die die lokal spezifizierten Politiken an das zentrale Repositorium kommunizieren.

2. Die zentrale Zugriffskontrolle behandelt alle eingehenden Zugriffswünsche der teilnehmenden Dienstnutzer. Hierdurch wird der zentrale Zugriffspunkt schnell zu einem Engpass des Systems. Dies kann zum einen durch Warteschlangenverzögerungen zu einem Kommunikationsstau führen. Fällt zum anderen die zentrale Komponente (zeitweise) aus, so ist die virtuelle Organisation für diesen Zeitraum nicht nutzbar.

Im Hinblick auf den Einsatz der Zugriffskontrolle in einem verteilten Rechensystem werden wir in diesem Abschnitt einen Vorschlag für eine verteilte Architektur erarbeiten. Hierfür gehen wir von der im obigen Abschnitt 8.1 vorgestellten zentralen Architektur aus und betrachten, welche Komponenten unter welchen Einschränkungen oder mit welchen Vorteilen dezentralisiert werden können. Ein wichtiger Aspekt ist der für zustandsdynamische Zugriffskontrollpolitiken notwendige Sicherheitskontext, der in Abschnitt 7.2 eingeführt wurde. Die Abbildung 8.5 veranschaulicht ein Szenario, in dem für eingehende Anfragen der Sicherheitskontext berücksichtigt wird. Wie in der Abbildung erkennbar, besteht die Zugriffskontrolle auch im verteilten Szenario aus drei Hauptkomponenten: Die Speicherung der Domänenkonfiguration, die Entscheidungskomponente und das Domänenkonfigurationsupdate.

8.3.1. Speicherung der Domänenkonfiguration

Die aktuelle Repräsentation der Domänenkonfiguration ist unabdingbar für die Zugriffsentscheidung. Insofern muss jeder Teilnehmer, der über Zugriffswünsche entscheidet, Zugriff auf diese haben. Die relevanten Parameter betreffend für die Konfiguration sind die *Anzahl der Kopien*, die im Umlauf sind, und der jeweilige *Speicherort* der Konfiguration(en). Die Abbildung 8.6 veranschaulicht die möglichen Einstellungen der Konfigurationsparameter. Sofern wir von einer globalen Kopie der Konfigurationsrepräsentation ausgehen, kann diese in einem zentralen Repositorium gespeichert sein. Im Fall einer zentralen Zugriffsentscheidung verbleibt diese Kopie *statisch* in dem Repositorium. Für den Fall lokaler Zugriffsentscheidungen kann die Konfigurationsrepräsentation *beweglich* an die jeweiligen lokalen zuständigen Teilnehmer der virtuellen Organisation weitergegeben werden. Für die Weitergabe existieren zwei Alternativen:

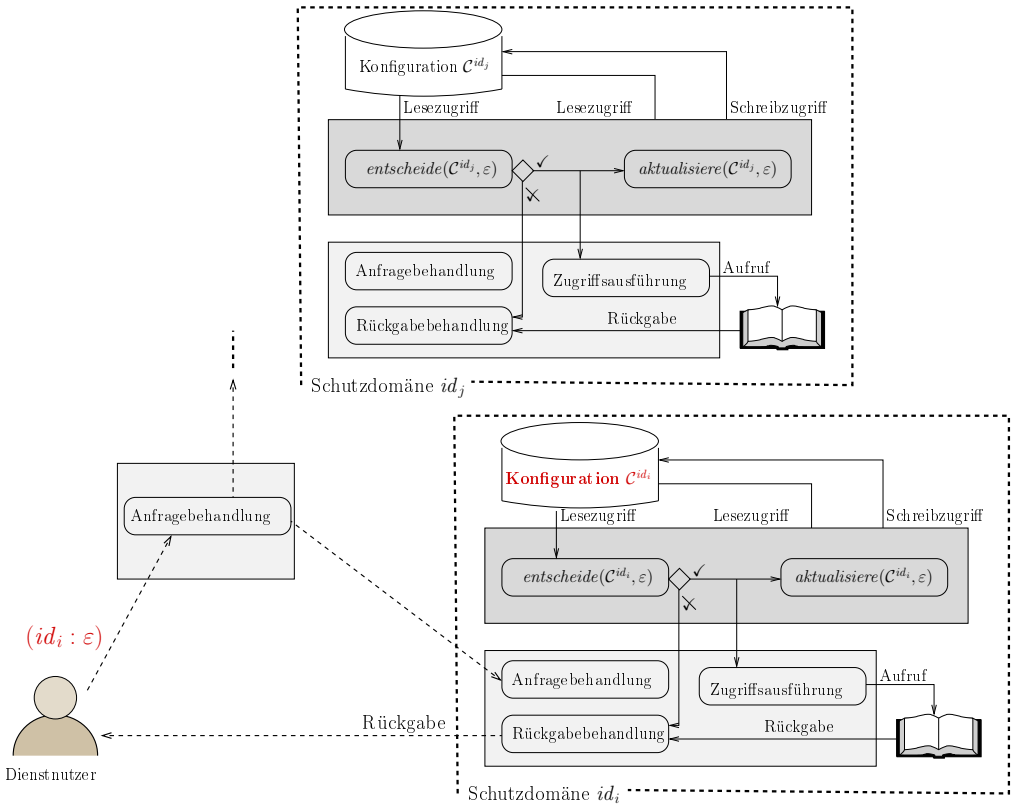


Abbildung 8.5.: Eine verteilte Architektur unter Berücksichtigung des Sicherheitskontextes.

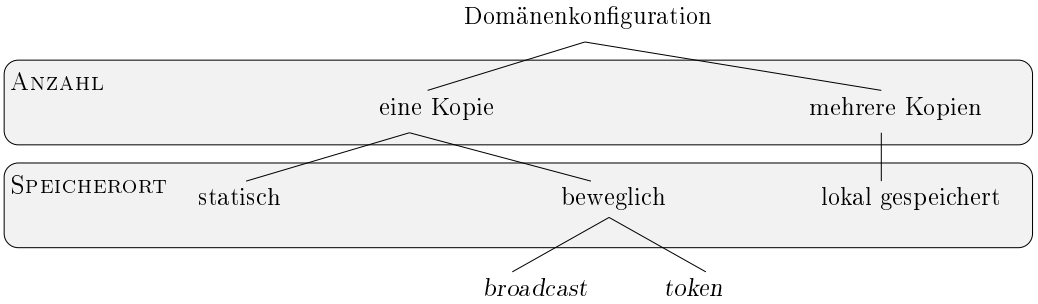


Abbildung 8.6.: Mögliche Parametereinstellungen der Konfigurationsspeicherung.

Broadcast oder *Token*-basiert. Im ersten Punkt wird die globale Konfigurationsrepräsentation sternförmig vom zentralen Repositorium an die Teilnehmer der virtuellen Organisation verteilt. Im zweiten Punkt wird die globale Konfigurationsrepräsentation von einem Teilnehmer zum anderen, ähnlich einem *token*, weitergereicht.

Soll die Konfigurationsrepräsentation an mehrere Teilnehmer verteilt werden, so kann jeder involvierte Teilnehmer der virtuellen Organisation eine Replikation lokal speichern. Für diese Variante der Speicherung treten die Schwierigkeiten der korrekten Synchronisierung auf. Es müssen in diesem Fall geeignete Mechanismen eingesetzt werden, damit jeder Teilnehmer, der über einen Zugriffswunsch entscheidet, auf eine aktuelle Kopie der Konfigurationsrepräsentation zugreifen kann. Nach erfolgreicher Aktualisierung dieser muss sie wiederum an die anderen Teilnehmer kommuniziert werden.

8.3.2. Die Entscheidungskomponente

Als *Speicherort* der Entscheidungskomponente existieren zwei Alternativen. Diese Komponente kann *zentral* gespeichert werden, so dass alle Teilnehmer, die über einen Zugriffswunsch entscheiden müssen, Zugriff auf die Komponente haben. Für die Akzeptanz unter den Teilnehmern muss für die Speicherung eine von allen als vertrauenswürdig anerkannte Stelle ausgewählt werden. Alternativ speichern die Teilnehmer *lokal*, also innerhalb ihrer zugehörigen Schutzdomänen eine Kopie der Komponente. Dann kann lokal über Zugriffswünsche entschieden werden. Wie in Abbildung 8.7 zu erkennen, werden einige Kombinationsmöglichkeiten der Parametereinstellungen bereits verworfen (X). Die zentrale Speicherung der Entscheidungskomponente ist nur dann sinnvoll, wenn auch die Repräsentation der Domänenkonfiguration zentral gespeichert wird. In diesem Fall kann das genutzte Repositorium auch für die

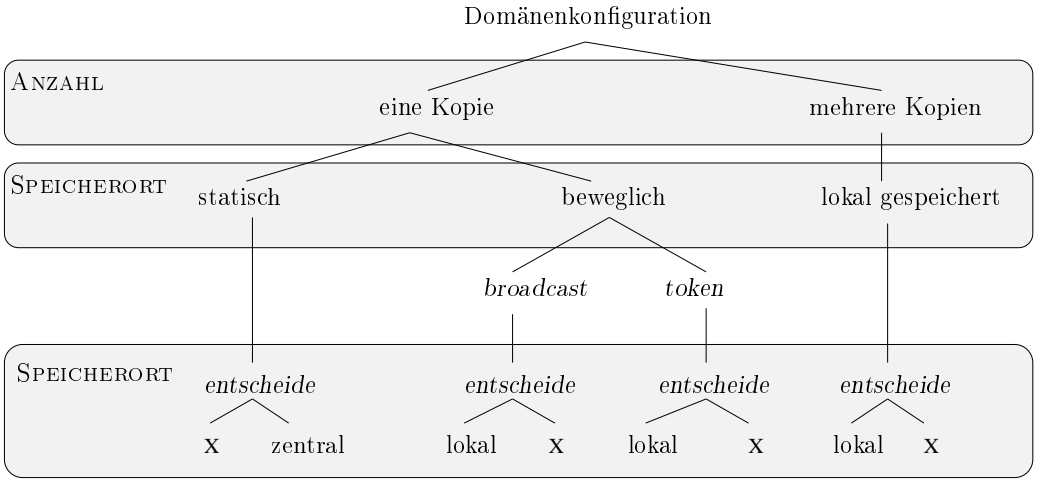


Abbildung 8.7.: Mögliche Parametereinstellungen der Konfigurationsspeicherung und der Entscheidungskomponente.

Speicherung der Entscheidungskomponente genutzt werden. Wird die Konfigurationsrepräsentation oder werden Kopien dieser an die Teilnehmer weitergegeben, sollte zur Reduzierung des Kommunikationsaufwandes auch die Entscheidungskomponente lokal gespeichert sein.

8.3.3. Das Domänenkonfigurationsupdate

Analog zu den im obigen Abschnitt 8.3.2 beschriebenen Speichermöglichkeiten der Entscheidungskomponente, kann auch die Komponente des Domänenkonfigurationsupdates *zentral* oder *lokal* gespeichert werden. Auch für diese Komponente sind nur einige Parametereinstellungen sinnvoll in Abhängigkeit von den zuvor ausgewählten Varianten. Eine zentrale Speicherung ist immer dann sinnvoll, wenn ohnehin ein zentrales Repositorium genutzt wird. Dies tritt ein, wenn die Konfigurationsrepräsentation und die Entscheidungskomponente bereits in einem zentralen Repositorium gespeichert sind, aber auch wenn nur die Konfigurationsrepräsentation zentral gespeichert ist und mittels *broadcast* kommuniziert wird. In den übrigen Fällen sollte auch die Komponente des Domänenkonfigurationsupdate lokal in den Schutzdomänen der Teilnehmer gespeichert sein. Die Abbildung 8.8 veranschaulicht die möglichen Einstellungen der Parameter.

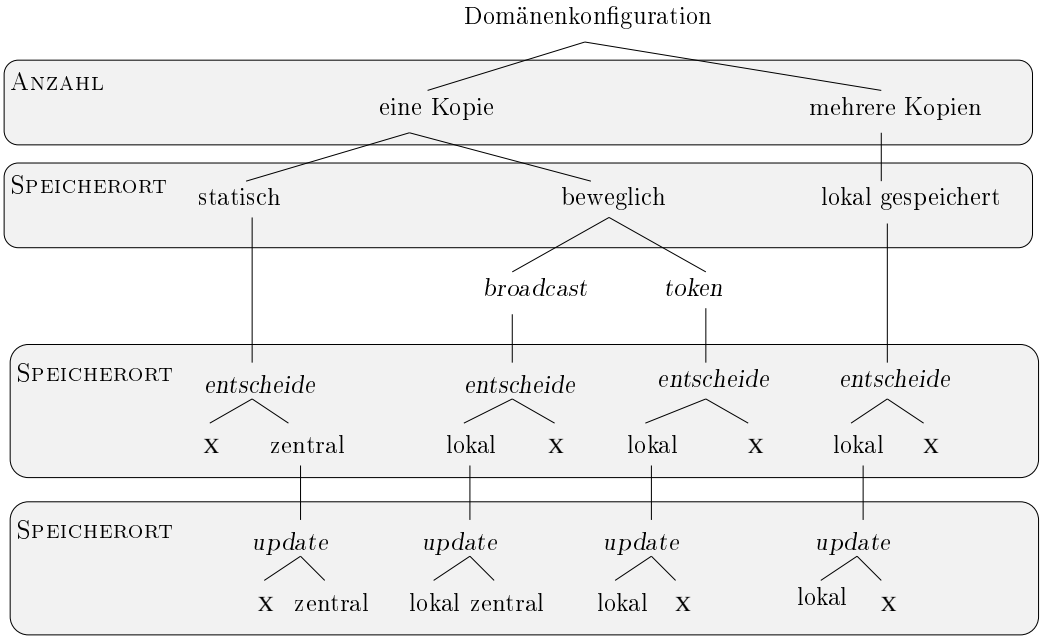


Abbildung 8.8.: Mögliche Parametereinstellungen im Gesamtüberblick.

8.4. Bewertung der dezentralen Architekturentwürfe

In Abschnitt 8.3 haben wir verschiedene Alternativen aufgezeigt, wie die wichtigen Komponenten der Zugriffskontrolle geeignet dezentralisiert werden können. In Abbildung 8.8 ist zu sehen, dass fünf verschiedene Parametereinstellungen sinnvoll für den verteilten Architekturentwurf diskutiert werden können. Ein gewichtiger Einflussfaktor in dezentralen Architekturen ist der zusätzliche Nachrichtenverkehr, der sich im Vergleich zu zentralen Lösungen erhöht. Die folgenden Nachrichtentypen treten bereits in einem zentralen Architekturentwurf für zustandsdynamische Politiken auf, siehe Abbildung 8.1 zur Erinnerung.

1. *Anfrage*nachricht
   Die Anfragenachricht wird von einem potenziellen Dienstnutzer an einen Zugriffspunkt gesendet.

2. *Rückgabe*nachricht
   Die Rückgabenachricht wird von einer Durchsetzungskomponente an den anfragenden Dienstnutzer gesendet.

3. *Aufruf*nachricht
   Nach erfolgreicher Überprüfung des Zugriffswunsches ruft die Durchsetzungskomponente die gewünschte Funktionalität des Dienstes auf.

Betrachten wir eine verteilte Architektur, so können die folgenden Nachrichtentypen hinzukommen.

4. *Status*anfrage
   Sofern die Konfigurationsrepräsentation zentral gespeichert wird, aber die Teilnehmer lokal über Zugriffswünsche entscheiden, müssen am zentralen Repositorium Statusanfragen über den aktuellen Zustand der Konfiguration behandelt werden.

5. *Transfer*nachricht
   In einer verteilten Umgebung werden die Transfernachrichten dazu verwendet, Kopien der Konfigurationsrepräsentation weiterzugeben.

6. *Trigger*nachricht
   *Trigger*nachrichten sind spezielle Nachrichten, die für die Kommunikation zwischen der Entscheidungskomponente und der Komponente, die das Domänenkonfigurationsupdate ausführt, eingesetzt werden.

Im Folgenden betrachten wir, welche Auswirkungen die verschiedenen Parametereinstellungen einer (möglichst) verteilten Architektur auf den Nachrichtenverkehr haben. Die Tabellen 8.1 und 8.2 stellen eine Übersicht zusammen und geben pro Zeile eine mögliche Variante der Parametereinstellungen. In den Spalten der Tabelle 8.2 wird gelistet, wie oft eine Nachricht des jeweiligen Nachrichtentypes für einen einmaligen Ablauf eines erlaubten Zugriffswunsches versendet wird. Wir vernachlässigen die Nachrichten, welche ausschließlich innerhalb von Schutzdomänen stattfinden, da sie nicht zulasten des verteilten Rechensystems fallen. Optimierungen des Nachrichtenverkehrs, wie beispielsweise eine direkte Rückgabe von einem ausgeführten Dienst zu einem Dienstnutzer werden hier nicht berücksichtigt. Dies hat keine Auswirkung auf die Auswertung, da eine Änderung an dem Rückgabeverfahren sich auf alle Varianten der Parametereinstellungen gleich auswirkt.

1. Die erste Zeilen der Tabellen 8.1 und 8.2 illustrieren die zentrale Architektur, wie sie in Abschnitt 8.1 vorgestellt wurde. In diesem Fall wird ein zentrales Repositorium eingesetzt, in dem die Konfigurationsrepräsentation, die Entscheide- und Durchsetzungskomponente der Zugriffskontrolle beherbergt sind. Für die Anfrage wird eine Nachricht benötigt, ebenso für den konkreten Aufruf der Funktionalität des gewünschten Dienstes. Da sich der Dienst in einer lokalen Schutzdomäne befindet, werden für die Rückgabe zwei Nachrichten des Rückgabetyps benötigt. Eine Nachricht vom Dienst zur Durchsetzungskomponente, eine weitere von dieser

Komponente zum aufrufenden Dienstnutzer. Die Anzahl der benutzten Nachrichten beschränkt sich also auf **vier**.

In den folgenden Zeilen 2) - 4) werden Kopien der Konfigurationsrepräsentation weitergereicht, so dass die Entscheidekomponente der Zugriffskontrolle lokal bei den Teilnehmern liegt.

2. Die zweiten Zeilen der Tabellen 8.1 und 8.2 schlagen eine Parametereinstellung mit zentraler Durchsetzungskomponente vor. Aus diesem Grund findet eine Aufrufnachricht bei einem externen Dienst statt und die Rückgabe benötigt ebenfalls zwei Nachrichten. Da sich die Entscheidekomponente lokal bei den Teilnehmern befindet, muss eine Statusanfrage an das zentrale Repositorium gemacht werden, die dem jeweilig anfragenden Teilnehmer eine exklusive Kopie kommuniziert. Die Entscheidungskomponente triggert dann das Konfigurationsupdate bei der zentralen Durchsetzungskomponente. Die Anzahl der Nachrichten für diese Parametereinstellungen beschränkt sich also auf **sieben**.

Die Parametereinstellungen der Zeilen 3) - 5) der Tabellen 8.1 und 8.2 schlagen eine vollständig lokale Zugriffskontrolle vor, in der sowohl die Entscheidungs- als auch die Durchsetzungskomponenten lokal in den Schutzdomänen der Teilnehmer liegen.

3. Die dritte Zeile schlägt eine Variante vor, in der die Konfigurationsrepräsentation nach einer Statusanfrage an dem zentralen Repositorium exklusiv an den jeweilig anfragenden Teilnehmer versendet wird. Nach dem Konfigurationsupdate versendet dieser Teilnehmer die aktualisierte Konfigurationsrepräsentation zurück an das zentrale Repositorium. Falls der angefragte Dienst innerhalb der Schutzdomäne des Teilnehmers ist, produzieren die Rückgabe- und die Aufrufaktivität nur eine Nachricht. Liegt der angefragte Dienst in einer externen Schutzdomäne, werden drei Nachrichten benötigt. Die Anzahl der Nachrichten für diese Parametereinstellungen beläuft sich auf **fünf oder sieben**.

4. In der vierten Zeile werden die Paramatereinstellungen der dritten Zeile insofern variiert, als dass die Kopie der aktuellen Konfigurationsrepräsentation zwischen den Teilnehmern als eine Art *token* weitergereicht wird. Der Vorteil dieser Variante ist, dass gänzlich auf ein zentrales Repositorium verzichtet werden kann. Die Kosten der Transfernachrichten hängen in dieser Variante von der tatsächlichen Implementierung des *tokens* ab. In Kapitel 9 wird eine Implementierung vorgeschlagen, welche sich an dem in Abschnitt 2.4.2 vorgestellten Ansatz zur zustandsabhängigen Zugriffskontrolle orientiert.

5. In der fünften Zeile schließlich werden mehrere Kopien der Konfigurationsrepräsentation betrachtet. Ähnlich wie in Zeile 4) ist hier der Nachrichtentyp Transfer relevant. Verfügen alle Teilnehmer über eine eigene lokale Kopie der Konfigurations-

	Domänenkonfiguration	*entscheide*	*update*
1)	*eine Kopie*, statisch	zentral	zentral
2)	*eine Kopie*, *broadcast*	lokal	zentral
3)	*eine Kopie*, *broadcast*	lokal	lokal
4)	*eine Kopie*, *token*	lokal	lokal
5)	*mehrere Kopien*, lokal	lokal	lokal

Tabelle 8.1.: Mögliche Parametereinstellungen einer Domänenkonfiguration.

repräsentation, muss garantiert werden, dass die aktuelle Kopie dieser zur Zugriffsentscheidung vorliegt. Sinnvoll ist, wenn ein berechtigter Teilnehmer hierfür eine Blockierung aussprechen kann, so dass alle anderen Teilnehmer für einen festgesetzten Zeitraum keine Zugriffsentscheidungen treffen können. Nach einem Konfigurationsupdate müssen dann alle lokalen Kopien der Konfigurationsrepräsentation synchronisiert werden und die Blockierung kann wieder aufgehoben werden.

Abstrahieren wir zunächst von den Transfernachrichten, so benötigen die Varianten der Zeilen 4) und 5) je nach Speicherungsort des Dienstes **zwei oder vier** Nachrichten.

Vergleichen wir die mögliche Datengröße der Nachrichten, so ist diese bei den Nachrichtentypen Rückgabe, Aufruf und Trigger ähnlich. Die zugehörigen Nachrichten stoßen bei dem Empfänger die jeweilig gewünschten Aktionen an und transportieren keine großen Datensätze. Statusnachrichten sind ebenfalls vergleichbar, da diese reine „Frage-Antwort“-Nachrichten sind. Transfernachrichten dagegen kommunizieren Kopien der aktuellen Konfigurationsrepräsentation und transportieren von den genannten Nachrichtentypen die größten Datensätze. Zur Bestimmung der notwendigen Transfernachrichten für die Parametereinstellungen, vorgestellt in den Zeilen 4) und 5), schauen wird uns im folgenden Kapitel 9 eine Implementierungsvariante im Detail an.

	Anfrage	*Rückgabe*	*Aufruf*	*Status*	*Transfer*	*Trigger*
1)	1	2	1	-	-	-
2)	1	2	1	1	1	1
3)	1	2 / 1	1 / -	1	1 + 1	-
4)	1	2 / 1	1 / -	-	*Token*-Weitergabe	-
5)	1	2 / 1	1 / -	Synchronisieren & Blockieren	-	-

Tabelle 8.2.: Die Parametereinstellungen aus Tabelle 8.1 und ihre Auswirkungen auf den Nachrichtenverkehr.

Kapitel 9.

Realisierung eines dezentralen Architekturentwurfes

Für die Realisierung einer dezentralen Architektur unseres Zugriffskontroll-Ansatzes betrachten wir die Einbettung in das Szenario einer virtuellen Organisation. Virtuelle Organisationen durchlaufen einen sogenannten *Lebenszyklus*. Dieser Zyklus besteht aus der *Planungsphase*, der *Formationsphase*, der *Betriebsphase* und der *Auflösungsphase*. Für einen detaillierten Überblick über den Lebenszyklus einer virtuellen Organisation sei auf [24, 67] verwiesen. Zur Implementierung virtueller Organisationen hat sich unseres Wissens bisher noch kein Standard etabliert, in den wir unseren Ansatz der zustandsdynamischen Zugriffskontrolle einbetten könnten. Wir stellen hier losgelöst von konkreten Rahmenwerken zwei Realisierungsmöglichkeiten zustandsabhängiger und dezentraler Zugriffskontrolle vor.

Grundlegend für beide Realisierungsmöglichkeiten sind die folgenden Annahmen.

Annahme 9.1. *Das Rechensystem verfügt über eine SPKI/SDSI Infrastruktur, wie sie in Kapitel 4 zur Realisierung zentraler Zugriffskontrolle bereits eingeführt wurde.*

Annahme 9.2. *Der Einsatz von SPKI/SDSI ermöglicht die Nutzung lokaler Namensräume, wie bereits in Abschnitt 8.2 eingeführt. Wir gehen davon aus, dass Subjektbezeichner, wie sie in Konfigurationsspezifikationen verwendet werden, entweder identifizierte Subjekte oder lokale Namen bezeichnen.*

Annahme 9.3. *Jeder Teilnehmer t des Systems verfügt über ein Schlüsselpaar (pk_t, sk_t), welches aus einem öffentlichen Schlüssel pk_t und einem privaten Schlüssel sk_t besteht.*

Annahme 9.4. *Die Teilnehmer des Systems werden über die Einrichtung einer virtuellen Organisation von derem Verwalter unterrichtet. Sofern sie Interesse haben dieser virtuellen Organisation als Dienstanbieter beizutreten, treten sie mit dem Verwalter in Kontakt.*

Annahme 9.5. *Dienstanbieter betreiben für den Lebenszyklus der virtuellen Organisation spezielle Komponenten zur Zugriffskontrolle, welche von allen Teilnehmern als vertrauenswürdig angesehen werden.*

9.1. Eine *Token*-basierte Realisierung

Aus den in Abschnitt 8.4 analysierten Entwürfen haben wir für die in diesem Abschnitt vorgestellte Variante die Parametereinstellungen hervorgehoben in Abbildung 9.1 gewählt. Relevant für die Realisierung der Zugriffskontrolle sind die Formationsphase

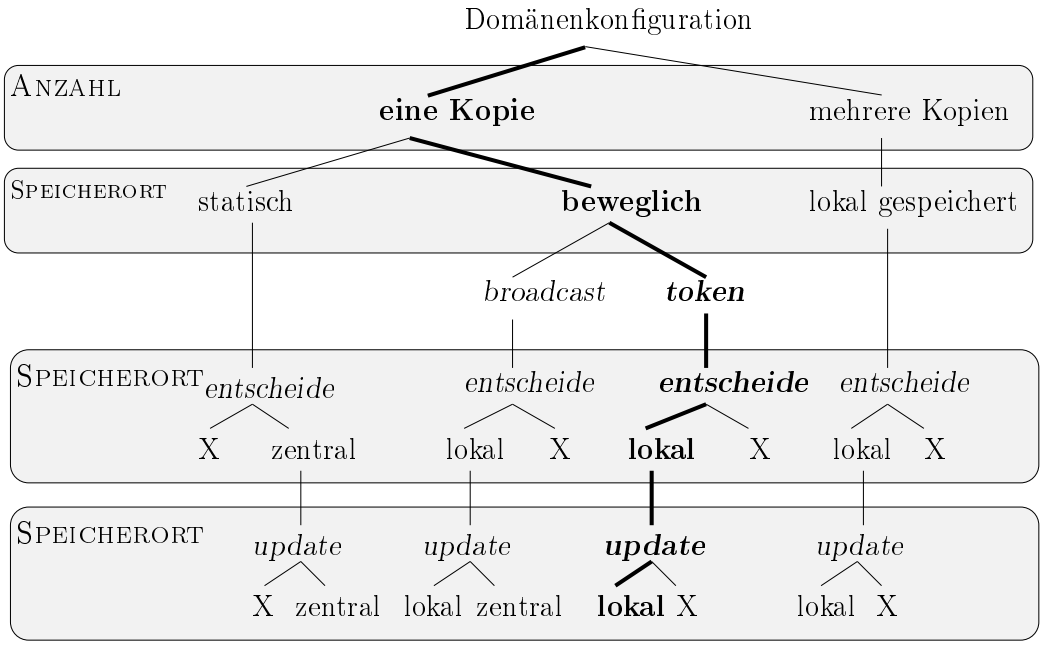


Abbildung 9.1.: Gewählte Parametereinstellungen für die *Token*-basierte Realisierung.

und die Betriebsphase einer virtuellen Organisation.

9.1.1. Die Formationsphase

Die Formationsphase beinhaltet die *Initialisierung der Zugriffskontrolle* wie folgt.

Teilnehmer, welche sich an der virtuellen Organisation beteiligen, werden Dienstanbieter genannt. Sofern die Dienstanbieter nicht bereits über ein solches verfügen, findet in dieser Phase die Generierung der kryptographischen Schlüssel statt.

Jeder Dienstanbieter verpflichtet sich für den Zeitraum des Lebenszyklus der virtuellen Organisation eine Kontrollkomponente zur Nachrichtenabwicklung und eine Entscheidungskomponente zu betreiben, wie sie in Abbildung 9.2 visualisiert ist. In der

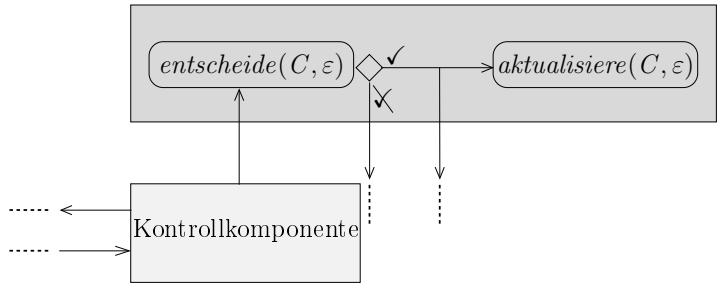


Abbildung 9.2.: Eine generische Kontroll- und Entscheidungskomponente (ohne Schreib- und Lesezugriffe).

Formationsphase wird weiterhin ein elektronisches „Schwarzes Brett“ (*bulletin board*) eingerichtet, welches der Anfrage- und Rückgabebehandlung der virtuellen Organisation dient. Dienstanbieter können das Schwarze Brett nutzen, um angebotene Dienste bekannt zu geben. Potenzielle Dienstnutzer können identifizierte Anfragen auf dem Schwarzen Brett veröffentlichen.

Der Verwalter der virtuellen Organisation spezifiziert die Zugriffskontrollpolitik $\mathcal{P}$. Die Spezifizierung kann explizit erfolgen, indem der Verwalter selbst die Konfigurationen definiert. Sie kann aber auch implizit erfolgen, indem der Verwalter Konfigurationen von den Dienstanbietern anfordert. Dann erzeugt er den zugehörigen Automaten $A_{\mathcal{P}}$ und setzt die Dienstanbieter über das Schwarze Brett der virtuellen Organisation in Kenntnis.

Sprechen wir im Folgenden von einem „aktuellen Automaten“, so meinen wir tatsächlich ein Tupel $(A_{\mathcal{P}}, q)$, bestehend aus dem der Zugriffskontrollpolitik $\mathcal{P}$ zugehörigen Automaten $A_{\mathcal{P}}$ und dem jeweils aktuellen Zustand q. Dieser „aktuelle Automat“ wird während der Betriebsphase der virtuellen Organisation als *Token* unter den Dienstanbietern weitergegeben, wie im Folgenden erklärt.

9.1.2. Die Betriebsphase

Nun startet die Betriebsphase der virtuellen Organisation. Wir betrachten einen Dienstnutzer t, der eine identifizierte Anfrage $(id : \varepsilon)$ stellt. Die zustandsdynamische Zugriffskontrolle läuft in den folgenden Schritten ab.

1. *Identifizierung des zuletzt aktiven Dienstanbieters* t_{last}
   Ein Dienstanbieter *act* entdeckt auf dem Schwarzen Brett eine Anfrage $(id : \varepsilon)$, die er aufgrund des angegebenen Sicherheitskontextes *id* einem seiner Dienste zuordnen kann und möchte die Anfrage abarbeiten.

   a) Ist $(id : \varepsilon)$ die erste Anfrage, die im Rahmen der virtuellen Organisation bearbeitet werden soll, so übernimmt der initiierende Teilnehmer die Rolle des zuletzt aktiven Dienstanbieters *last* .

   b) In allen übrigen Fällen kann der Dienstanbieter *act* anhand einer entsprechenden Block-Markierung der Anfrage erkennen, welcher Dienstanbieter der zuletzt aktive ist. Die Block-Markierung von Anfragen wird unter Schritt 2. behandelt.

2. *Block-Markierung der Anfrage* $(id : \varepsilon)$
   Möchte ein Dienstanbieter *act* ausdrücken, dass er für die Bearbeitung der Anfrage $(id : \varepsilon)$ verantwortlich ist, so muss er sie zunächst auf dem Schwarzen Brett als „in Bearbeitung“ markieren. Hierfür erzeugt er einen logischen Zeitstempel *ts* und signiert diesen gemeinsam mit der Anfrage $\{(id : \varepsilon, ts)_{sk_{act}}\}$.

3. *Weitergabe des Automaten* $\mathcal{A}_{\mathcal{P}}$
   Dienstanbieter *act* fragt bei dem zuletzt aktiven Dienstanbieter *last* den aktuellen Automaten an. Hierfür sendet er eine Statusanfrage $(status, id : \varepsilon)_{sk_{act}}$. Der zuletzt aktive Dienstanbieter kann anhand der Block-Markierung von $(id : \varepsilon)$ auf dem Schwarzen Brett die Verantwortlichkeit des Dienstanbieters *act* verifizieren. Dienstanbieter *last* verschlüsselt die aktuelle Instanz des Automaten für den Dienstanbieter *act*, signiert diese und übersendet ihn. Dienstanbieter *act* schickt eine signierte Quittung an Dienstanbieter *last*.

4. *Zugriffsentscheidung*
   Dienstanbieter *act* führt die Zugriffsentscheidung bezüglich der Anfrage $(id : \varepsilon)$ durch, wie sie in Abschnitt 8.1 festgelegt wurde. Hierfür wird seine lokale Entscheidungskomponente hinzugezogen, die nach erfolgreicher Ausführung von Schritt 3. lokalen Zugriff auf den aktuellen Automaten hat. Trifft die Entscheidungskomponente eine positive Zugriffsentscheidung, so wird mit Schritt 5. fortgefahren. Trifft die Entscheidungskomponente eine negative Zugriffsentscheidung, so wird

mit Schritt 6. fortgefahren.

5. *Zustandsübergang*
   Eine positive Zugriffsentscheidung seitens der Entscheidungskomponente zieht zum einen die Ausführung der gewünschten Funktionalität des betreffenden Dienstes nach sich und zum anderen einen Zustandsübergang im zugehörigen Automaten. Der Zustandsübergang wird wie in Abschnitt 8.1 festgelegt durchgeführt.

6. *Bearbeitet-Markierung der Anfrage*
   Dienstanbieter *act* markiert die Block-Markierung $(id : \varepsilon, ts)_{sk_{act}}$ als bearbeitet, indem er einen neuen logischen Zeitstempel hinzufügt $((id : \varepsilon, ts)_{sk_{act}}, ts + 1)_{sk_{act}}$, und nimmt somit den Status als zuletzt aktiver Dienstanbieter ein.

Die Abbildung 9.3 veranschaulicht die Architektur.

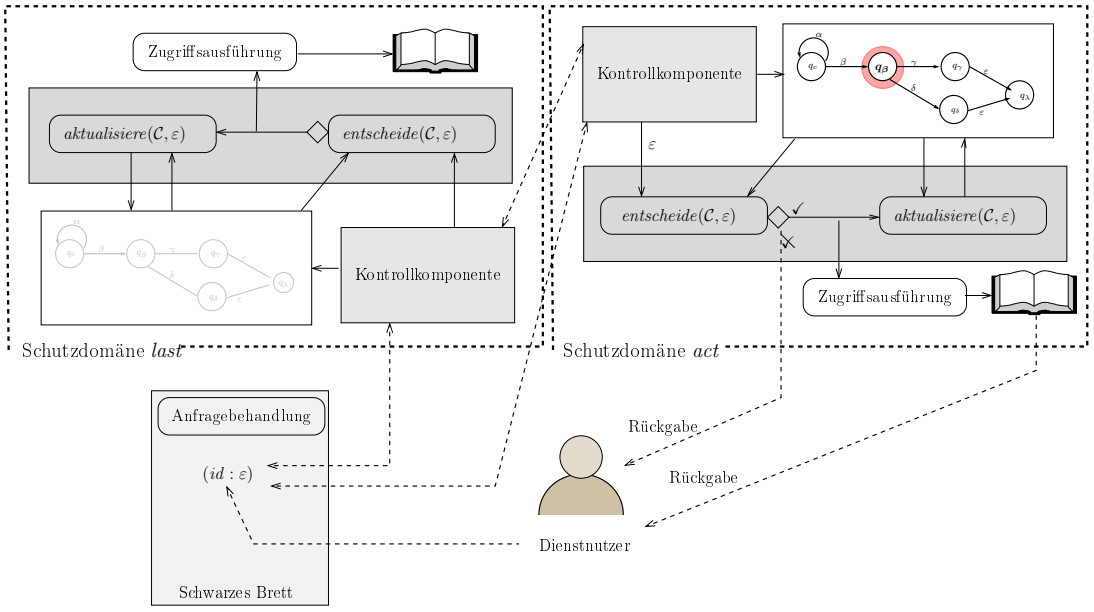


Abbildung 9.3.: Eine Architektur für die *Token*-basierte Realisierung.

9.2. Eine echt verteilte *Token*-basierte Realisierung

Die in diesem Abschnitt vorgestellte Realisierungsmöglichkeit basiert ebenfalls auf den in Abbildung 9.1 hervorgehobenen Parametereinstellungen. Im Folgenden fokussieren wir uns zunächst wieder auf die Formationsphase und die Betriebsphase einer virtuellen Organisation.

9.2.1. Die Formationsphase

Die Formationsphase beinhaltet die *Initialisierung der Zugriffskontrolle*.

Auch in dieser Realisierungsvariante werden die Teilnehmer der virtuellen Organisation Dienstanbieter genannt und generieren in dieser Phase die Schlüsselpaare. Des Weiteren sammelt jeder Teilnehmer t die ihm ausgestellten Attributzertifikate $\mathcal{NC}_t$.

Für die Betriebsphase dieser Realisierung müssen die Dienstanbieter in der Lage sein, einen Container geeignet zu verarbeiten. Neben dem Aufruf der eigentlichen Funktionalität eines Dienstes müssen die Dienstanbieter den Container vor dem Versenden aktualisieren. Hierfür werden sogenannte *Interpreter* eingesetzt. Ein Interpreter bezeichnet eine spezielle Kontrollkomponente, die von den Dienstanbietern für den Zeitraum des Lebenszyklus der virtuellen Organisation eingesetzt wird, siehe Abbildung 9.2. Zu den Aufgaben eines Interpreters[1] zählen: Empfang eines Containers, Auswertung von Bedingungen, Entschlüsselung von *parcels*, Extraktion des Kontrollausschnittes, Aufruf der gewünschten Funktionalität des Dienstes, sowie die Weiterleitung des aktualisierten Containers an nachfolgende Dienstanbieter.

Liegt einem initiierenden Dienstanbieter eine Anfrage bezüglich eines kompositionalen Webdienstes vor, so lokalisiert dieser geeignete Webdienste. Der initiierende Dienstanbieter gleicht die Anforderungen der Anfrage mit den Funktionalitäten der Webdienste ab; dies kann durch geeignete UDDI-Suchfunktionalitäten geschehen. Sind die passenden Webdienste lokalisiert, so werden anhand der WSDL-Spezifikationen die öffentlichen Schlüssel und Informationen über notwendige Eingabeparameter ausgetauscht. Anschließend bindet der initiierende Dienstanbieter die Webdienste als Dienstanbieter der kompositionalen Ausführungssequenz an die entsprechende Funktionalität. Schließlich liegt eine vollständige BPEL-Prozessbeschreibung der Ausführungssequenz mit statisch gebundenen Dienstanbietern vor. Der anfragende Dienstnutzer übermittelt dem initiierenden Dienstanbieter geforderte Attributzertifikate, die zum Aufruf von Funktionalitäten notwendig sind.

[1] Zur konkreten Realisierung wird ein Interpreter selbst als vertrauenswürdiger Dienstanbieter implementiert, welcher als „Dienst“ die Aufgaben anbietet.

9.2.2. Die Betriebsphase

Anhand der Ausführungssequenz generiert der initiierende Dienstanbieter den Container, siehe Kapitel 5 für den konkreten Aufbau eines Containers. Dann startet die Betriebsphase der virtuellen Organisation. Die Zugriffskontrolle läuft in den folgenden Schritten ab:

1. *Versenden des Containers durch den initiierenden Dienstanbieter*
   Der initiierende Dienstanbieter versendet den Container an die Kontrollkomponente des ersten Dienstanbieters WD_i, welcher für die Ausführung der ersten Funktionalität zuständig ist.

2. *Aufruf einer Funktionalität durch einen Interpreter*
   Unmittelbar nach dem Empfang des Containers extrahiert der Interpreter des ersten Dienstanbieters WD_i den verschlüsselten *parcel*-Bezeichner und den verschlüsselten, symmetrischen *parcel*-Schlüssel $Enc_{K_i}(l_i, syk_i)$ des *parcels* $parcel_i$ aus der Ausgabe-Schicht. Anschließend entschlüsselt der Interpreter den *parcel*-Schlüssel und den Bezeichner mit dem privaten Schlüssel des Dienstanbieters WD_i. Aufgrund des entschlüsselten Bezeichners l_i kann der Interpreter das symmetrisch verschlüsselte *parcel* $Enc_{syk_i}(parcel_i)$ in der Nutzlast-Schicht des Containers lokalisieren, und er entschlüsselt dieses mit dem symmetrischen Schlüssel syk_i.

   Erst zu diesem Zeitpunkt ist der Interpreter dazu imstande, auf die Kontrollvariablen seines *parcels* und dadurch auf die erforderlichen Eingabeparameter und die Signatur der auszuführenden Funktionalität zuzugreifen. Er entschlüsselt die für den Webdienst relevanten Attributzertifikate des Dienstnutzers aus der Zertifikat-Schicht, und entschlüsselt notwendige Rückgabewerte anderer Webdienste aus der Ausgabe-Schicht. Dann ruft der Interpreter die angegebene Funktionalität des Webdienstes auf.

3. *Aktualisierung des Containers durch einen Interpreter*
   Zuletzt entnimmt der Interpreter seinem *parcel* den verschlüsselten *parcel*-Schlüssel $Enc_{K_{i+1}}(l_{i+1}, syk_{i+1})$ des unmittelbar nachfolgenden Dienstanbieters WD_{i+1} und speichert diese Information in der Ausgabe-Schicht des Containers.

4. *Weitergabe des Containers durch einen Interpreter*
   Im letzten Schritt versendet der Interpreter den aktualisierten Container an den Interpreter des unmittelbar nachfolgenden Webdienstes WD_{i+1}. Aufgrund der Struktur des Containers kann der Interpreter von WD_{i+1} erst nach dem Empfang seines *parcel*-Schlüssels auf sein *parcel* und die darin enthaltenen Informationen zugreifen.

Der initiierende Dienstanbieter hat den Container so generiert, dass nach der Abarbeitung der Ausführungssequenz der aktualisierte Container zurück an ihn gesendet wird. Dann kann der initiierende Dienstanbieter eventuelle Rückgabewerte an den Dienstnutzer übermitteln.

9.3. Nachrichtenverkehr der Realisierungen

Rufen wir uns die möglichen Parametereinstellungen der dezentralen Architekturentwürfe mit den Tabellen 8.1 und 8.2 in Erinnerung, so sehen wir, dass die in Abschnitt 9.1 und Abschnitt 9.2 beschriebenen Realisierungsmöglichkeiten Varianten der *Token*-Weitergabe von Zeile 4) darstellt. Im Folgenden untersuchen wir den Nachrichtenaufwand der Realisierungsmöglichkeiten.

9.3.1. Nachrichtenverkehr der *Token*-basierten Realisierung

Zur Weitergabe der aktuellen Automateninstanz sind zwei informative Nachrichten und ein Automatentransfer notwendig. Der aktuelle Dienstanbieter fragt den Automaten bei dem zuletzt aktiven Dienstanbieter an. Nach Erhalten des Automaten sendet der aktuelle Dienstanbieter eine Quittung an den zuletzt aktiven Dienstanbieter. Da die Anfragen in dieser Realisierung über das Schwarze Brett laufen und die Dienstanbieter Markierungen an diesen vornehmen, erhöht sich der informative Nachrichtenverkehr für die Bearbeitung einer Anfrage auf vier.

Die Anzahl der Nachrichten für diese Realisierungsmöglichkeit beläuft sich also auf **acht oder zehn**. Tabelle 9.1 visualisiert den entsprechend ausgefüllten Auszug aus der Übersichtstabelle 8.2. Betrachten wir den Nachrichtenverkehr im Detail, so ist der Ablauf

Nachrichtenverkehr					
Anfrage	*Rückgabe*	*Aufruf*	*Status*	*Transfer*	*Trigger*
4	2 / 1	1 / -	2	1	-

Tabelle 9.1.: Nachrichtenverkehr der *Token*-basierten Realisierung.

wie folgt. Zur Nachrichtenbeschreibung dient die Nummerierung in Abbildung 9.4.

1. *Anfrage*-Nachrichtenverkehr
   Der Dienstnutzer t sendet seine Anfrage gemeinsam mit ihm ausgestellten Attributszertifikaten $((id : \varepsilon), \mathcal{NC})_{sk_t}$.

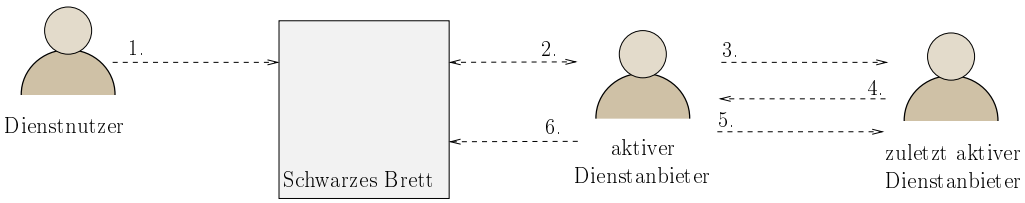


Abbildung 9.4.: Nachrichtenverkehr der *Token*-basierten Realisierung.

2. *Anfrage*-Nachrichtenverkehr
   Der aktive Dienstanbieter *act* führt einen Lesezugriff auf dem Schwarzen Brett aus, um eine Anfrage auszuwählen. Dann setzt er die Block-Markierung:
   $\{((id : \varepsilon), \mathcal{NC})_{sk_t}, ((id : \varepsilon), ts)_{sk_{act}}\}$.

3. *Status*-Nachrichtenverkehr
   Der aktive Dienstanbieter *act* sendet eine Statusanfrage an den zuletzt aktiven Dienstanbieter *last*: $((id : \varepsilon), \mathit{Status})_{sk_{act}}$.

4. *Transfer*-Nachrichtenverkehr
   Der zuletzt aktive Dienstanbieter *last* verschickt den aktuellen Automaten an Dienstanbieter *act*: $((\mathcal{A}_\mathcal{P}, q)_{pk_{act}})_{sk_{last}}$.

5. *Status*-Nachrichtenverkehr
   Der aktive Dienstanbieter *act* sendet eine signierte Quittung an Dienstanbieter *last*: $(\mathit{Quittung})_{sk_{act}}$.

6. *Anfrage*-Nachrichtenverkehr
   Der aktive Dienstanbieter *act* markiert die Anfrage als bearbeitet:
   $\{((id : \varepsilon), \mathcal{NC})_{sk_t}, (((id : \varepsilon), ts)_{sk_{act}}, ts + 1)_{sk_{act}}\}$.

9.3.2. Nachrichtenverkehr der echt verteilten *Token*-basierten Realisierung

Zur Weitergabe des Containers sind in dieser Realisierungsmöglichkeit keine informativen Nachrichten notwendig. Da der initiierende Dienstanbieter die Ausführungssequenz in die Struktur des Containers codiert, wird dieser von einem Webdienst zum nächsten Webdienst übermittelt. Aufgrund der statischen Ausführungssequenz, erfolgt nur eine Anfrage eines Dienstnutzers zu Beginn der Betriebsphase der virtuellen Organisation. Gehen wir von nur vertrauenswürdigen Interpretern (Kontrollkomponenten) der eingebundenen Webdienste aus, so sind während der Betriebsphase keine Nachrichten und

keine Übermittlungen des Containers zum initiierenden Dienstanbieter notwendig. In diesem Fall wird den Interpretern vertraut, dass sie autonom über eventuell auftretende (Verzweigungs)-Bedingungen entscheiden. Das Vorkommen nicht vertrauenswürdiger Interpreter wird später in Abschnitt 9.4 betrachtet. Eventuelle Rückgabewerte der Webdienste gehen nicht direkt zum Dienstnutzer, sondern werden ebenfalls über den Container übermittelt. Statusanfragen und *Trigger*nachrichten müssen in dieser Realisierungsmöglichkeit nicht berücksichtigt werden.

Die Anzahl der Nachrichten für diese Realisierungsmöglichkeit beläuft sich also auf **fünf**. Tabelle 9.2 visualisiert den entsprechend ausgefüllten Auszug aus der Übersichtstabelle 8.2.

Nachrichtenverkehr					
Anfrage	*Rückgabe*	*Aufruf*	*Status*	*Transfer*	*Trigger*
1	2	1	-	1	-

Tabelle 9.2.: Nachrichtenverkehr der echt verteilten *Token*-basierten Realisierung.

9.4. Sicherheitsbetrachtungen

In diesem Kapitel betrachten wir mögliche *Angriffe* auf konkrete Sicherheitsziele in unserem Szenario. Konkrete Spezialfälle der Sicherheitsbetrachtungen bezüglich der echt verteilten *Token*-basierten Realisierung wurden im Rahmen der Diplomarbeit [72] untersucht.

Ein *Angreifer* ist ein Teilnehmer, der hierfür Schwächen unserer Architektur ausnutzt oder das vorgestellte Verfahren zur Zugriffskontrolle in nicht vorgesehener Weise missbraucht. Folgende Annahmen legen wir für die Sicherheitsbetrachtungen zugrunde.

Annahme 9.6. *Der der Zugriffskontrollspezifikation zugehörige Automat wurde korrekt erstellt.*

Annahme 9.7. *Der initiierende Dienstanbieter, Verwalter, einer virtuellen Organisation verhält sich für die Dauer des Lebenszyklus dieser vertrauenswürdig.*

Annahme 9.8. *Die lokalen Kontroll- und Entscheidungskomponenten der Dienstanbieter sind nicht korrumpiert.*

Annahme 9.9. *Die im System erstellten und verwendeten Signaturen können durch einen polynomiell beschränkten Angreifer nicht gefälscht werden.*

Folgende Sicherheitsziele betrachten wir im Rahmen der Realisierung unseres Ansatzes.

Fälschung von ...	Abwehrmechanismus	wann erfolgreich?
Block-Markierung	Signatur	geheime Schlüssel kompromittiert
vorgelegten Zertifikaten	Signatur	geheime Schlüssel kompromittiert
Anfragen	Signatur	geheime Schlüssel kompromittiert

Tabelle 9.3.: Angriffe gegen die Authentizität.

9.4.1. Authentizität

Unter der Authentizität eines Teilnehmers verstehen wir die Echtheit des Teilnehmers, die je nach Realisierung anhand der eindeutigen Identität oder anhand der durch vorgelegte Zertifikate beweisbaren, ihn charakterisierenden freien Merkmale überprüfbar ist.

Unter der Authentizität der Daten verstehen wir die Echtheit der Daten. Dies bedeutet, dass eine nachträgliche Veränderung von Daten oder das Fälschen von Signaturen unterbunden werden sollte.

***Token*-basierte Realisierung:** In Schritt 1. der Betriebsphase muss ein Dienstanbieter den zuletzt aktiven Dienstanbieter identifizieren, um in Besitz des aktuellen Automaten zu gelangen. Da das Protokoll in Schritt 2. vorsieht, dass Dienstanbieter die Block-Markierungen signieren, kann unter Voraussetzung der Annahme 9.9 der jeweils zuletzt aktive Dienstanbieter identifiziert werden.

Unter Annahme 9.2, dass Subjektbezeichner lokale Namen darstellen, wird der Entscheidungskomponente die in Abschnitt 8.2 vorgestellte Evaluierungsfunktion zur Namensauflösung vorgeschaltet. Hierdurch wird die Authentizität der anfragenden Teilnehmer sichergestellt, bevor über den tatsächlichen Zugriffswunsch entschieden wird.

Mögliche Angriffe auf die Authentizität der Teilnehmer sind Versuche, die Block-Markierungen zu fälschen, vorgelegte Zertifikatmengen zu manipulieren oder Anfragen auszutauschen. Tabelle 9.3 gibt einen Überblick über mögliche Angriffe.

Echt verteilte *Token*-basierte Realisierung: Die einzige Möglichkeit eines Angriffs auf die Authentizität der Teilnehmer sind Versuche, vorgelegte Zertifikatmengen zu manipulieren. Dies wird aufgrund der verwendeten Signaturen unterbunden. Tabelle 9.4 skizziert den möglichen Angriff.

Die Authentizität der Daten wird in beiden Realisierungsmöglichkeiten durch den Einsatz fälschungssicherer digitaler Signaturen gewährleistet.

Fälschung von ...	Abwehrmechanismus	wann erfolgreich?
vorgelegten Zertifikaten	Signatur	geheime Schlüssel kompromittiert

Tabelle 9.4.: Angriff gegen die Authentizität.

9.4.2. Verfügbarkeit

Unter Verfügbarkeit verstehen wir, dass authentifizierte Teilnehmer nicht in der Inanspruchnahme ihres erlaubten Zugriffswunsches beeinträchtigt werden können. Dies bedeutet, dass Dienstanbieter die notwendigen Informationen erhalten, um den zu einem erlaubten Zugriffswunsch gehörenden Dienst ausführen zu können. Es existieren eine Reihe von Angriffen, welche sich gegen die Verfügbarkeit richten, diese werden allgemein auch als *Denial of Service* bezeichnet.

***Token*-basierte Realisierung:** Zur Ausführung eines gewünschten Dienstes benötigt ein Dienstanbieter die Anfrage des Teilnehmers, vorgelegte Zertifikate und den aktuellen Automaten. Die Anfrage, sowie vorgelegte Zertifikate entnimmt der Dienstanbieter dem Schwarzen Brett der virtuellen Organisation. Die Weitergabe des aktuellen Automaten erfolgt nach Schritt 3. des Protokolls.

Mögliche Angriffe auf die Verfügbarkeit sind somit Angriffe auf die Weitergabe des Automaten. Da wir die Authentizität der Teilnehmer sicherstellen, können wir davon ausgehen, dass der aktive Dienstanbieter sich an den „echten“ zuletzt aktiven Dienstanbieter wendet. Einem Angreifer bleibt somit nur die Möglichkeit, die Kommunikation durch das Abfangen des Automaten zu stören. Im Allgemeinen kann eine Störung der Kommunikation der Teilnehmer untereinander, wie beispielsweise das Zurückhalten oder Verwerfen von Nachrichten nicht verhindert werden. In diesen Fällen können wir aber Mechanismen einsetzen, mit denen eine nachträgliche Aufdeckung des Angriffes ermöglicht wird. Dieser Aspekt wird später in Abschnitt 9.4.5 aufgegriffen.

Tabelle 9.5 stellt den möglichen Angriff dar.

Abfangen ...	Abwehrmechanismus	wann erfolgreich?
des Automaten	nicht möglich	in jedem Fall

Tabelle 9.5.: Angriff gegen die Verfügbarkeit.

Echt verteilte *Token*-basierte Realisierung: Mögliche Angriffe sind hier Angriffe auf die Weitergabe eines Containers oder das Manipulieren von Informationen aus den Schichten eines Containers vor der Weitergabe an den nächsten, vorgesehenen Empfänger. Auch in dieser Realisierungsmöglichkeit kann das Blockieren der Weitergabe eines Containers nicht verhindert werden, da die eingebundenen Webdienste keine zeitlichen Vorgaben zur Ausführung der Funktionalitäten haben.

Auch das Entfernen von Informationen aus den Schichten eines Container kann nicht verhindert werden. Allerdings können Angriffe dieser Art durch nachfolgende Empfänger aufgedeckt werden, da die Nutzlast- und die Zertifikat-Schicht vom initiierenden Dienstanbieter signiert worden ist.

Tabelle 9.6 stellt mögliche Angriffe dar.

Abfangen ...	Abwehrmechanismus	wann erfolgreich?
des Containers	nicht möglich	in jedem Fall
Entfernen ...	**Abwehrmechanismus**	**wann erfolgreich?**
von Informationen	Signatur	geheime Schlüssel kompromittiert

Tabelle 9.6.: Angriffe gegen die Verfügbarkeit.

9.4.3. Vertraulichkeit

Unter Vertraulichkeit verstehen wir, dass die Dienstanbieter nur solche Informationen über den Ablauf der virtuellen Organisation erhält, die minimal notwendig zur Ausführung eines zu einem erlaubten Zugriffswunsch gehörenden Dienstes sind. Alle zusätzlichen Informationen über den anfragenden Teilnehmer und andere Dienstanbieter sollen vertraulich bleiben.

***Token*-basierte Realisierung:** Die Vertraulichkeit kann in der Realisierungsmöglichkeit der *Token*-Weitergabe nicht gewährleistet werden. Jeder Dienstanbieter, der einen gewünschten Dienst ausführen soll, erhält nach Schritt 3. des Protokolls den gesamten aktuellen Automaten. Dieser Automat beinhaltet die gesamte Information über die Zugriffskontrollspezifikation. Somit weiß ein Dienstanbieter, welche anderen Dienstanbieter der virtuellen Organisation beigetreten sind und welche Teilnehmer welche Zugriffsrechte zugesprochen bekommen.

Echt verteilte *Token*-basierte Realisierung: Es existieren zwei naheliegende Möglichkeiten, mit denen ein Angreifer die Vertraulichkeit der Daten eines Containers gefährden kann. Eine Möglichkeit ist das Verschaffen unbefugten Zugriffs auf die Informationen eines Containers. Dies ist allerdings nur möglich, falls der Angreifer in den Besitz der zugehörigen symmetrischen Schlüssel gelangt. Durch den speziellen Aufbau eines Containers ist sichergestellt, dass diese Schlüssel ausschließlich den vorgesehenen Webdiensten zur Verfügung gestellt werden können.

Eine andere Möglichkeit besteht in der Überwachung des Netzverkehrs. Ein Angreifer kann während der Betriebsphase versuchen, Rückschlüsse über eingebundene Webdienste und deren aufgerufene Funktionalitäten zu erlangen. Die Anzahl der *parcels* in der Nutzlast-Schicht eines Containers entspricht zwar der Anzahl der potenziell vorgesehenen Webdienste, aber allein anhand der Anzahl lassen sich keine Informationen über konkret eingebundene Webdienste erlangen. Zudem sind die verschlüsselten *parcels* in zufälliger Anordnung. Nicht verhindert werden kann allerdings eine Überwachung der Kommunikation der eingebundenen Webdienste.

Tabelle 9.7 stellt mögliche Angriffe dar.

ZUGRIFF AUF ...	ABWEHRMECHANISMUS	WANN ERFOLGREICH?
den Container	Signatur	geheime Schlüssel kompromittiert
ABHORCHEN ...	ABWEHRMECHANISMUS	WANN ERFOLGREICH?
der Kommunikation	nicht möglich	Zugang zu Verkehrsdaten

Tabelle 9.7.: Angriffe gegen die Vertraulichkeit.

9.4.4. Integrität

Unter Integrität verstehen wir die Einhaltung der jeweiligen Zugriffskontrollspezifikation.

***Token*-basierte Realisierung:** Annahme 9.6 setzt voraus, dass anhand einer Zugriffskontrollspezifikation $\mathcal{P}$ ein korrekter Automat $\mathcal{A}_\mathcal{P}$ erstellt wird. Zur Wahrung der Integrität muss gewährleistet werden, dass der durch den Automaten gegebene (strukturierte) Ablauf eingehalten wird. Da der Automat nur verschlüsselt und zusätzlich signiert weitergereicht wird, siehe Schritt 3. des Protokolls, gehen wir davon aus, dass einem aktiven Dienstanbieter ein korrektes Tupel $(\mathcal{A}_\mathcal{P}, q)$ vorliegt. Interpretieren wir dieses Tupel als *Autorisierung* für den Dienstanbieter, einen erlaubten Zustandsübergang auszuführen, so müssen wir die folgenden Eigenschaften sicherstellen:

1. Die *Autorisierung* $(\mathcal{A}_{\mathcal{P}}, q)$ ist nur für den jeweiligen aktiven Dienstanbieter gültig.

2. Nach Ausführung eines erlaubten Zustandsüberganges ist die *Autorisierung* $(\mathcal{A}_{\mathcal{P}}, q)$ auch für den jeweiligen aktiven Dienstanbieter nutzlos.

3. Der erlaubte Zustandsübergang überführt den Automaten in einen korrekten Anschlusszustand.

Die Eigenschaft 1. wird dadurch sichergestellt, dass nach Schritt 3. des Protokolls der aktuelle Automat mit dem öffentlichen Schlüssel des aktiven Dienstanbieters verschlüsselt wird. Ein Angreifer kann zwar die Übermittlung des aktuellen Automaten stören, siehe Sicherheitsziel Verfügbarkeit, kann aber einen eventuell abgefangenen aktuellen Automaten aufgrund der Verschlüsselung nicht für sich nutzen.

Die Eigenschaft 2. wird durch den Aspekt der Annahme 9.8 sichergestellt, dass die Kontrollkomponente eines Dienstanbieters nicht korrumpiert ist. Selbst eine vertrauenswürdige Entscheidungskomponente kann bei erneuter Eingabe des aktuellen Automaten nicht feststellen, ob die Eingabe eine duplizierte und somit ein Betrugsversuch ist. Allein die Kontrollkomponente veranlasst nach einem erlaubten Zustandsübergang eine Bearbeitet-Markierung der zugehörigen Anfrage, siehe Schritt 6. des Protokolls.

Die Eigenschaft 3. wird ebenfalls durch Annahme 9.8 gesichert.

Echt verteilte *Token*-basierte Realisierung: Die spezifizierte Ausführungssequenz wird in den realisierenden Container codiert. Der grundlegende Aufbau eines Containers wurde in Kapitel 5 vorgestellt. Zur kompositionalen Spezifizierung stehen die Kontrollflussstrukturen *Sequenz*, *parallele Ausführung*, *bedingte Verzweigung* und *Iteration* zur Verfügung. Die Zertifikat- und die Ausgabe-Schicht eines Containers werden nicht von den verwendeten Kontrollflussstrukturen beeinflusst. Nur der innere Aufbau der *parcels* der Nutzlast-Schicht codiert die zur Spezifizierung verwendeten Kontrollflussstrukturen. Im Folgenden betrachten wir die einzelnen Strukturen:

- Sequenz:
  Der Aufbau der einzelnen *parcels* in der Nutzlast-Schicht bei einer Sequenz von n Webdiensten (siehe Beispiel 5.5) zeigt, dass die Integrität der Ausführungsreihenfolge gegeben ist. Ein Webdienst WD_i erhält nur dann seinen symmetrischen *parcel*-Schlüssel syk_i, falls sein direkter Vorgänger WD_{i-1} seinen *parcel*-Schlüssel syk_{i-1} kennt und somit auf sein *parcel* zugreifen konnte. Dies bedeutet, dass nur dann WD_{i-1} Zugriff auf $Enc_{K_i}(l_i, syk_i)$ hat. Durch diese Struktur ist sichergestellt, dass ein Dienstanbieter erst dann auf seine Kontrollinformationen zugreifen kann, wenn die vorgehenden Webdienste ihre Funktionalitäten bereits ausgeführt haben.

- parallele Ausführung:
  Der Aufbau der einzelnen *parcels* in der Nutzlast-Schicht bei einer parallelen Ausführung von Webdiensten (siehe Beispiel 5.6) zeigt, dass die Integrität der Ausführungsreihenfolge gegeben ist. Wie in der Semantik einer BPEL Spezifikation zur parallelen Ausführung von Webdiensten vorgesehen, fungiert ein Webdienst als „Aufspaltungsstelle“ und ein Webdienst als „Synchronisationsstelle“.

  Der „aufspaltende“ Webdienst muss aufgrund seiner *parcel*-Struktur an jeden seiner unmittelbaren Nachfolger die passenden *parcel*-Schlüssel verteilen, die in verschlüsselter Form in seinem *parcel* enthalten sind, nachdem er die gewünschte Funktionalität ausgeführt hat. Auch die „Synchronisationsstelle“ muss zunächst von allen unmittelbaren Vorgängern die entsprechenden Teilgeheimnisse erhalten, um seinen *parcel*-Schlüssel rekonstruieren zu können.

- bedingte Verzweigung:
  Tritt in der Ausführungsreihenfolge eine bedingte Verzweigung auf, so muss der Webdienst, der als „Aufspaltungsstelle“ fungiert, eine Bedingung auswerten. Betrachten wir zum Aufbau der *parcels* eines Containers das Beispiel 5.7, so sehen wir, dass dort der Webdienst WD_1 als „Aufspaltungsstelle“ nur genau einen seiner $n-2$ potenziell nachfolgenden Webdienste als tatsächlichen Nachfolger bestimmt. Da alle $n-2$ verschlüsselten *parcel*-Schlüssel in dem $parcel_1$ enthalten sind, muss dem Webdienst WD_1 vertraut werden, dass er die Bedingung korrekt auswertet und anschließend ausschließlich dem vorgesehenen Nachfolger den aktualisierten Container zusendet und den zugehörigen, verschlüsselten *parcel*-Schlüssel mitteilt. Die Annahme 9.8 sichert, dass die eingesetzten Interpreter vertrauenswürdig sind. Aus diesem Grund ist die Integrität an dieser Stelle der Ausführungsreihenfolge gewährleistet.

- Iteration:
  Die weitgehend statische Struktur eines Containers birgt Probleme im Hinblick auf eine Ausführungsreihenfolge, in der beliebige Iterationen auftreten. Das Problem an dem Aufbau der *parcels* in der Nutzlast-Schicht eines Containers ist, dass teilnehmende Webdienste $\{WD_1, \ldots, WD_n\}$ ihre *parcel*-Schlüssel, und somit auch die in den entsprechenden *parcels* enthaltenen Informationen bereits nach einem ersten Durchlauf kennen. Bei einem wiederholten Durchlauf würde jedem Webdienst WD_i der verschlüsselte *parcel*-Schlüssel mitgeteilt, den er bereits aus der Vorrunde kennt. Eine dynamische Aktualisierung der *parcel*-Schlüssel oder Ausgabe-Schlüssel sind in dem bisherigen Ansatz nicht vorgesehen.

  Eine pragmatische Lösung des Problems scheitert unter anderem daran, dass beim Auftreten von Iterationen innerhalb einer BPEL Prozessdefinition im Allgemeinen nicht exakt vorherbestimmt werden kann, wie oft eine Iteration auszuführen ist, da

diese Entscheidung dynamisch getroffen wird. Diese Problematik wurde im Rahmen der Diplomarbeit [72] untersucht. Die Ergebnisse der Diplomarbeit zielten darauf, die beliebige Iteration durch eine strukturierte Iteration zu ersetzten, die dann in der Container-Struktur umgesetzt werden kann.

9.4.5. Die zentrale Instanz

Zur nachträglichen Überprüfung, ob die Zugriffskontrollspezifikation eingehalten wurde, kann in jedem Fall eine vertrauenswürdige *zentrale Instanz* eingerichtet werden. Diese Instanz kann im Nachhinein einen Angriff aufdecken. Durch die logischen Zeitstempel der Block- und Bearbeitet-Markierungen auf dem Schwarzen Brett wird es einer solchen Instanz ermöglicht, den Ablauf zu rekonstruieren. Anhand der Zugriffskontrollspezifikation des Verwalters der virtuellen Organisation kann ein korrekter Automat erzeugt werden. Anhand des Ablaufes, der auf dem Schwarzen Brett dokumentiert ist, kann die zentrale Instanz das zugehörige Wort ableiten, das als Eingabe für den Automaten dient. Akzeptiert der Automat das Wort nicht, so ist an dem entsprechenden nicht ausführbaren Zustandsübergang ein Angriff gefunden worden. Dem nicht ausführbaren Zustandsübergang kann eine entsprechende Eintragung auf dem Schwarzen Brett zugeordnet werden. Da sämtliche Eintragungen auf dem Schwarzen Brett signiert sind, kann die zentrale Instanz ebenso feststellen, welcher Teilnehmer beteiligt ist.

Sollte der Verdacht bestehen, dass die eingesetzten Interpreter nicht vertrauenswürdig sind, so kann in diesem Fall ebenfalls eine vertrauenswürdige zentrale Instanz eingerichtet werden. Diese Instanz übernimmt dann sensible Aufgaben eines nicht-vertrauenswürdigen Interpreters, wie beispielsweise das Auswerten von Bedingungen vor einer bedingten Verzweigung. Da der initiierende Dienstanbieter die Ausführungsreihenfolge in den Container codiert, muss er vor der Betriebsphase die Entscheidung über die Vertrauenswürdigkeit der potenziell eingesetzten Interpreter treffen. Dann kann er den Container so generieren, dass dieser an den sensiblen Stellen der Ausführungsreihenfolge zur vertrauenswürdigen zentralen Instanz gesendet wird.

Teil V.

Einordnung und Bewertung

Dieser Teil beschließt die Arbeit, indem die hier erarbeiteten Ansätze mit wichtigen Vertretern aus der Literatur vergleichend bewertet werden.

Kapitel 10.

Bewertung

In dieser Arbeit wurde ein Rahmenwerk entworfen, mit dem eine kompositionale und zustandsbasierte Zugriffskontrolle für Web-basierte Umgebungen ermöglicht wird.

Traditionelle Zugriffskontrollmodelle überprüfen die Authentizität einer Zugriffsanfrage eines Teilnehmers und treffen die Zugriffsentscheidung aufgrund zuvor vergebener Zugriffsrechte. In klassischen Strategien wird für jedes zu schützende Objekt im System individuell und meist lokal festgelegt, welche Aktionen einzelne Subjekte auf diesem Objekt ausführen dürfen. Für moderne und verteilte Rechensysteme, wie Web-basierte Umgebungen, müssen Zugriffskontrollmodelle entworfen werden, die eine potenziell sehr großen Anzahl unbekannter Teilnehmer und eine Vielfalt von (Web)-Diensten kontrollieren können. Für Web-basierte Umgebungen typisch sind kooperative Zusammenschlüsse von autonomen Dienstanbietern. Diese Zusammenschlüsse wirken sich sowohl auf die Struktur angebotener Dienste aus, als auch auf die Anforderungen an einzusetzende Zugriffskontrollpolitiken. Bisher existieren Zugriffskontrollmodelle, die (vorrangig) für die Vergabe und Kontrolle von Zugriffsrechten für einzelne Funktionalitäten angebotener Dienste eingesetzt werden können. Einige Vorschläge zur Zugriffskontrolle einzelner Funktionalitäten im Anwendungsgebiet strukturierter Dienste finden sich für sogenannte *workflow*-Systeme. Um komplexe Anwendungen kontrollieren zu können, müssen Zugriffskontrollpolitiken nicht nur für einzelne, atomare Funktionalitäten der Dienste festgelegt werden, sondern auch für komplexe Folgen der Funktionalitäten strukturierter Dienste. Dieses Konzept findet sich in zustandsbasierten Zugriffskontrollmodellen wieder. Eine notwendige Erweiterung der zustandsbasierten Zugriffskontrollmodelle ergibt sich unmittelbar aus den möglichen Kompositionsarten, die *workflow*-Systeme für die Spezifizierung strukturierter Dienste anbieten. Bisher sind unseres Wissens noch keine kompositionalen und zustandsbasierte Zugriffskontrollmodelle zur Kontrolle und Überwachung strukturierter Dienste vorgeschlagen worden.

Die Tabelle 10.1 visualisiert den abschließenden Vergleich der in Kapitel 2 betrachteten Zugriffskontrollansätze im Hinblick auf die in Abschnitt 1.3 erarbeiteten Kriterien einer adäquaten Zugriffskontrolle für Web-basierte Umgebungen. Die letzte Spalte der

Ansatz	Komposit.	Zustandsbas.	Verteilt	Parameter	Rollen
[19]:	ja	nein	nein	ja	nein
[84]:	ja	ja	nein	nein	nein
[86]:	nein	nein	ja	ja	nein
[47]:	ja	nein	nein	ja	nein
[57]:	nein	nein	ja	nein	implizit
[62, 63]:	nein	nein,*ja*	ja	ja	ja
[53]:	nein	ja	nein	ja	ja
[32]:	nein	ja	ja	ja	ja
[21]:	nein	ja	nein	nein	ja
[25]:	nein	ja	nein	nein	nein
unser Ansatz	ja	ja	ja	ja	ja

Tabelle 10.1.: Abschließender Vergleich der Zugriffskontrollansätze.

Tabelle 10.1 visualisiert die Eigenschaften des in dieser Arbeit entwickelten Rahmenwerks für zustandsdynamische Zugriffskontrollpolitiken. In unserer Arbeit haben wir in Kapitel 8 und Kapitel 9 aufgezeigt, wie der Ansatz zustandsdynamischer Zugriffskontrollpolitiken sowohl zentral als auch dezentral durchgesetzt werden kann. Die Tabelle 10.2 gibt eine Übersicht darüber, ob die aus der Literatur besprochenen Ansätze Vorschläge für eine zentrale oder dezentrale Umsetzung enthalten.

Ansatz	zentrale Umsetzung	dezentrale Umsetzung
[19]:	ja	nein
[84]:	nein	nein
[86]:	skizziert	nein
[47]:	ja	nein
[57]:	nein	ja
[62, 63]:	nein	ja
[53]:	skizziert	skizziert
[32]:	ja	ja
[21]:	ja	nein
[25]:	nein	nein
unser Ansatz	ja	ja

Tabelle 10.2.: Vergleich der Zugriffskontrollansätze im Hinblick auf Vorschläge zur zentralen oder dezentralen Implementierung.

Literaturverzeichnis

[7] Adam, Nabil R. ; Atluri, Vijayalakshmi ; Bertino, Elisa ; Ferrari, Elena: A content-based authorization model for digital libraries. In: *IEEE Transactions Knowledge and Data Engineering* 14 (2002), Nr. 2, S. 296–315

[8] Antonio Cau, Ben M. ; Zedan, Hussein: *Interval temporal logic.* Software Technology Research Laboratory, De Montfort University, UK. `http://www.cse.dmu.ac.uk/STRL/ITL/`. Version: 2008

[9] Ao, Xuhui ; Minsky, Naftaly H.: Flexible regulation of distributed coalitions. In: *8th European Symposium on Research in Computer Security* Bd. 2808, Springer Verlag, 2003 (Lecture Notes in Computer Science), S. 39–60

[10] Ao, Xuhui ; Minsky, Naftaly H.: Regulated Delegation in Distributed Systems. In: *POLICY*, IEEE Computer Society, 2006, S. 215–226

[11] Atluri, Vijayalakshmi ; Chun, Soon A. ; Mazzoleni, Pietro: Chinese wall security for decentralized workflow management systems. In: *Journal of Computer Security* 12 (2004), Nr. 6, S. 799–840

[12] Bell, D. E. ; LaPadula, Leonard J.: Secure Computer Systems: A Mathematical Model, Volume II. In: *Journal of Computer Security* 4 (1996), Nr. 2/3, S. 229–263

[13] Berardi, Daniela ; Calvanese, Diego ; Giacomo, Giuseppe D. ; Lenzerini, Maurizio ; Mecella, Massimo: Automatic service composition based on behavioral descriptions. In: *International Journal of Cooperative Information Systems* 14 (2005), Nr. 4, S. 333–376

[14] Bertino, Elisa ; Ferrari, Elena ; Atluri, Vijayalakshmi: The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. In: *ACM Transactions on Information and System Security* 2 (1999), Nr. 1, S. 65–104

[15] Biskup, Joachim: Credential-basierte Zugriffskontrolle: Wurzeln und ein Ausblick. In: Schubert, Sigrid E. (Hrsg.) ; Reusch, Bernd (Hrsg.) ; Jesse, Norbert (Hrsg.): *GI Jahrestagung*, GI, 2002, S. 423–428

[16] Biskup, Joachim ; Karabulut, Yücel: A hybrid PKI model: Application to secure mediation. In: *16th Annual IFIP WG 11.3 Working Conference on Data and Application Security*, Kluwer Academic Publishers, 2003 (Research Directions in Data and Application Security), S. 271 – 282

[17] Biskup, Joachim ; Leineweber, Thomas: State-dependent security decisions for distributed object-systems. In: Olivier, Martin S. (Hrsg.) ; Spooner, David L. (Hrsg.): *Proceedings of the 15th Annual Working Conference on Database and Application Security* Bd. 215, Kluwer Academic Publishers, 2002 (IFIP Conference Proceedings), S. 105–118

[18] Blaze, Matt ; Feigenbaum, Joan ; Ioannindis, John ; Kermytis, Angelos D.: *The KeyNote trust management system version 2.* http://www.rfc-editor.org/rfc/rfc2704.txt, September 1999. – Internet RFC 2704

[19] Bonatti, Piero ; de Capitani di Vimercati, Sabrina ; Samarati, Pierangela: An algebra for composing access control policies. In: *ACM Transactions on Information and System Security* 5 (2002), Nr. 1, S. 1–35

[20] Bonatti, Piero A. ; Samarati, Pierangela: A Uniform Framework for Regulating Service Access and Information Release on the Web. In: *Journal of Computer Security* 10 (2002), Nr. 3, S. 241–272

[21] Botha, Reinhardt A. ; Eloff, Jan H. P.: A framework for access control in workflow systems. In: *Information Management & Computer Security* 9 (2001), Nr. 3, S. 126–133

[22] Brands, Stefan A.: *Rethinking Public Key Infrastructures and Digital Certificates.* 1. Auflage. MIT Press, 2000. – ISBN 0–262–02491–8

[23] Brewer, David F. C. ; Nash, Michael J.: The chinese wall security policy. In: *IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, 1989, S. 206–214

[24] Camarinha-Matos, Luis M. ; Afsarmanesh, Hamideh ; Ollis, Martin: *Virtual Organizations: Systems and Practices.* Springer, 2005

[25] Chander, Ajay ; Mitchell, John C. ; Dean, Drew: A state-transition model of trust management and access control. In: *14th IEEE Computer Security Foundations Workshop*, IEEE Computer Society, 2001, S. 27–43

[26] Chaum, David: Security without identification: transaction systems to make big brother obsolete. In: *Communications of the ACM (CACM)* 28 (1985), Nr. 10

[27] Clarke, Dwaine E. ; Elien, Jean-Emile ; Ellison, Carl M. ; Fredette, Matt ; Morcos, Alexander ; Rivest, Ronald L.: Certificate Chain Discovery in SPKI/SDSI. In: *Journal of Computer Security* 9 (2001), Nr. 4, S. 285–322

[28] Coetzee, Marijke ; Eloff, Jan H. P.: Towards Web Service access control. In: *Computers & Security* 23 (2004), Nr. 7, S. 559–570

[29] Denning, Dorothy: *Cryptography and Data Security.* Addison-Wesley, 1982

[30] Department of Defence (USA): *Trusted Computer Systems Evaluation Criteria.* DOD 5200.28-STD, 1985

[31] Dreo-Rodosek, Gabi: A Generic Model for IT Services and Service Management. In: *Integrated Network Management* Bd. 246, Kluwer Academic Publishers, 2003 (IFIP Conference Proceedings), S. 171–184

[32] Eckert, Christian: *Zustandsabhängige Sicherheitsspezifikationen und ihre Durchsetzung*, Universität Dortmund, Diss., 1997

[33] Eckert, Claudia: *IT-Sicherheit: Konzepte - Verfahren - Protokolle.* 4. Auflage. Oldenbourg Verlag, 2006. – ISBN 978–3–486–57851–5

[34] Ellison, Carl M. ; Frantz, Bill ; Lampson, Butler ; Rivest, Ronald L. ; Thomas, Brian M. ; Ylonen, Tatu: *Simple public key certificate.* http://world.std.com/cme/html/spki.html. `http://world.std.com/~cme/html/spki.html`. Version: July 1999

[35] Ellison, Carl M. ; Frantz, Bill ; Lampson, Butler ; Rivest, Ronald L. ; Thomas, Brian M. ; Ylonen, Tatu: *SPKI certificate theory.* http://www.rfc-editor.org/rfc/rfc2693.txt. `http://world.std.com/~cme/html/spki.html`. Version: September 1999. – Internet RFC 2693

[36] Emig, Christian: *Zugriffskontrolle in dienstorientierten Architekturen*, Universität Karlsruhe (TH), Diss., 2008

[37] Ferraiolo, David F. ; Sandhu, Ravi S. ; Gavrila, Serban I. ; Kuhn, D. R. ; Chandramouli, Ramaswamy: Proposed NIST standard for role-based access control. In: *ACM Transactions on Information and System Security* 4 (2001), Nr. 3, S. 224–274

[38] Goguen, Joseph A. ; Meseguer, José: Security Policies and Security Models. In: *IEEE Symposium on Security and Privacy*, 1982, S. 11–20

[39] Gollmann, Dieter: *Computer Security.* 2. Auflage. John Wiley & Sons, 2006

[40] Group, IETF Network W.: *Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) Profile.* http://www.rfc-editor.org/rfc/rfc5280.txt, 2008. – Internet RFC 5280

[41] Group, Network W.: *A Framework for Policy-based Admission Control.* http://www.rfc-editor.org/rfc/rfc2753.txt, 2000. – Internet RFC 2753

[42] Harrison, Michael A. ; Ruzzo, Walter L. ; Ullman, Jeffrey D.: Protection in operating systems. In: *Communications of the ACM (CACM)* 19 (1976), Nr. 8, S. 461–471

[43] Hielscher, Julia: *Dynamische Sicherheitspolitiken und Zugriffsentscheidungen in Credential-basierten, verteilten Systemen*, Universität Dortmund, Diplomarbeit, 2006

[44] Hilty, Manuel ; Pretschner, Alexander ; Basin, David A. ; Schaefer, Christian ; Walter, Thomas: A policy language for distributed usage control. In: *ESORICS* Bd. 4734, Springer Verlag, 2007 (Lecture Notes in Computer Science), S. 531–546

[45] IETF: *public key infrastructure (X.509).* http://www.ietf.org/html.charters/pkix-charter.html, 1998. – IETF X.509 Working Group

[46] International Organization for Standardization: *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture.* ISO 7498-2, 1989

[47] Janicke, Helge ; Cau, Antonio ; Siewe, François ; Zedan, Hussein ; Jones, Kevin: A compositional event & time-based policy model. In: *Proceedings of the 7th IEEE International Workshop on Policies for Distributed Systems and Networks*, IEEE Computer Society, 2006, S. 173–182

[48] Kang, Myong H. ; Park, Joon S. ; Froscher, Judith N.: Access control mechanisms for inter-organizational workflow. In: Ferraiolo, David F. (Hrsg.) ; Ray, Indrakshi (Hrsg.): *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, ACM Press, 2001, S. 66–74

[49] Karabulut, Yücel: *Secure mediation between strangers in cyberspace*, Universität Dortmund, Diss., 2002

[50] Koshutanski, Hristo ; Martinelli, Fabio ; Mori, Paolo ; Vaccarelli, Anna: Fine-grained and history-based access control with trust management for autonomic grid services. In: *2006 International Conference on Autonomic and Autonomous Systems (ICAS)*, IEEE Computer Society, 2006, S. 34

[51] Koshutanski, Hristo ; Massacci, Fabio: Interactive access control for web services. In: *Security and Protection In Information and Processing Systems, IFIP 18th World Computer Congress*, Kluwer Academic Publishers, 2004, S. 151–166

[52] Koshutanski, Hristo ; Massacci, Fabio: Interactive credential negotiation for stateful business processes. In: Herrmann, Peter (Hrsg.) ; Issarny, Valérie (Hrsg.) ; Shiu, Simon (Hrsg.): *Proceedings of the 3rd International Conference on Trust Management (iTrust)* Bd. 3477, Springer Verlag, 2005 (Lecture Notes in Computer Science), S. 256–272

[53] Li, Ninghui ; Wang, Qihua: Beyond separation of duty: An algebra for specifying high-level security policies. In: *ACM Conference on Computer and Communications Security*, ACM Press, 2006, S. 356–369

[54] Li, Ninghui ; Wang, Qihua: Beyond separation of duty: An algebra for specifying high-level security policies. In: *Journal of the ACM* 55 (2008), Nr. 3

[55] Medjahed, Brahim ; Bouguettaya, Athman ; Elmagarmid, Ahmed K.: Composing web services on the semantic web. In: *The International Journal on Very Large Databases* 12 (2003), Nr. 4, S. 333–351

[56] Minsky, Naftaly H.: Law governed interaction (LGI): A distributed coordination and control mechanism (An introduction, and a reference manual) / Department of Computer Science, Rutgers University. 2005. – Forschungsbericht. – http://www.moses.rutgers.edu/documentation/manual.pdf

[57] Minsky, Naftaly H. ; Ungureanu, Victoria: Law-governed interaction: A coordination and control mechanism for heterogeneous distributed systems. In: *ACM Transactions on Software Engineering and Methodology* 9 (2000), Nr. 3, S. 273–305

[58] Mowshowitz, Abbe: Social Dimensions of Office Automation. In: *Advances in Computers* 25 (1986), S. 335–404

[59] Oasis: *UDDI version 3.0.2.* http://uddi.org/pubs/uddi_v3.htm, 2005

[60] Oasis: *Web Services Business Process Execution Language, version 2.0.* http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html, 2007

[61] Park, Jaehong ; Sandhu, Ravi S.: The UCON_{ABC} usage control model. In: *ACM Transactions on Information and System Security* 7 (2004), Nr. 1, S. 128–174

[62] Pretschner, Alexander ; Hilty, Manuel ; Basin, David A.: Distributed usage control. In: *Communications of the ACM* 49 (2006), Nr. 9, S. 39–44

[63] Pretschner, Alexander ; Schütz, Florian ; Schaefer, Christian ; Walter, Thomas: Policy Evolution in Distributed Usage Control. In: *Electronic Notes in Theoretical Computer Science* 244 (2009), S. 109–123

[64] Priebe, Torsten ; Fernández, Eduardo B. ; Mehlau, Jens I. ; Pernul, Günther: A pattern system for access control. In: *Proccedings of the 18th Annual Conference on Data and Applications Security (DBSec)*, Kluwer Academic Publishers, 2004, S. 235–249

[65] Reiter, Raymond: A Logic for Default Reasoning. In: *Artificial Intelligence* 13 (1980), Nr. 1-2, S. 81–132

[66] Rivest, Ronald L. ; Lampson, Butler: SDSI – A simple distributed security infrastructure. In: *Presented at CRYPTO'96 Rumpsession*, 1996

[67] Robinson, Philip ; Karabulut, Yücel ; Haller, Jochen: Dynamic virtual organization management for service oriented enterprise applications. In: *IEEE Intl. Conf. on Collaborative Computing: Networking, Applications and Worksharing, USA*, 2005, S. 10 – 18

[68] *Kapitel* 2: Regular Languages. In: Rozenberg, Grzegorz (Hrsg.) ; Salomaa, Arto (Hrsg.): *Handbook of Formal Languages*. Bd. 1. Springer Verlag, 1997, S. 41 – 110

[69] Samarati, Pierangela ; Vimercati, Sabrina De C.: Access control: policies, models, and mechanisms. In: *International School on Foundations of Security Analysis and Design (FOSAD)* Bd. 2171, Springer Verlag, 2000 (Lecture Notes in Computer Science), S. 137–196

[70] Sandhu, Ravi S. ; Coyne, Edward J. ; Feinstein, Hal L. ; Youman, Charles E.: Role-Based Access Control Models. In: *IEEE Computer* 29 (1996), Nr. 2, S. 38–47

[71] Schneider, Fred B.: Enforceable Security Policies. In: *ACM Transactions on Information and System Security* 3 (2000), Nr. 1, S. 30–50

[72] Schwonke, Björn: *Anforderungen, Nachweise und Erweiterungen der Sicherheitseigenschaften für Secure Execution Orders (SEO) von Web-basierten Diensten*, Technische Universität Dortmund, Diplomarbeit, 2008

[73] Shamir, Adi: How to share a secret. In: *Communication of the ACM* 22 (1979), Nr. 11, S. 612–613

[74] Siewe, François: *A compositional framework for the development of secure access control systems*, De Montfort University, UK, Diss., 2005. `http://www.cse.dmu.ac.uk/~fsiewe/papers/fsiewephd.pdf`

[75] Siewe, François ; Cau, Antonio ; Zedan, Hussein: A compositional framework for access control policies enforcement. In: *Proceedings of the 2003 ACM Workshop on Formal Methods in Security Engineering*, ACM Press, 2003, S. 32–42

[76] Tai, Stefan ; Khalaf, Rania ; Mikalsen, Thomas A.: Composition of coordinated web services. In: *Proceedings of the 5th International Conference on Distributed Systems Platforms and Open Distributed Processing* Bd. 3231, Springer Verlag, 2004 (Lecture Notes in Computer Science), S. 294–310

[77] Tanenbaum, Andrew S. ; Steen, Maarten van: *Distributed Systems: Principles and Paradigms*. 2. Auflage. Prentice-Hall, Inc., 2007

[78] Telecommunication Standardization Sector: *Information technology – Open Systems Interconnection – Security frameworks for open system: Access control framework*. ITU-T X.812, 1996

[79] W3C Working Group: *EXtensible Markup Language XML*. http://www.w3.org/XML,

[80] W3C Working Group: *SOAP*. http://www.w3.org/TR/soap12-part0/, . – Technical Report

[81] W3C Working Group: *Web Services Description Language (WSDL)*. http://www.w3.org/TR/wsdl20-primer/, . – Technical Report

[82] W3C Working Group: *Web Services Architecture*. http://www.w3.org/TR/ws-arch/, 2004

[83] W3C Working Group: *Web Services Glossary*. http://www.w3.org/TR/ws-gloss/, 2004

[84] Wijesekera, Duminda ; Jajodia, Sushil: A propositional policy algebra for access control. In: *ACM Transactions on Information and System Security* 6 (2003), Nr. 2, S. 286–325

[85] William H. Davidow, Michael S. M.: *The Virtual Corporation*. HarperCollins, 1992

[86] Woo, Thomas Y. C. ; Lam, Simon S.: Authorizations in distributed systems: A new approach. In: *Journal of Computer Security* 2 (1993), Nr. 2-3, S. 107–136

[87] Yuan, Eric ; Tong, Jin: Attribute based access control (ABAC) for web services. In: *IEEE International Conference on Web Services*, IEEE Computer Society Press, 2005, S. 561 – 569

Danksagung

An dieser Stelle möchte ich all denen DANKE sagen, ohne die ich niemals ein Licht am Ende meiner Dissertation gesehen hätte.

Joachim Biskup, gilt mein ganz besonderer Dank für die fachliche und menschliche Betreuung. Ohne seine Geduld und seine Bereitschaft für zahlreiche, wertvolle Diskussionen wäre diese Arbeit niemals entstanden. Ich habe in meiner Zeit als wissenschaftliche Angestellte in seiner Arbeitsgruppe viel gelernt.
Heiko Krumm als Zweitgutachter und Peter Marwedel und Holger Schmidt, für ihre Bereitschaft als Mitglieder meiner Prüfungskommission das Promotionsverfahren zum Abschluss zu bringen.
Frau Prof. Gitta Domik, die mir mit moralischer Unterstützung und praktischen Ratschlägen geholfen hat, meine Forschung und meine Familie zumindest zeitweise unter einen Hut zu bekommen.
Jan-Hendrik Lochner, der mir in der Endphase dieser Arbeit mit Rat und Tat zur Seite gestanden hat. Ohne seinen Einsatz wäre Vieles nicht fertig geworden.
Frank Müller, der es ausgehalten hat, die Höhen und Tiefen dieser Arbeit mitzuerleben.
Barbara Sprick, die mich in meinen wissenschaftlichen Anfängen begleitet hat. Sie hat wesentlich zu meiner wissenschaftlichen, aber auch meiner persönlichen Weiterentwicklung beigetragen.
Torben Weibert, der zur richtigen Zeit die richtigen Worte gefunden hat.
Lena Wiese, für die Motivation und Zeit, die sie sich für mich und meine Arbeit genommen hat.
Unserer „Kaffee-grillen“ Runde (ehemaliger) Kollegen, die ich sonst nie kennen gelernt hätte und mit denen eine uns lieb gewonnene Tradition entstanden ist: Claudia Graute, Thomas Leineweber, Jan-Hendrik Lochner, Frank Müller, Jörg Parthe, Manuela Ritterskamp und Barbara Sprick.
Der ISSI-Arbeitsgruppe, die die perfekte Arbeitsatmosphäre am Lehrstuhl ausmacht.
Alfons & Cilli, Manuela und Sabine, die sich so oft Zeit für Jonas und Timo genommen haben, damit ich mal wieder am Schreibtisch verschwinden konnte.

Vor allem meiner lieben Familie:
Jonas und Timo, für ihre Geduld. Christoph, für seine Liebe.