Universität Hamburg
Fakultät für Mathematik,
Informatik und Naturwissenschaften

Diplomarbeit

Erweiterung einer Middleware für das Mobile Computing um nicht-funktionale Sicherheits- und Vertrauensaspekte

September 2007

Alice Winnicki

mail@alice-winnicki.de
Studiengang Informatik
Matrikel-Nr. 5525792
Fachsemester 10

Erstgutachter: Prof. Dr. W. Lamersdorf
Zweitgutachterin: Prof. Dr. C. Eschenbach

Danksagung

Ich möchte allen Personen danken, die mir während der Ausarbeitung dieser Diplomarbeit den Weg gewiesen haben und mir mit Rat und Tat zur Seite standen.
Ganz besonders danke ich meinen Tutoren Prof. Dr. Carola Eschenbach, Dr. Christian P. Kunze und Prof. Dr. Winfried Lamersdorf für ihr Interesse an meiner Arbeit und für ihre kritischen Anmerkungen und Verbesserungsvorschläge.
Ein herzlicher Dank gilt auch allen Mitarbeitern des Arbeitsbereiches *Verteilte und Informationssysteme* am *Department Informatik* für die freundliche Arbeitsatmosphäre und für die Bereitstellung eines hervorragenden Arbeitsplatzes für die Dauer der Diplomarbeit.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel wird zunächst motiviert, weshalb die Erweiterung einer Middleware für das Mobile Computing um nicht-funktionale Sicherheits- und Vertrauensaspekte erforderlich ist und welche Aufgaben und Ziele dieser Arbeit zugrunde liegen. Anschließend wird die Vorgehensweise erläutert und die Gliederung der Arbeit vorgestellt.

1.1 Motivation

Der rasante Fortschritt im Bereich der drahtlosen Kommunikation sowie der Herstellung mobiler Endgeräte führt zu einer stetig wachsenden Zahl an Teilnehmern sowohl im privaten als auch im geschäftlichen Bereich. Dabei geht die Entwicklung vom Einsatz gewöhnlicher, auf reine Sprachübertragung spezialisierter Mobiltelefone bis hin zu multimedialen Alleskönnern, die sich durch eine Übertragung mit immer höheren Datenraten zu erschwinglichen Preisen auszeichnen. Die bereitgestellten Mobilfunknetze umspannen schon jetzt den Großteil der Welt und ermöglichen eine beinahe nahtlose Erreichbarkeit von Kommunikationspartnern.

Die Integration drahtloser Kommunikationstandards wie *Bluetooth*, *IrDA* und *WLAN* in mobile Geräte ermöglicht eine direkte Anbindung an andere Endgeräte wie Laptops, Handys, PDAs und stationäre Systeme wie etwa Desktop-PCs oder WLAN-Access-Points, die wiederum Zugriff zu fest verkabelten Rechnernetzen bieten können. Durch diese Mobilität wird eine Fülle von Anwendungen denkbar, die Benutzern mobiler Geräte gestattet, von unterwegs wichtige Aufgaben zu erledigen und Zugriff auf wichtige Informationen zu erhalten, die sie sonst nur zuhause oder an ihren stationären Arbeitsplatzrechnern innerhalb des Unternehmensnetzwerks erhalten könnten. Die übertragenen Daten können dabei z.B. Firmengeheimnise, Zugangspasswörter, PINs, TANs oder Überweisungsaufträge enthalten und bei Offenlegung dieser Informationen zu großen Schäden finanzieller oder wirtschaftlicher Natur führen.

So entsteht die Frage, wie man trotz aller Mobilität, komplexen Anwendungen und gewünschter einfacher Benutzbarkeit (*Usability*) der Systeme Vertraulichkeit und Integrität der übertragenen Daten sowie eine zuverlässige Authentifizierung der Kommunikationspartner sicherstellen kann. Dies ist erforderlich, da im Prinzip jeder Benutzer in Funkreichweite Zugriff auf den Übertragungskanal Luft hat und einen potentiellen Angreifer darstellen könnte. Täglich bedienen sich Milliarden von Menschen ihrer elektronischen Hilfsmittel. Der Großteil der Benutzer ist jedoch mit der technischen Funktionsweise der benutzten Technologien nicht vertraut oder verfügt nur über eine grobe Vorstellung darüber. Die betreffenden Benutzer sind sich häufig auch nicht bewußt, welche Implikatio-

nen die Interaktionen mit ihren vernetzten mobilen Geräten bezüglich verschickter und offengelegter Informationen mit sich führen. Durch den Mangel an Verständnis dieser Implikationen resultiert auch eine gewisse Gleichgültigkeit diesen gegenüber. Abläufe werden so alltäglich, dass über Ein- und Ausgabe nicht mehr viel nachgedacht wird. So kommt es häufig vor, dass Straftäter beim Begehen einer Straftat ihr eingeschaltetes Handy bei sich führen, weil sie gar nicht wissen, dass damit genaue Bewegungsprofile erstellt werden können, oder es schlicht vergessen. Da sich mobile Geräte heute immer mehr ihrer Umgebung bewußt sind (*Context Awareness*), können große Sammlungen personenbezogener Daten entstehen, die z.B. die genauen Informationen über den Aufenthaltsort und die Identität der Kommunikationspartner dokumentieren. Diese Informationen könnten zu maßgeschneiderten Werbezwecken oder zur Überwachung des einzelnen Benutzers ausgenutzt werden. Hiermit würde gegen das im Grundgesetz verankerte Recht jedes Bürgers auf informationelle Selbstbestimmung verstossen und in seine Privatsphäre eingedrungen werden.

Die Absicherung mobiler Systeme durch Integration von Sicherheitsmechanismen und Vertrauensmodellen ist daher ein wesentlicher Aspekt bei der Entwicklung dieser Systeme. Der Grad an Sicherheit in mobilen Systemen spielt eine bedeutende Rolle dabei, ob diese Systeme breite Akzeptanz nicht nur im privaten, sondern vor allem im unternehmerischen Bereich finden. Sicherheitskritische Anwendungsbereiche sind beispielsweise die Anbindung mobiler Geräte an Unternehmensnetzwerke, mobiles Einkaufen (*M-Commerce*), mobiles *Banking* etc.

1.2 Ziele dieser Arbeit

Obwohl die facettenreiche Thematik der Sicherheit in verteilten Systemen unlängst bekannt ist und vielfältig konkrete Lösungen existieren, wirft der Faktor der Mobilität bereits als gelöst betrachtete Fragen auf diesem Gebiet wieder neu auf. Dies liegt an den besonderen Eigenschaften mobiler Systeme, die sich stark von stationären Systemen unterscheiden. So sind mobile Systeme zum einen durch eine beständige Ressourcenknappheit gekennzeichnet, zum anderen benutzen sie die Luft als Übertragungskanal, welcher somit als *Broadcast*-Medium fungiert. Darüber hinaus herrscht im Allgemeinen eine hohe Anonymität unter den Teilnehmern, da diese häufig und dynamisch wechseln können. Diese Merkmale werfen einen Schatten auf das enorme Zukunftspotential des *Mobile Computing*. Der physisch gesehen intrinsisch unsichere Übertragungskanal erfordert besondere Sicherheitsmechanismen und -richtlinien, um vor unbefugtem Mithorchen, Sammeln und Manipulieren von Daten und Informationen zu schützen. Diese Mechanismen beanspruchen jedoch Speicher- und Rechenressourcen. Da Sicherheit an sich jedoch keinen Mehrwert oder zusätzliche Funktionalität für Benutzer bietet, wird aus Kosten-, Zeit- oder Designgründen häufig an der Absicherung von mobilen Endgeräten, mobilen

Diensten und Kommunikationskanälen gespart.

Die vorliegende Arbeit ist innerhalb des Projektes DEMAC (*Distributed Environment for Mobility-Aware Computing*) entstanden. Ziel ist es, grundlegende Mechanismen wie Authentifizierung der Kommunikationspartner, Verschlüsselung der Kommunikation, den Schutz sensitiver Daten und den Aufbau von Vertrauensbeziehungen innerhalb mobiler Middleware zu ermöglichen, ohne die Kernfunktionalität mobiler Geräte, Anwendungen und Dienste zu beeinträchtigen. Dabei soll ein möglichst generisches Lösungskonzept zur Erweiterung mobiler Systeme um ausgewählte Sicherheits- und Vertrauensaspekte entwickelt werden, welches in die Middleware der Systeme einfließt und von konkreten Anwendungen und Übertragungsprotokollen abstrahiert. Dieses Konzept wird im Rahmen der DEMAC-Middleware prototypisch implementiert.

1.3 Vorgehensweise

Um ein besseres Verständnis für die Besonderheiten mobiler Systeme zu entwickeln, werden in Kapitel 2 zunächst die Grundlagen mobiler Systeme, des Mobile Computing sowie den damit eng zusammenhängenden Bereichen, wie beispielsweise Kontextbewußtsein, erläutert. Darauf aufbauend wird in Kapitel 3 die Sicherheit informationstechnischer Systeme behandelt, mit besonderem Fokus auf die Angriffsanfälligkeiten und Sicherheitsanforderungen mobiler Systeme sowie mit Fokus auf konzeptionelle Ideen und praktische Lösungsansätze, das Risiko eines Angriffs zu minimieren und einen Sicherheitsstandard zu etablieren. Da die Teilnehmer mobiler Systeme sich gegenseitig häufig fremd sind und kein oder nur wenig Vorwissen übereinander haben, zur Erreichung jeweils eigener Ziele eventuell dennoch mit verfügbaren Teilnehmern kooperieren möchten, werden in Kapitel 4 Konzepte des sogenannten *Trust Managements* besprochen, die den Aufbau und den Erhalt von Vertrauensbeziehungen modellieren und besonders für Interaktionen erforderlich sind, bei denen sensible Informationen ausgetauscht werden oder besonders sicherheitsbedürftige Transaktionen abgewickelt werden. Aus mehreren verfügbaren potentiellen Kooperationspartnern soll nach Möglichkeit der zuverlässigste und vertrauenswürdigste Partner durch seine Reputation, also durch Informationen über vergangene Interaktionen bestimmt werden können.

Mit den gewonnenen Erkenntnissen aus den Kapiteln 2, 3 und 4 werden in Kapitel 5 eigene Komponenten entwickelt, die grundlegende Sicherheits- und Vertrauensdienste innerhalb mobiler Middleware erbringen können, um diese für den Einsatz in einem sicherheitskritischen Umfeld mit Anbindung an drahtlose Netze aufzurüsten. Die Vorgehensweise bei der Implementierung des entworfenen Konzeptes aus Kapitel 5 sowie verwendete Werkzeuge werden in Kapitel 6 dokumentiert. Kapitel 7 bildet eine Zusammenfassung der vorliegenden Arbeit, in der erzielte Ergebnisse betrachtet werden und

ein Ausblick für zukünftige Arbeiten sowie ein abschließendes Fazit geliefert werden.

1.4 Anmerkungen formeller Art

In dieser Arbeit wird aus Gründen der Lesbarkeit und Verständlichkeit des Textes darauf verzichtet, stets die männliche und weibliche Form der verwendeten personenbezogenen Substantive zu nennen. Selbstverständlich ist mit „der Benutzer“ auch „die Benutzerin“ gemeint.

Worte und Fachbegriffe englischer Herkunft werden so gut wie möglich ins Deutsche übersetzt. In manchen Fällen ist jedoch keine Übersetzung möglich, die so treffend und prägnant wie der englische Begriff wäre. Unter diesen Umständen wird daher der ursprüngliche Begriff englischer Herkunft beibehalten.

2 Grundlagen mobiler Systeme

Da die vorliegende Arbeit zentral auf mobile Systeme ausgerichtet ist, werden in diesem Kapitel die grundlegenden Eigenschaften mobiler Systeme eingeführt. Dazu werden zunächst gängige Definitionen betrachtet und erklärt, um mit Hilfe dieses Vokabulars bekannte mobile Technologien und ihre Anwendungsbereiche vorzustellen. Desweiteren werden insbesondere die Beschränkungen mobiler Systeme im Vergleich zu stationären Systemen aufgezeigt und aktuelle Herausforderungen im Bereich mobiler Systeme diskutiert.

2.1 Einführung und Definitionen

Um eine gemeinsame Fachsprache für die nachfolgenden Kapitel festzulegen, werden in diesem Abschnitt wichtige Begriffe aus dem Bereich mobiler Systeme eingeführt und erläutert.

2.1.1 Pervasive Computing / Ubiquitous Computing

Die Fortschritte im Bereich mobiler Systeme und die stetige Verbesserung bei der Herstellung von Mikrochips lassen neue Visionen über die Zukunft digitaler Technologien heranwachsen. Der Wunsch nach einer umfassenden Informationsversorgung (*Anywhere-Anytime-Computing*) führt zu einem neuen Ziel: Zur Verschmelzung der physischen Umgebung des Menschen mit den vielfältigen Aspekten und Möglichkeiten leistungsfähiger Recheneinheiten. Allgegenwärtig sollen nützliche kleine Rechner und Sensoren ihre Dienste und Anwendungen mit einer größtmöglichen Transparenz für den Benutzer anbieten und ausführen. Mit Transparenz ist dabei die Verborgenheit der komplexen Architektur und der Arbeitsabläufe der eingesetzten Systeme gegenüber dem Benutzer gemeint. Die Systeme sollen so in die Umgebung mit einfließen, dass sie in den meisten Fällen nur noch unbewusst eingesetzt werden, so wie es Mark Weiser [Wei91] schon im Jahre 1991 vorhergesagt hat. Diese Visionen werden unter dem Begriff des *Pervasive Computing* oder *Ubiquitous Computing* zusammengefasst. Der fundamentale Unterschied zur traditionellen Rechnerbenutzung liegt dabei in der Art der Interaktion des Benutzers mit den Recheneinheiten und ihren Diensten, da die Arbeitsabläufe bei gewöhnlichen PCs oder Handys in der Regel vom Benutzer initiiert werden. [Sta02]

2.1.2 Mobile Computing

Das Forschungsgebiet des *Mobile Computing* ist ein Teilbereich des Ubiquitous Computing und beschäftigt sich sowohl mit den Technologien der Mobilkommunikation als auch mit den dafür vorgesehenen mobilen Endgeräten sowie mobilen Anwendungen

[Rot05]. Da das Mobile Computing also sehr facettenreich ist, werden die einzelnen Aspekte nachfolgend in eigenständigen Abschnitten behandelt. Ein zentraler Stellenwert kommt dabei auch der Forderung nach Mobilität zu.

2.1.3 Mobilität

Unter dem Begriff der Mobilität versteht man das Bedürfnis nach umfassender Informationsversorgung an jedem Ort und zu jeder Zeit mit Hilfe vielfältiger Kommunikationsmöglichkeiten. Früher jedoch standen sich die Forderungen nach Mobilität und aktueller Verfügbarkeit von Informationen einander wechselseitig ausschließend gegenüber. So hatte man entweder ein isoliertes tragbares Gerät ohne Anbindung an weitere Informations- und Kommunikationssysteme oder aber man hatte Zugriff auf ein stationäres Informationssystem und war dadurch nicht mehr mobil. Mobilität geht also über die bloße Tragbarkeit eines Endgerätes hinaus und wird ausschlaggebend durch die Eigenschaften des mobilen Endgerätes sowie von der Verfügbarkeit der benutzten Kommunikationsmechanismen bestimmt. [DH95]

2.1.4 Arten der Mobilität

Roth [Rot05] unterscheidet im Allgemeinen drei Arten von Mobilität bei der Mobilkommunikation (vgl. Abbildung 2.1):

- Endgerätemobilität
  Die Endgerätemobilität erhält die ununterbrochene Vernetzung eines Endgerätes auch dann aufrecht, wenn dieses Gerät räumlich bewegt wird. So wird es beispielsweise dem Benutzer eines Mobiltelefons ermöglicht, die Dienste der Sprachübertragung in Anspruch zu nehmen, während er selbst große Entfernungen zurücklegt. Um Endgerätemobilität zu gewährleisten werden drahtlose Netze benötigt. [Rot05]

- Benutzermobilität
  Bei der Benutzermobilität kann der Benutzer beliebige unpersönliche Endgeräte, die in seiner Umgebung gerade verfügbar sind, verwenden, um bestimmte Dienste in Anspruch nehmen zu können. Das setzt voraus, dass der Benutzer eindeutig identifiziert und authentifiziert werden kann. Dies kann beispielsweise durch den Besitz einer persönlichen SIM-Karte (*Smartcard*), durch das Wissen um ein Passwort oder durch biometrische Merkmale geschehen. [Rot05]

- Dienstmobilität
  Die Dienstmobilität versucht, die Verfügbarkeit bestimmter Dienste an jedem Ort und zu jeder Zeit zu gewährleisten, so dass ein Benutzer bei Bedarf immer darauf zugreifen kann. Dies wünscht man sich beispielsweise für das Abrufen persönlicher E-Mail-Nachrichten. [Rot05]

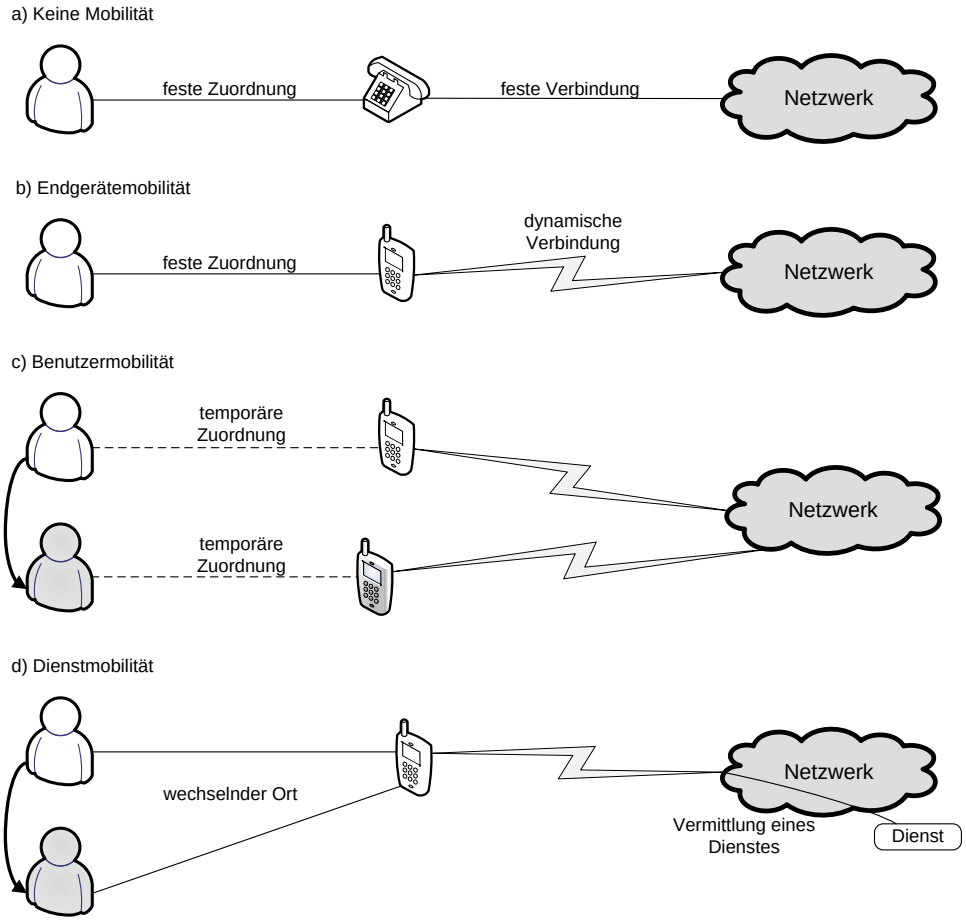

Abbildung 2.1: Arten der Mobilität (nach [Rot05, S. 8])

2.1.5 Eigenschaften mobiler Endgeräte

Die Eigenschaften mobiler Endgeräte bestimmen die Eignung dieser Geräte für verschiedene (mobile) Anwendungen. Da diese Eigenschaften in den meisten Fällen deutlich eingeschränkter sind als Eigenschaften stationärer Rechner, bieten mobile Anwendungen dementsprechend meist weniger Funktionen. Die Beschränkungen mobiler Endgeräte wirken sich u.a. auf folgende Eigenschaften aus [ED06]:

- Größe des Bildschirmes
- Speicherkapazität
- Prozessorleistung
- unterstützte Kommunikationsstandards
- Datenrate
- Energieversorgung

- Netzwerkanbindung (Konnektivität)

Die jeweiligen Ausprägungen der hier genannten Eigenschaften bei mobilen Geräten bilden auch Kriterien zu einer Klassifizierung der Geräte in Mobiltelefone, *Smartphones*, PDAs (*Personal Digital Assistants*), Notebooks etc. Durch die Beschränkungen mobiler Endgeräte in den genannten Merkmalen ergeben sich intrinsische Einschränkungen für das Mobile Computing, die im Folgenden erläutert werden.

- Ressourcenarmut
  Für eine gegebene Technologie und die damit verbundenen Kosten führen Überlegungen und (Entwurfs-) Entscheidungen in Bezug auf Gewicht, Größe und Ergonomie eines Rechensystems meist zu Einbußen bei den internen Ressourcen wie der Prozessorgeschwindigkeit, des Arbeitsspeichers und der Festplattenkapazität. Obwohl mobile Geräte in ihrer Leistungsfähigkeit und ihrem Funktionsumfang stetig verbessert werden, so sind sie dennoch als ressourcenarm gegenüber stationären Rechnern zu betrachten, bei denen Größe und Gewicht eine untergeordnete Rolle spielen. [Sat96]

- Anfälligkeit für Sicherheitsrisiken
  Aufgrund ihrer Portabilität sind mobile Geräte anfällig für Verlust und Diebstahl. Es ist viel wahrscheinlicher, dass ein mobiles Gerät seinem Eigentümer unterwegs entwendet wird oder verloren geht, als dass ihm sein Arbeitsplatzrechner in seinem abgeschlossenen Büro gestohlen wird. Auch der Kommunikationskanal *Luft* birgt große Risiken, da es ein *Broadcast*-Medium ist und im Prinzip von jedem abgehört werden kann. Es müssen daher geeignete Gegenmaßnahmen bereitgestellt werden. [Sat96]

- Unzuverlässigkeit mobiler Verbindungen
  Mobile Geräte bedienen sich drahtloser Netze zur Anbindung an ein Netzwerk. Drahtlose Netze sind jedoch nicht flächendeckend vorhanden und können so zur Isolierung von Geräten durch Funklöcher führen. Insbesondere ist eine Versorgung auf dem Lande, in Gebäuden mit dicken Mauern oder unter der Erde (z.B. im U-Bahn-Tunnel) häufig nicht gegeben. Im Vergleich zu drahtgebundenen Netzen werden in drahtlosen Netzen je nach Standard nur geringe Datenraten zur Verfügung gestellt, die durch Störungen des Funkverkehrs (z.B. auf dem Flughafen) zusätzlich verschlechtert werden können. Drahtlose Netze sind in Bezug auf Datenrate und Verfügbarkeit also unzuverlässiger als drahtgebundene Netze. [Sat96]

- Endlicher Energievorrat
  Mobile Geräte verfügen über endliche Energievorräte, die stets wieder neu aufgeladen werden müssen. Es ist zwar anzunehmen, dass im Bereich der Batterien- und Akkumulator-Technologie in der Zukunft weitere Fortschritte erzielt werden, dennoch wird der sorgfältige und sparsame Einsatz des Energievorrats mobiler Geräte

> als wichtige einschränkende Bedingung im Mobile Computing bestehen bleiben. Dies ist unter anderem in der Annahme begründet, dass bei einem zukünftigen Einsatz energiereicherer Batterien auch energiebedürftigere Hardwarekomponenten und rechenaufwändigere Anwendungen verwendet werden würden. [Sat96]

Zheng und Ni [ZN06] unterscheiden zwei verschiedene Betriebsmodi mobiler Endgeräte im Mobile Computing:

- vernetzter Betrieb (*connected mode*)
- isolierter Betrieb (*disconnected mode*)

Bei isoliertem Betrieb eines mobilen Gerätes erfolgt der Zugriff auf Informationen lokal auf dem Gerät selbst, wie z.B. bei der Nutzung eines Adressbuches oder eines Terminplaners auf einem PDA. Dagegen stellt ein mobiles Gerät bei vernetztem Betrieb mehrere Zugriffsmöglichkeiten auf drahtlose oder drahtgebundene Netzwerke bereit. Da die mobilen, drahtlosen Technologien immer besser werden und höhere Datenraten zu günstigeren Preisen ermöglichen, wird eine Reihe neuartiger mobiler Geräte und Anwendungen ermöglicht, die dem Wunsch nach ubiquitärem mobilen Zugang zu Informationen und Diensten immer mehr nachkommen. [ZN06]

2.1.6 Der Begriff des Smartphone

Der Begriff *Smartphone* ist ursprünglich aufgekommen, als eine neue Klasse mobiler Telefone auf den Markt kam, deren Funktionsumfang über die traditionelle Sprachübertragung und das Versenden von kurzen Textnachrichten (*Short-Message-Service, SMS*) hinaus ging und die erstmals über signifikant schnellere Prozessoren verfügte und bereits einen ersten mobilen Datenzugriff durch paketorientierte Datenübertragung z.B. mittels GPRS (*General Packet Radio Service*) ermöglichte. Ein Smartphone bietet die Möglichkeit, persönliche Informationen und Aufgaben mit Hilfe digitaler Kalender, Adressbücher oder Terminplaner (*Personal Information Management*) zu verwalten und zu diesem Zwecke drahtlos zu kommunizieren. Allgemein kann man sagen, dass ein Smartphone ein kleiner, multi-funktionaler und vernetzter Rechner in der Form und Größe eines Mobiltelefons ist. [ED06]
Zheng und Ni [ZN06] sind der Auffassung, der schnelle Fortschritt im Bereich mobiler Systeme führe zunehmend zur Vereinigung dreier Sektoren, die historisch betrachtet bislang eher disjunkt und wenig interoperabel gewesen sind. Hierbei handelt es sich um die Sektoren der Kommunikation, der elektronischen Datenverarbeitung (*Computing*) und der Unterhaltungselektronik (*Consumer Electronics*). Ziel ist es unter anderem auch, koexistierende heterogene Netzinfrastrukturen interoperabel zu machen, um eine flächendeckende Versorgung, einen nahtlosen Übergang und ein gegenseitiges Weiterreichen von Teilnehmern (*Handover*) zu ermöglichen. Dieser Vorgang soll für den Teilnehmer selbst möglichst transparent bleiben und kann sowohl drahtlose als auch drahtgebundene Zwischenstationen enthalten. Das Smartphone, welches als universelles mobiles

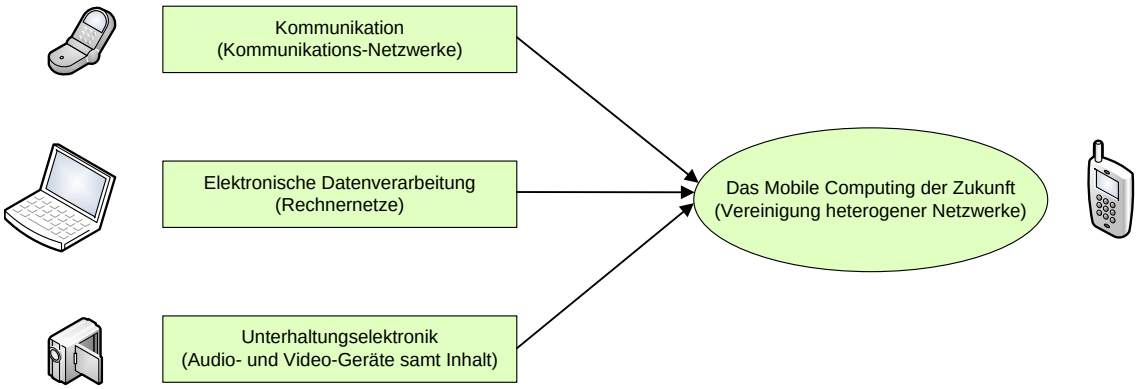

Abbildung 2.2: Digitale Vereinigung (nach [ZN06, S. 7])

Datenendgerät fungiert und durch den (fast) allgegenwärtigen Zugang zu drahtlosen Netzen in seinem Funktionsumfang erweitert wird, ist Produkt dieser Vereinigung. Veranschaulicht wird diese *digitale Vereinigung* in Abbildung 2.2.

2.1.7 Drahtgebundenes und drahtloses Mobile Computing

Häufig wird die Bedeutung der beiden Adjektive *mobil* und *drahtlos* durch eine irreführende Verwendung und Werbung in den Medien durcheinander gebracht oder aber die Begriffe werden als Synonyme betrachtet. Man unterscheidet jedoch zwischen dem drahtgebundenen und dem drahtlosen Mobile Computing. Bei Ersterem wird der Zugang zu einem Netzwerk von dem mobilen Gerät aus ermöglicht, indem dieses an ein drahtgebundenes Netz angeschlossen wird. So können beispielsweise berufstätige Personen ihren Laptop vom Arbeitsplatz zu einer auswärtigen Konferenz mitbringen und sich dort per Kabel an das lokale Netzwerk des Veranstalters anschliessen oder aber das DSL-Internetangebot eines Hotels nutzen. Da es eine Vielfalt an drahtlosen Technologien gibt, wie z.B. Mobilfunknetze (GSM, UMTS), drahtlose lokale Netze (WLAN), Bluetooth und Infrarot, die einem stetigen Fortschritt unterliegen, wird es einem Benutzer ermöglicht, ein Maximum an Mobilität zu erreichen. Das bedeutet, dass Benutzer sich mit ihren mobilen Geräten nicht mehr an einem fixen Punkt aufhalten müssen, wenn sie ihre Aufgaben erledigen wollen, sondern dass sie diese Aufgaben nunmehr von unterwegs an jedem Ort und zu jeder Zeit ausführen können (*Anywhere and Anytime Computing*). Dies setzt natürlich voraus, dass drahtlose Netze flächendeckend verfügbar sind. [ZN06], [ED06]

2.1.8 Drahtlose Kommunikation

Wie das Adjektiv bereits besagt, findet man bei drahtlos kommunizierenden Geräten keine Kabel, die für die Kommunikation vorgesehen sind. Vielmehr werden zahlreiche verschiedene Möglichkeiten und Standards der Funkübertragung genutzt, um zwei oder mehrere mobile Geräte oder aber ein Gerät und eine hybride Infrastruktur miteinander

zu verbinden. Das Mobile Computing bedient sich dieser Funktechnologien, um jede Art der Mobilität (vgl. Abschnitt 2.1.4) zu ermöglichen. Dabei bedeutet Drahtlosigkeit aber nicht immer gleichzeitig Mobilität. So kann drahtlose Kommunikation als Kabelersatz in einem stationären Rechnernetz benutzt werden, falls eine Verlegung von Kabeln zu viel Aufwand verursachen würde. Auch Funktechnologien mit kurzer Funkreichweite oder die eine direkte Sichtbarkeit (*Direct Line of Sight*) erfordern, bieten eine geringe Endgerätemobilität, da die Geräte während einer Übertragung wenig bewegt werden dürfen und der Benutzer sich somit auch nicht weit mit ihnen entfernen darf. Zusammenfassend kann man sagen, dass sich das Gebiet der drahtlosen Kommunikation eher mit der Signalisierung und Übertragung von Funksignalen auf der Bitübertragungs- und der Datensicherungsschicht beschäftigt, während sich das Mobile Computing der drahtlosen Kommunikation bedient, um darauf komplexe Anwendungen und Dienste aufbauen zu können. [ZN06], [Rot05]

2.2 Mobile Technologien und ihre Anwendungsbereiche

Da es mittlerweile eine Fülle an mobilen Umgebungen gibt, die sich unterschiedlicher Technologien bedienen und für verschiedene Anwendungsbereiche geeignet sind, soll hier ein kurzer Überblick darüber vermittelt werden.

2.2.1 Zellulare Mobilfunknetze und Mobilfunkgenerationen

Mobilfunknetze sollen möglichst flächendeckend sein, damit die Erreichbarkeit mobiler Teilnehmer innerhalb eines ganzen Landes oder über Ländergrenzen hinweg gewährleistet werden kann. Da die räumliche Ausbreitung elektromagnetischer Wellen jedoch begrenzt ist, führte dies zur Organisation dieser Netze in Zell-Infrastrukturen, in der je eine Basis-Sendestation für je eine ganze Zelle verantwortlich ist, und Teilnehmer an nächstgelegene Basisstationen weitergereicht werden können, sofern die Teilnehmer die aktuelle Zelle verlassen. Mobilfunknetze werden daher häufig auch als zellulare Netze bezeichnet. [Rot05]
Die Geschichte der Mobiltelefonie mit Hilfe von Mobilfunknetzen ist lang. Unter dem Mobilfunk der ersten Generation (1G) fasst man die analogen A-, B- und C-Netze zusammen. Die Ära dieser ersten Generation begann in Deutschland mit der Inbetriebnahme des A-Netzes im Jahre 1958 und endete im Jahre 2001 mit der Abschaltung des C-Netzes. Diese ersten Netze waren jedoch nicht flächendeckend nutzbar. Abgelöst wurde die erste Generation von den digitalen GSM-Netzen (*Global System for Mobile Communications*) der zweiten Generation (2G). Dies geschah zeitlich ineinander übergreifend, da der Wandel von der analogen zur digitalen Technik bereits 1992 mit Einführung der D-Netze vollzogen wurde, während das B- und das C-Netz noch in Betrieb waren. Eine Weiterentwicklung des GSM-Standards führte 1994 zur Einführung der E-Netze, die im höheren Frequenzbereich als die D-Netze arbeiten. Da GPRS (*General Packet Radio Service*) eine

	802.11	**802.11a**	**802.11b**	**802.11g**
Frequenzbereich	2,4 GHz	5 GHz	2,4 GHz	2,4 GHz
Datenrate	2 Mbit/s	54 Mbit/s	11 Mbit/s	54 Mbit/s

Tabelle 2.1: Die meistgenutzten WLAN-Standards

Erweiterung des GSM-Standards um eine paketorientierte Datenübertragung ist, wird es manchmal auch als 2.5G bezeichnet. Die dritte Generation der Mobilfunknetze (*3G*) wird vom UMTS-Standard (*Universal Mobile Telecommunications System*) geprägt, der sich unter anderem durch höhere Datenraten auszeichnet, die zur Erbringung multimedialer Dienste genutzt werden können. [Eck06]

2.2.2 Drahtlose lokale Netze nach IEEE 802.11 (WLAN)

Ein WLAN (*Wireless Local Area Network*) ist ein drahtloses Netz nach dem *IEEE 802.11*-Standard, welches sich zur Übertragung von Daten der Funktechnologie bedient. Aus diesem Grunde wird es manchmal auch als lokales Funknetz bezeichnet. WLANs werden häufig zur bequemen Vernetzung von Rechnern innerhalb von Gebäuden benutzt, oder aber auch zur Überbrückung von Entfernungen zwischen Gebäuden, wie z.B. in einer Universität oder innerhalb eines Firmengeländes. [Eck03]
Aus Gründen der Interoperabilität wurde für die WLAN-Technologie eine Reihe von Standards entwickelt. Diese unterscheiden sich hauptsächlich in der maximalen Datenrate und ihrer Bandbreite[1]. Einen Überblick über die meist genutzten Standards verschafft Tabelle 2.1. Ein Grund, warum WLANs so populär sind, ist, dass die meisten WLAN-Standards auf lizenzfreien Frequenzen, dem sogenannte ISM-Frequenzband (*Industrial, Scientific, Medical*), funkt. Es handelt sich hierbei um Frequenzen, die im Bereich von 2,4 bis 2,48 GHz angesiedelt sind. Da die Installation eines WLANs im Allgemeinen kostengünstig und mit einer großen Signalausbreitung und hohen Datenraten verbunden ist, gibt es vielfältige Einsatzgebiete dafür. So gibt es beispielsweise eine große Verbreitung sogenannter *Hot-Spots*, um eine drahtlose Internetanbindung auf öffentlichen Plätzen wie Bahnhöfen oder Flughäfen zu ermöglichen. [Eck03]

2.2.3 Wireless Personal Area Network (WPAN)

Die Bezeichnung WPAN fasst eine Sammlung von Kurzstrecken-Funktechnologien zusammen, die speziell zur Vernetzung kleinerer Geräte wie Handys, Digitalkameras, PDAs, drahtlosen Kopfhörern etc. konzipiert worden sind und die reduzierten Fähigkeiten dieser Geräte im Vergleich zu stationären Rechnern oder Notebooks berücksichtigen. [Rot05]
Die Namensgebung *Personal* resultiert dabei aus der Tatsache, dass kommunizierende Geräte in einem WPAN meist eine geringe räumliche Distanz von maximal einigen wenigen Metern haben und von einer einzelnen Person bedient werden können. Während

[1] gemeint ist das Frequenzband

bei WLANs prinzipiell jedes teilnehmende Gerät mit jedem anderen WLAN-Teilnehmer kommunizieren kann (*Multipoint-to-Multipoint*) ist es innerhalb eines WPAN-Netzes in der Regel so, dass nur ein ausgezeichnetes Gerät mit mehr als einem anderem Gerät kommunizieren kann (*Point-to-Multipoint*). Meist werden sogar reine Punkt-zu-Punkt-Verbindungen benutzt. Zu den WPANs zählt man unter anderem folgende Funkstandards: [Rot05]

- IrDA (*Infrared Data Association*)
  IrDA ermöglicht den Austausch von Daten mittels infrarotem Licht über kurze Strecken unter Verwendung einer Infrarot-Schnittstelle. IrDA ist somit ein Vertreter der optischen Datenübertragung. Da Infrarotlicht den Gesetzen der Strahlenoptik unterliegt und massive Gegenstände nicht durchdringen kann, erfordert die drahtlose Infrarotkommunikation eine Sichtverbindung zwischen den Kommunikationspartnern und hat daher nur eine begrenzte Reichweite. Die Spezifikation sieht nur 100 cm vor. Dadurch kommt dem Standard jedoch eine intrinsische Abhörsicherheit zugute. [Abd06]

- Bluetooth wurde von der Firma Ericsson ursprünglich dazu entwickelt, Kabelverbindungen zwischen tragbaren Geräten zu ersetzen. Das Ziel änderte sich jedoch im Laufe der Zeit dahingehend, dass auch Geräte, die am Körper getragen werden oder in die Kleidung integriert sind, untereinander vernetzt werden können. Eine Infrarotverbindung kam dafür wegen der schlechten Durchdringung von beispielsweise Kleidung nicht in Frage. Bluetooth bietet eine drahtlose Schnittstelle, über die sowohl mobile Kleingeräte wie Mobiltelefone und PDAs als auch Computer und Peripheriegeräte miteinander kommunizieren können. Bluetooth wurde als IEEE 802.15.1 standardisiert. [Abd06]

2.3 Herausforderungen im Mobile Computing

Die fortschreitende Entwicklung mobiler drahtloser Technologien ermöglicht das Aufkommen einer Reihe neuartiger Anwendungen und Dienste, um den Bedarf von Unternehmen an kostengünstigen, leistungsfähigen und robusten Arbeitsabläufen zu decken und um ein vielfältiges Angebot für den privaten Benutzer bereitzustellen. Diese Entwicklungen werfen jedoch ihrerseits Herausforderungen und Schwierigkeiten auf, die als Forschungsgegenstand adressiert und gelöst werden müssen. Diese Herausforderungen werden im Folgenden näher betrachtet.

2.3.1 Mobile Lokalisierung und ortsbasierte Dienste

Durch das Aufkommen des Mobile Computing gewinnt der physische Aufenthaltsort von Entitäten enorm an Bedeutung. Fragen wie

- Wo befinde ich mich?

- Was passiert gerade an den Orten, die mich interessieren?
- Wie gelange ich zu einem bestimmten Ort?
- Wo befinden sich andere Entitäten (Personen, Objekte), zu denen ich Zugang erhalten möchte?
- Welche Informationen sind in der näheren Umgebung verfügbar?

können nunmehr in mobilen Anwendungen und Diensten nützlich verarbeitet werden. Dies erfordert jedoch eine Zusammenarbeit im Bereich der mobilen Positionsbestimmung, der Kursverfolgung und einem Zugriff auf geografische Informationen sowohl innerhalb eines Gebäudes, als auch im Freien. [ZN06]

2.3.2 Context-Aware Computing

Anwendungen und Dienste, die Informationen des Ortes verarbeiten, um selektierend individuelle Lösungen zu erstellen, fallen in den Bereich konzextbezogener Systeme (*Context-Aware Computing*). Die Schwierigkeiten im Bereich des Context-Aware Computing liegen vor allem im Auffinden und Messen von Kontextinformationen und bei der Modellierung und Interpretation der Kontextdaten. [RBB03]
Mobile Anwendungen und Geräte werden zunehmend kontextbewußt und kontextbezogen gestaltet. Eine Anwendung ist kontextbezogen, wenn ihr Verhalten durch Kontextinformationen beeinflusst werden kann. Dies setzt voraus, dass das Gerät, auf welchem die Anwendung betrieben wird, in der Lage ist, Kontextinformationen zu sammeln. Die Sammlung von Kontextinformationen kann durch physikalische Sensoren oder durch den Austausch von Informationen unter verschiedenen Geräten erfolgen. [RBB03]
Kontextinformationen charakterisieren die Situation, in der sich eine Entität befindet. Eine Entität kann hierbei eine Person, einen Ort oder ein Objekt bezeichnen, welches relevant für die Interaktion zwischen einem Benutzer und einer Anwendung ist. Dies schließt den Benutzer selbst und die Anwendung mit der er interagiert als Entität mit ein. Besonders wichtig ist dabei die Möglichkeit, auf Veränderungen im Kontext reagieren zu können (kontextbezogene Anpassungsfähigkeit) und übergeordnete Rückschlüsse aus einer Menge von Kontextinformationen zu ziehen. [ADB+99]
In seiner Diplomarbeit untersucht Turjalei [Tur06] die hier genannten Punkte genauer, um ein Konzept für die Integration von *Context-Awareness* in eine Middleware für mobile Systeme zu entwickeln. Die Realisierung erfolgt für die dieser Arbeit ebenfalls zugrundeliegende DEMAC-Middleware, die in Abschnitt 2.4 vorgestellt wird.
Die Sammlung von Kontextinformationen kann zu großen Datenbeständen führen, die schutzbedürftige Informationen wie beispielsweise personenbezogene Angaben oder Aufenthaltsorte enthalten. Die Offenlegung dieser Datenbestände an unberechtigte Personen stellt einen Verlust der Privatsphäre dar. In kontextbezogenen Systemen ist das Thema

Datenschutz und Privatsphäre daher besonders wichtig und erfordert geeignete Mechanismen zur vertraulichen Übermittlung von Kontextinformationen und zur Zugriffskontrolle darauf.

2.3.3 Drahtlose Sensornetze

Sensoren sind im Allgemeinen kleine, leichte Geräte mit wenig Energieverbrauch, die bestimmte Felder und Kräfte der physikalischen Welt zu messen vermögen und in der Lage sind, Stimuli wie Bewegung, Hitze, Licht, Geräusche und Druck in numerische Werte zu übersetzen. Mit Hilfe der Funktechnologie kann eine Vielzahl von drahtlosen Sensorknoten zu drahtlosen Sensornetzwerken zusammengeschlossen werden, die sich gegenseitig über Änderungen der gemessenen Werte informieren können. Drahtlose Sensornetzwerke können besonders als Frühwarnsysteme zur Einleitung von Katastrophenschutzmaßnahmen nützlich sein. Sie erweitern kontextbezogene Systeme um physische Informationen aus der Umgebung. [ZN06]

2.3.4 Mobile Ad-Hoc-Netze

Ein *Ad-Hoc*-Netz besteht aus einer selbstorganisierten Gruppe von Knoten ohne eine feste Infrastruktur, die sich zum Zwecke der Kommunikation und der Kooperation zusammenschließen. Jeder Knoten in einem Ad-Hoc-Netz verläßt sich dabei auf seine benachbarten Knoten, um mit Teilnehmern zu kommunizieren, die nicht unmittelbar erreichbar sind. Falls die Knoten sich ständig frei bewegen können, so kann sich die Topologie des Netzes dynamisch verändern. Man spricht dann von einem mobilen Ad-Hoc-Netz (*Mobile Ad Hoc Network*, MANET). Die möglichen Anwendungsfelder für eine Ad-Hoc-Kommunikation und -Kooperation zwischen mobilen Geräten liegen beispielsweise auf Schlachtfeldern während eines Krieges zur Aufspürung verletzter Soldaten, bei Katastrophenschutzmaßnahmen zur rechtzeitigen Warnung, falls kritische Grenzwerte bei Sensormessungen überschritten werden usw. [BCG04]
Die Abwesenheit einer festen Infrastruktur in Ad-Hoc-Netzen erfordert eine dynamische Netzwerkkonfiguration, das Auffinden von Diensten und ein *Multihop-Routing*. Da ein Knoten jederzeit ein Ad-Hoc-Netz verlassen oder ihm beitreten kann, müssen die Informationen über die Konfiguration in regelmäßigen Abständen aktualisiert werden. Jeder Knoten muss hierzu über eine zuverlässige Prozedur verfügen, um seinen Kontext wie benachbarte Knoten, Ressourcen und Dienste möglichst schnell erschließen zu können. Um auf Ressourcen und Diense zugreifen zu können, die ausserhalb der eigenen Reichweite liegen sind zuverlässige (Multihop)-Routingalgorithmen notwendig [BCG04].

2.3.5 Sicherheit, Privatsphäre und Vertrauen

Da das Mobile Computing eine immer stärker werdende Anwendung in allen Bereichen des alltäglichen Lebens findet, gewinnen Aspekte der Sicherheit, der Privatsphä-

re und des Vertrauens zunehmend an Bedeutung. Ohne entsprechende Sicherheitsmechanismen kann die Zuverlässigkeit von Anwendungen und Diensten sowie die Vertrauenswürdigkeit der Kooperationspartner nicht gewährleistet und Risiken somit nicht ausgeschlossen werden. Da in mobilen Systemen mit ihren vielfältigen Nutzungsmöglichkeiten große Datenmengen mit dem Bedarf an Geheimhaltung entstehen können, ist es daher eine große Herausforderung, den Schutz dieser Daten herzustellen. Dies kann durch Verschlüsselung dieser Daten auf dem Speichermedium oder aber auf dem Kommunikationsmedium geschehen. Desweiteren muss der Zugriff auf die Daten und weitere Ressourcen durch Authentifizierung und Authorisierung geregelt werden, was besonders bei gewünschter Unterstützung aller drei Arten von Mobilität (vgl. Abschnitt 2.1.4) erforderlich ist. Zudem ist zu beachten, dass die Integration und Ausführung von Sicherheitsmechanismen selbst Ressourcen beansprucht. Es muss also stets ein Gleichgewicht zwischen den verfügbaren Ressourcen, der Sicherheitsbedürftigkeit der Anwendung und ihrer Daten sowie der Ressourcenbeanspruchung der Sicherheitsmechanismen hergestellt werden. Dies gestaltet sich in der Praxis häufig als schwierig. [Sta02], [Eck06]

Laut Satyanarayanan [Sat03] existiert zwischen der Gewährleistung von Sicherheit, Privatsphäre und Vertrauen in mobilen Systemen und dem Mobile Computing ein grundsätzlicher Konflikt: Das Mobile Computing hat nämlich zum Ziel, alltäglich, allgegenwärtig und für jeden ohne grosse Vorkenntnisse nutzbar zu sein (vgl. Abschnitt 2.1.1). Die mobilen Systeme sollen möglichst unabhängig voneinander und dennoch interoperabel sein, möglichst viel Funktionalität abdecken, und bei Bedarf Dienste anderer Systeme drahtlos in Anspruch nehmen. Dieser Ablauf soll für den Benutzer jedoch größtenteils transparent (verdeckt) ablaufen. Er soll nur die nötigsten Interaktionen vornehmen und Informationen eingeben. Seine Aufmerksamkeit auf andere Tätigkeiten soll durch den Ablauf nicht abgelenkt werden. Dazu bedarf das System jedoch vieler Informationen über den Benutzer selbst, seine Vorlieben, seinen aktuellen Aufenthaltsort etc. Die Problematik liegt darin, diese Informationen nur berechtigten Entitäten zur Verfügung zu stellen und vor unberechtigtem Zugriff und Veränderung zu schützen. [Sat03]
Man unterscheidet zwei Möglichkeiten der Kontrolle über die persönlichen Daten eines Benutzers:

- Kontrolle durch den Benutzer selbst, da seine personenbezogenen Daten in seinem eigenen Einflussbereich gespeichert sind und er den Zugriff darauf restriktieren kann. [Sat03]
- Kontrolle durch dritte Parteien, sobald personenbezogene Daten eines Benutzers an diese übermittelt worden sind und somit den Einflussbereich des Benutzers verlassen. Aus technischer Sicht hat er dann keine Kontrolle mehr über diese Daten, allerdings sollte dennoch geregelt sein, dass diese Daten nicht unberechtigt verändert oder an andere weitergegeben werden dürfen. Dies kann einerseits gesetzlich z.B. durch das Bundesdatenschutzgesetz (BDSG) geregelt werden, andererseits

werden jedoch auch Mechanismen und Standards wie beispielsweise P3P zur technischen Realisierung benötigt.

2.3.6 Autonomie und Anpassungsfähigkeit mobiler Geräte

Mobile Systeme sind verteilte Systeme. Der Faktor der Mobilität erhöht jedoch die Spannungen zwischen Autonomie und gegenseitiger Abhängigkeit, die charakteristisch für alle verteilten Systeme sind. Die Ressourcenknappheit mobiler Geräte und ihre allgemein geringere Robustheit könnte dazu verleiten, den Einsatz eines stationären (zentralen) Servers der Eigenständigkeit der Geräte vorzuziehen. Für eine weitgehende Autonomie mobiler Geräte und dezentralisierte Netze spricht jedoch einerseits die Notwendigkeit, mit unzuverlässiger Konnektivität und begrenzten Energieressourcen zurecht kommen zu müssen, und andererseits die Vorteile, die mobile Ad-Hoc-Netze zu bieten haben. Eine gewisse Abhängigkeit von anderen Teilnehmern bleibt aber auch in MANETs bestehen, wenn man zum Erreichen eigener Ziele auf die Kooperation mit anderen Entitäten angewiesen ist. Als Beispiel seien das Routing oder Reputationssysteme in MANETs genannt. [ZN06], [BCG04]
Je größer die Eigenständigkeit mobiler Geräte und ihre Unabhängigkeit von zentralen Systemenkomponenten ist, desto besser können die drei Arten der Mobilität unterstützt werden. Ansätze im Mobile Computing müssen die genannten Spannungen zwischen Autonomie und gegenseitiger Abhängigkeit zu einem Gleichgewicht bringen. Da die Umstände und der Kontext eines mobilen Gerätes ständigen Veränderungen unterworfen sein können, kann dieses Gleichgewicht nicht statisch festgelegt werden. Das mobile Gerät muss daher dynamisch und kontextsensitiv reagieren können; es muss anpassungsfähig sein. [BCG04]

2.4 Vorstellung des DEMAC-Projektes

Um die natürliche Ressourcenknappheit und den damit verbundenen geringeren Funktionsumfang mobiler Geräte zu überwinden und damit die Ausführung langlebiger komplexer Aufgaben und Dienste zu ermöglichen wurde die DEMAC-Middleware entwickelt, die eine explizite Unterstützung für mobile migrierende Prozesse und ihre verteilte Ausführung durch verschiedene Knoten im Netz realisiert. Die DEMAC-Middleware ist daher eine Platform für kontextbezogene mobile Anwendungen, die die prozessorientierte Kooperation zwischen mobilen Geräten ermöglicht. Durch die Auswertung von Kontextinformationen wird die Isolierung einzelner mobiler Geräte aufgebrochen und die Erschliessung weiterer Ressourcen und Dienste in der Umgebung erreicht. Die Fähigkeit mobiler Geräte, sich schnell an dynamisch ändernde Umgebungen anpassen zu können, wird als Kontextadaptivität bezeichnet und ist eines der wichtigsten Kernpunkte im Bereich des Pervasive Computing. [KZL07]

2.4.1 Mobile Prozesse

Im Mobile Computing ist es eine besondere Herausforderung, einen Zusammenschluss mobiler Einheiten zu Infrastrukturen zu ermöglichen, in denen die mobilen Einheiten miteinander kommunizieren, um bestimmte Aufgaben gemeinsam und verteilt zu erfüllen. Die mobilen Geräte sollen sich hierzu bei Bedarf spontan mit anderen Systemen verbinden können, um deren erweiterte Ressourcen und Dienste in Anspruch zu nehmen. Dabei soll es sich bei diesen Systemen nicht nur um stationäre Geräte und Netzwerke, sondern vor allem um mobile Geräte und drahtlose Netze handeln. Dies hat den Vorteil, dass die Benutzer ihre Aufgaben und Anwendungen ortsunabhängig mit einer zuverlässigen Verfügbarkeit der benötigten Dienste ausführen lassen können. Um diese verteilte Ausführung genauer zu charakterisieren wurde die Idee der Mobilen Prozesse entwickelt.

> *"A mobile process is a sequence of (remote) services which may last over a longer period of time and span several devices during its execution. The results of the process are the effects the initiator expects from it."* [KZL06, S. 2]

Mobile Prozesse können also als eine Sequenz von miteinander in Beziehung stehenden lokalen oder entfernten Diensten angesehen werden, die langlebig sind und verteilt durch mehrere Geräte erbracht werden können. Mobile Prozesse bilden daher die Grundlage für die Ausführung komplexer und anwendungsbezogener Aufgaben, die von einem Benutzer auf einem mobilen Gerät initiiert werden können. [KZL07]
Mobile Prozesse müssen dabei in einer Prozessbeschreibungssprache formuliert werden, die den speziellen Anforderungen mobiler Systeme gerecht wird, da sich diese Anforderungen erheblich von den Anforderungen stationärer Systeme unterscheiden (vgl. Abschnitt 2.1.5). Für den Benutzer soll es zusätzlich möglich sein, nicht-funktionale Kriterien oder Bedingungen in die Beschreibung der von ihm initiierten mobilen Prozesse zu integrieren, um die qualitativen Eigenschaften der Dienste, die zur erfolgreichen Prozessausführung erbracht werden müssen, festzulegen. Ein Prozess soll also über seine eigentliche Funktion hinaus durch vordefinierte Parameter bestimmt werden können. Dabei könnte es sich beispielsweise um Quality-of-Service Parameter oder um Sicherheitseinstellungen handeln. Denkbar wäre die Angabe vertrauenswürdiger Geräte, Einschränkungen für die Migration mobiler Prozesse oder eine Auswahl von Authentifikationsverfahren. Die nicht-funktionalen Kriterien sollen sowohl einzeln für jeden benötigten Dienst festgelegt werden können, als auch global für den kompletten Prozess. Die Einhaltung der nicht-funktionalen Kriterien soll dann für die gesamte Prozessausführung sichergestellt werden. [Zap05]

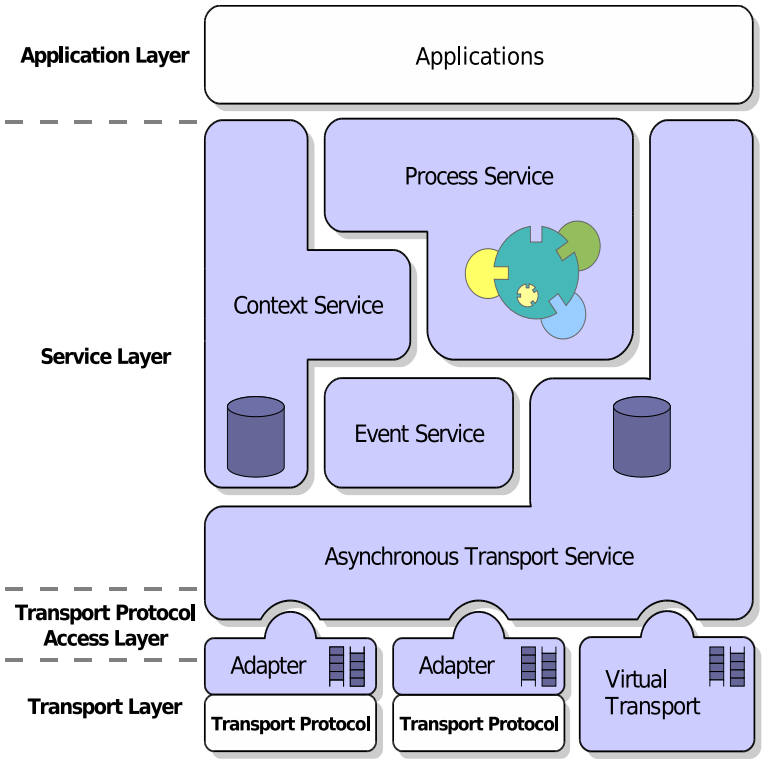

Abbildung 2.3: Abstrakte DEMAC-Architektur (aus [KZL07, S. 5])

2.4.2 Die Architektur der DEMAC-Middleware

Die nahtlose Integration mobiler Prozesse in mobile Middleware erfordert eine zugrundeliegende serviceorientierte Systeminfrastruktur, die ihre Realisierung in der DEMAC-Middleware gefunden hat [KZL07]. Die abstrakte Architektur der DEMAC-Middleware ist in Abbildung 2.3 dargestellt und gliedert sich in vier Hauptkomponenten, die im Folgenden beschrieben werden.

Kommunikationsbasis

Der Asynchronous Transport Service und der Event Service bilden die Kommunikationsbasis der DEMAC-Middleware. Der Transport Service stellt einen eigenen nachrichtenbasierten Transportmechanismus zum asynchronen Senden und Empfangen von Nachrichten zur Verfügung und abstrahiert auf diese Weise von den darunterliegenden konkreten Transportprotokollen wie beispielsweise Bluetooth, IrDA oder TCP/IP. Durch den Einsatz eines eigenen logischen Adressierungsschemas mit sogenannten *Device Handles* ist der Transport Service sowie alle darüberliegenden Komponenten zusätzlich unabhängig von den tatsächlich verwendeten physikalischen Adressen der Transportprotokolle. Der Transport Service bietet Komponenten und Anwendungen die Möglichkeit, sich mittels eines Transport Listeners am Transport Service zu registrieren, um über eingehende Nachrichten informiert zu werden. Jede über den Transport Service versandte Nachricht

hat dabei einen eindeutigen Bezeichner (*Message Handle*). Aufbauend auf dem Transport Service ist der Event Service für den Versand von Ereignisnachrichten (*Events*) zuständig, die Zustands- oder Attributsänderungen intern und extern bezüglich der lokalen Infrastruktur bekanntmachen. Ist ein Event Service am Erhalt fremder Events interessiert, so muss er sich am Event Service des betreffenden Gerätes registrieren. Die Benachrichtigung fremder Event Services geschieht unter Benutzung des Transport Service. Der benachrichtigte Event Service reagiert auf eingehende Events mit der Generierung lokaler Ereignisnachrichten, die an hierfür registrierte interne Komponenten weitergereicht werden. [KZL07], [KZL06]

Context Service

Der Context Service sammelt und verwaltet jegliche verfügbaren Informationen über den Kontext des implementierenden Gerätes. Das Wissen wird dabei entweder durch Events des Event Service oder aber durch einen direkten Nachrichtenaustausch über den Transport Service erworben. Fremden Geräten werden nur gefilterte und den Anfragen entsprechende Kontextinformationen präsentiert. Diese enthalten unter anderem *Quality-of-Service*-Parameter, Informationen über erreichbare Geräte sowie die Dienste, die diese anbieten, Ortsangaben und personenbezogene Daten über Benutzer und ihre Identitäten. Bereits vorhandenes lokales Wissen eines Context Service kann dabei nach dem Föderationsprinzip durch einen Informationsaustausch mit Geräten in der Umgebung erweitert werden. Damit diese Geräte oder andere Dienste gefunden werden können, enthält der Context Service einen verteilten Verzeichnisdienst, der seine Daten auf einer Peer-to-Peer-Basis etabliert. [Kun05], [KZL06]

Process Service

Der Process Service realisiert die Integration des Prozessmanagements in die DEMAC-Architektur. Er besteht zum einen aus der XML-basierten Prozessbeschreibungssprache DPDL (*DEMAC Process Description Language*) und zum anderen aus einer Ausführungsumgebung (*Execution Engine*) für mobile Prozesse. DPDL ermöglicht es einer Anwendung, Sequenzen auszuführender Aufgaben und die zu erreichbaren Zwischenergebnisse zu definieren. Neben den eigentlichen Konstrukten zur Prozessbeschreibung enthält DPDL ein Vokabular zur Definition nicht-funktionaler Kriterien eines Prozesses durch Benutzer oder Anwendungen, welches die Intentionen des Prozessinitiators über die gesamte Zeit der Prozessausführung hinweg wahren soll und daher die Formulierung von Bedingungen für eine verteilte Ausführung des Prozesses ermöglicht. Die Ausführungsumgebung kann die in der Prozessbeschreibung definierten Aufgaben und Aktivitäten lokal ausführen oder diese bei mangelnden Ressourcen an fremde Process Services delegieren. Hierzu muss der gesamte Prozess und alle benötigten Daten mit Hilfe des Transport Service übertragen werden. Der Process Service nutzt daher den Context Service zur

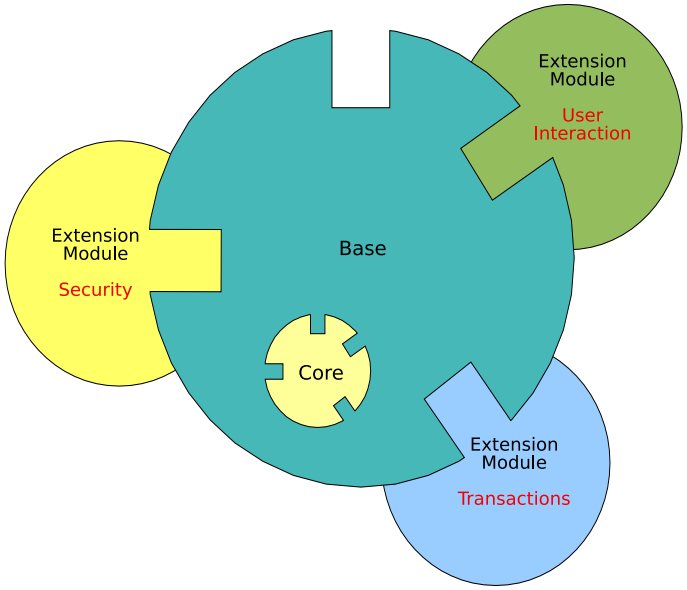

Abbildung 2.4: Architektur des DEMAC Process Service (aus [Zap05, S. 152])

Auffindung geeigneter Kooperationspartner. Die Ausführungsumgebung selbst ist modular aufgebaut. Die Kernkomponente (*Core Module*) enthält den Basis-Funktionsumfang zum Erhalt, zur Speicherung und zum Weiterleiten einer Prozessbeschreibung an andere Teilnehmer. Sie extrahiert die globalen nicht-funktionalen Kriterien (*Strategies*) des erhaltenen Prozesses und wertet diese aus. Die Basiskomponente (*Base Module*) soll die Aktivitäten des Prozesses ausführen und stellt somit die eigentliche Ausführungseinheit dar. Sie nutzt das Core Module zur Kommunikation mit anderen Geräten, falls es nicht den gesamten Prozess aufgrund fehlender Ressourcen und Dienste abarbeiten kann. Für jede Aufgabe überprüft das Base Module die Validität der eventuell vorhandenen nicht-funktionalen Kriterien. Das Base Module kann durch gerätespezifische optionale Zusatzkomponenten (*Extension Modules*) erweitert werden. Einen grafischenÜberblick über die Architektur des DEMAC Process Service verschafft Abbildung 2.4. [KZL07], [Zap05]

3 Sicherheit in informationstechnischen Systemen

Da der Begriff Sicherheit sehr weit gefasst ist und fast in allen Bereichen des Lebens eine Rolle spielt, soll in diesem Kapitel erläutert werden, was Sicherheit speziell in Computersystemen bedeutet und wie sie klassifiziert werden kann. Desweiteren sollen theoretische Konzepte sowie gängige praktische Lösungen zur Realisierung von Sicherheit in Computersystemen vorgestellt werden. Anhand dieser Ausführungen sollen die Unterschiede zwischen stationären und mobilen Systemen in Hinblick auf ihre Sicherheitsanforderungen geklärt und Realisierungsmöglichkeiten vorgestellt werden.

3.1 Grundlagen der Sicherheit

Der folgende Abschnitt erläutert zunächst wichtige Begriffe in Bezug auf Sicherheit informationstechnischer Systeme und zeigt mögliche Schwierigkeiten und Hindernisse auf. Aufbauend darauf werden Methoden und Konzepte sowie Praxisbeispiele für die Gewährleistung von Sicherheit und die Überwindung der dargestellten Hindernisse vorgestellt.

3.1.1 Definitionen und Klassifizierungen

Um den eher allgemeinen Begriff *Sicherheit* genauer zu charakterisieren wurden einige Versuche vorgenommen, eine treffende Definition zu finden. So definierte die *International Organization for Standardization* (ISO[1]) den Begriff Sicherheit als „*die Minimierung der Verwundbarkeit von Werten und Ressourcen*“ [Eck06, S. 5]. Diese Definition impliziert jedoch ein stets vorhandenes Restrisiko (*Residual Risk*). Es ist derzeit im Allgemeinen nicht möglich, ein System zu entwickeln, das zu hundert Prozent sicher ist. Eine Absicherung ist besonders dann schwierig, wenn das zu schützende System nicht isoliert betrieben wird, sondern z.B. Teil eines verteilten Systems ist und somit über eine Anbindung an Rechnernetze verfügt. Deshalb ist stets eine genaue Abwägung des Risikos eines potentiellen Schadens (meist finanzieller oder wirtschaftlicher Natur) gegen den Nutzen der auszuführenden Aktionen und Aufgaben erforderlich. [Sta02]

Um eine Präzisierung der Definition nach der ISO zu erreichen, klassifiziert Eckert [Eck06] die Sicherheit in informationstechnischen Systemen wie folgt:

- Funktionssicherheit (*Safety*)
  Unter der Funktionssicherheit eines Systems versteht man die Eigenschaft des Sy-

[1] Das Akronym *ISO* wurde ursprünglich vom griechischen Begriff *isos* für *gleich, einheitlich* abgeleitet.

stems, dass die spezifizierte Soll-Funktionalität seiner Komponenten mit der tatsächlich realisierten Ist-Funktionalität übereinstimmt. Dies bedeutet, dass ein funktionssicheres System keine Zustände annehmen kann, die nicht definiert oder unzulässig sind. [Eck06]

- Informationssicherheit (*Security*)
  Ein funktionssicheres System ist informationssicher, wenn es nur Systemzustände annehmen kann, die eine unauthorisierte Veränderung oder Gewinnung von Informationen unterbinden. [Eck06]

- Datensicherheit (*Protection*)
  Als Datensicherheit bezeichnet man die Eigenschaft eines funktionssicheren Systems, nur Systemzustände annehmen zu können, die keinen unauthorisierten Zugriff auf Ressourcen und hierbei insbesondere auf die enthaltenen Daten des Systems zulassen. Zur Gewährleistung der Datensicherheit gehören somit auch Mechanismen zur Anfertigung von Sicherungskopien (*Backups*), um vor Datenverlust zu schützen. [Hob98]

- Datenschutz / Privatsphäre (*Privacy*)
  Unter dem Begriff des Datenschutzes versteht man die Fähigkeit eines Menschen, die Nutzung und Verbreitung seiner personenbezogenen Informationen selbst bestimmen und kontrollieren zu können. Diese Eigenschaft wird als *informationelles Selbstbestimmungsrecht* bezeichnet und ist im Bundesdatenschutzgesetz (BDSG) verankert. [GH01]
  Bei den großen Datenmengen, die heutzutage alleine durch das *Surfen* im Internet von weltweit Millionen von Menschen auf Webseiten in- und ausländischer Betreiber entstehen, ist es in der Praxis nicht möglich, den tatsächlichen weiteren Gebrauch persönlicher Daten nach deren Preisgabe zu beeinflussen. Einem Mißbrauch kann in der Regel nur durch eine grundsätzliche Verschwiegenheit jeder Person bezüglich ihrer eigenen persönlichen Daten vorgebeugt werden. Eine Preisgabe personenbezogener Details sollte bei Notwendigkeit nur an geprüfte vertrauenswürdige Entitäten erfolgen. [Eck06]
  Als besonders wichtig wird die Gewährleistung der Anonymität von Entitäten und die Geheimhaltung ihrer Aufenthaltsorte (*Location Privacy*) betrachtet. Ist man beispielsweise als Mobilfunkteilnehmer bei einem Netzbetreiber registriert und trägt sein eingeschaltetes Mobiltelefon stets bei sich, so ist es theoretisch möglich, dass ein bis auf wenige Meter genaues Bewegungsprofil des Teilnehmers über die Zeit erstellt werden kann. Ebenso kann rekonstruiert werden, welche Rufnummern der Teilnehmer gewählt hat und welche Gespräche er empfangen hat. Die Offenlegung dieser Informationen wird jedoch durch Gesetze unterbunden und darf nur zum Zwecke der Verbrechensaufklärung durch gerichtliche Instanzen von den Betreibern angefordert werden. [BS03]

Die hier getroffene Klassifizierung erlaubt eine Unterscheidung zwischen Systemen, die sich auf den Schutz von verarbeiteten Informationen konzentrieren (Informationssicherheit), und solchen, die auf den Schutz von Daten als Repräsentanten von Informationen zielen (Datensicherheit). [Eck06]
Tatsächlich ist es jedoch so, dass sich die genannten vier Arten der Sicherheit informationstechnischer Systeme gegenseitig bedingen und nicht immer klar voneinander zu unterscheiden sind. So ist die Gewährleistung der Funktionssicherheit eines Systems eine wichtige Basis, um darauf aufbauend die anderen Klassen der Sicherheit realisieren zu können.

3.1.2 Gestaltung eines sicheren Systems

Die Gestaltung eines sicheren Systems ist ein Prozess, der unterschiedliche Aspekte berücksichtigen muss. Um adäquate Sicherheitsmechanismen und -strategien für ein System entwickeln zu können, muss man sich zunächst im Klaren darüber sein, welche Elemente des Systems überhaupt schutzbedürftig sind. Es ist also notwendig, die Schutzziele des Systems und somit auch die zu schützenden Güter zu ermitteln, um darauf aufbauend anforderungsgerechte und konkrete Sicherheitslösungen entwickeln und implementieren zu können. Eine solche Vorgehensweise wird als *Top-Down-Modell* bezeichnet. In informationssicheren Systemen müssen demnach Informationen geschützt werden, während in datensicheren Systemen die Daten die zu schützenden Güter darstellen. Sollen die Schutzziele eines Systems realisiert werden, muss der Zugriff auf die zu schützenden Güter beschränkt und kontrolliert werden. Nur authorisierten Entitäten (Personen, Rechnern) soll der Zugriff gewährt werden. [Eck06]

Die Schutzziele eines Systems stellen also die Anforderungen des Systems bezüglich Sicherheit dar. Diese Anforderungen müssen so früh wie möglich beim Entwurf des Systems mit berücksichtigt werden und erfordern eine sorgfältige Definition. Die Sicherheit eines Systems sollte also kein nachträgliches Anhängsel eines bereits fertigen Produktes sein. In der Praxis ist es jedoch leider häufig so, dass aus Zeit- und Kostengründen der Aspekt der Sicherheit bei den entwickelten Systemen und Produkten meist zu kurz kommt, da man diese möglichst schnell auf den Markt bringen möchte. So wurde beispielsweise die WLAN-Technologie nach IEEE 802.11 von vielen Herstellern gewinnbringend auf den Markt gebracht, obwohl längst bekannt war, dass die integrierte WEP-Verschlüsselung mit ihrer kurzen Schlüssellänge samt dem verwendeten (kryptographisch schwachen) RC4-Algorithmus keine ausreichende Sicherheit vor Eindringlingen in das so geschützte WLAN-Netz und allen damit verbundenen Konsequenzen bot. [Eck06]

Wenn die Schutzziele des Systems ermittelt worden sind, können Entscheidungen über die zu verwendenden Sicherheitsmechanismen getroffen werden. Sicherheitsmechanis-

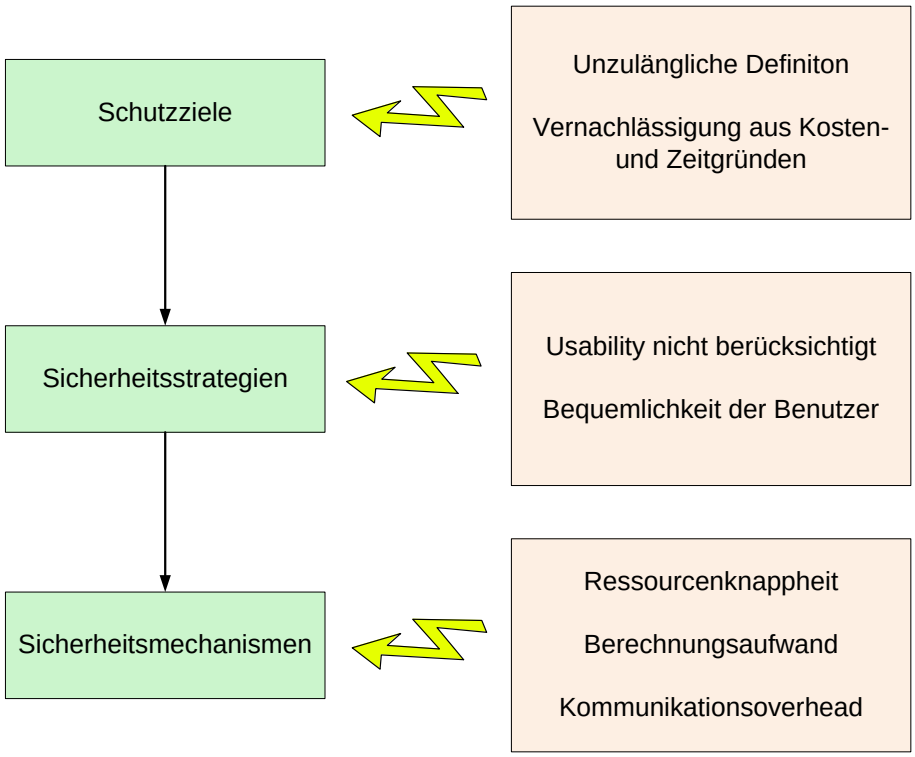

Abbildung 3.1: Störfaktoren eines Top-Down-Sicherheitsmodells

men werden als Teil des Systems implementiert. Sicherheitsmechanismen in informationstechnischen Systemen bedienen sich in der Regel kryptographischer Verfahren, um beispielsweise Konzepte wie die Verschlüsselung der Kommunikation zu realisieren. Die tatsächlich eingesetzen Sicherheitsmechanimsen hängen stark von den vorhandenen Ressourcen des Systems ab und von den Entwicklern, die darüber entscheiden, wie viele Anteile der Ressourcen sie für die Sicherheitsmechanismen reservieren wollen. Da gute Sicherheitsmechanismen alleine noch keine Systemsicherheit gewährleisten, ist es notwendig, dass Sicherheitsstrategien (sogenannte *Policies*) definiert werden, an die sich alle Benutzer des Systems halten müssen. Schlechte oder nicht vorhandene Sicherheitsstrategien führen dazu, dass Angreifer die an sich guten Sicherheitsmechanismen leicht überwinden können. Ein typisches Negativ-Beispiel ist das Anbringen von kleinen Zetteln mit Benutzernamen und Passwörtern an diejenigen Rechner, an denen die Authentifizierung und Authorisierung stattfindet, so dass jeder, der physischen Zugang zum Rechner erhält (z.B. eine Reinigungskraft), auch gleichzeitig einen (unbefugten) Zugriff auf Ressourcen erhält. Störfaktoren für ein verantwortungsbewußtes Einhalten von definierten Sicherheitsstrategien liegen einerseits in der Überforderung des Benutzers, wenn z.B. von ihm gefordert wird, sich viele verschiedene Passwörter zu merken, die aus zufälligen Zeichenketten bestehen. Andererseits kann es auch reine Bequemlichkeit sein, Sicherheitsanweisungen zu mißachten, weil man dadurch z.B. Zeit sparen kann. So könnte man seine Mitarbeiter dazu anhalten, stets die Passwortabfrage zur Benutzerauthentifizierung

einzuschalten, sobald sie sich von ihrem Rechner entfernen, um vor unauthorisiertem Zugriff zu schützen, allerdings haben einige Mitarbeiter möglicherweise keine Lust, immer ihr Passwort neu eintippen zu müssen, wenn sie ihren Arbeitsplatz für kurze Zeit verlassen. [Eck06]

Zusammenfassend kann man sagen, dass die sichere Gestaltung eines Systems ein Prozess ist, bei dem viele Aspekte eine Rolle spielen und ebenso viele Störfaktoren auftreten können. So gehört zur Auswahl von Sicherheitsmechanismen und Sicherheitsstrategien beispielsweise auch, dass man den Aspekt der *Usability*, also der Benutzbarkeit des Systems berücksichtigt. Dies fällt auch unter den Forschungsgegenstand der Mensch-Maschine-Interaktion (*Human-Computer Interaction, HCI*). Abbildung 3.1 zeigt einen Überblick über die größten Störfaktoren bei der Entwicklung und Durchsetzung der Schutzziele, der Sicherheitsmechanismen und der Sicherheitsstrategien.

3.1.3 Eigenschaften eines sicheren Systems

Im Folgenden sollen die Sicherheitsanforderungen informationstechnischer Systeme vorgestellt werden. Zudem soll aufgezeigt werden, wie die unterschiedlichen Anforderungen miteinander zusammenhängen.

Vertraulichkeit

Die Vertraulichkeit ist eng mit der Zugriffskontrolle auf Objekte verbunden. Vertraulichkeit bedeutet, dass man einzelne Objekte, wie z.B. Nachrichten, vor dem Zugriff anderer bewahrt, damit der Inhalt dieser Objekte geheim bleibt. Der Inhalt besteht aus Informationen, zu denen nur Befugte Zugang erhalten sollen. Zum Schutze der Vertraulichkeit von Informationen kann beispielsweise die Verschlüsselung benutzt werden. Auch vordergründig irrelevant erscheinende Informationen, wie z.B. der Verkehrsfluss zwischen Teilnehmern eines Kommunikationssystems, sollten vertraulich bleiben, wenn dadurch Rückschlüsse auf andere Informationen verhindert werden können. [Sta02]

Integrität

Die Integrität von Daten zu sichern bedeutet, die Daten vor unbefugter Veränderung zu schützen, bzw. eine Veränderung anzuzeigen, um zu gewährleisten, dass zwischen Sender und Empfänger ein und dasselbe Objekt ausgetauscht worden ist. Die Daten sollen also Konsistenz aufweisen. Dies kann durch die Berechnung von *Hashwerten* (kryptographische Prüfsummen, vgl. Abschnitt 3.2.3) auf Seiten des Senders und des Empfängers und einen Vergleich derselben geschehen. Bekannte Hash-Algorithmen sind z.B. *MD5* und *SHA-1*. [Rot05]

Authentizität / Authentifizierung

Die Authentizität einer Information ist ihre Echtheit und Zuverlässigkeit. Authentizität bezieht sich also auf die ausgetauschten Objekte selbst und ihre Glaubwürdigkeit. Die Authentifizierung hingegen betrifft die kommunizierenden Parteien. Sie sollen sich möglichst gegenseitig authentifizieren, um sicher zu gehen, mit wem sie es auf der anderen Seite zu tun haben. Es soll dabei verhindert werden, dass ein böswilliger Teilnehmer die Identität eines anderen missbrauchen kann (englisch *impersonation*). Kommunizieren zwei authentifizierte Parteien miteinander, und ist die Integrität der ausgetauschten Informationen gesichert, so können die Parteien diese Informationen als authentisch betrachten. Die Authentifizierung von Entitäten ist durch das Wissen um ein gemeinsames Geheimnis (z.B. Passwörter, geheime Schlüssel), durch den Besitz spezieller Gegenstände (z.B. *Smartcards*) oder durch das Aufweisen biometrischer Merkmale (z.B. Iris, Fingerabdruck) realisierbar. [Vac07]

Verfügbarkeit

Die Verfügbarkeit eines Systems soll so geartet sein, dass das System seinen Benutzern bei Bedarf zur Verfügung steht. Es muss gegen Angriffe auf die Verfügbarkeit (*Denial-of-Service-Angriffe*) geschützt werden, auch wenn dies sehr schwer zu realisieren ist. Die Verfügbarkeit von Systemen mit kommerziellen Dienstleistungsangeboten spielt heutzutage eine große Rolle als wirtschaftliche Komponente, da die Nichtverfügbarkeit dieser Angebote auch für nur kurze Zeit bereits den Verlust von Aufträgen oder gar Kunden an die Konkurrenz bedeuten kann. [Eck06]

Nicht-Bestreitbarkeit

Nicht-Bestreitbarkeit bedeutet, dass Sender bzw. Empfänger von Nachrichten nicht leugnen können, diese verschickt bzw. empfangen zu haben. Der Sender der Daten braucht also einen Nachweis (Quittung) darüber, dass seine Sendung beim Empfänger angekommen ist. Umgekehrt soll der Empfänger auch über die Identität des Senders sicher sein können, so dass keiner von beiden hinterher den Austausch der Sendung abstreiten kann. Man spricht auch von der *Verbindlichkeit* der Nachricht. [Vac07]

Zugriffskontrolle

Um eine Zugriffskontrolle zu realisieren und so vorhandene Ressourcen vor unbefugtem Zugriff zu schützen, kann man beispielsweise Zugriffskontrolllisten benutzen. Auf diesen sind dann Benutzer mit ihren Befugnissen festgehalten. Eine gute Zugriffskontrolle setzt voraus, dass es möglich ist, die Identität eines Nutzer mit Sicherheit bestimmen zu können. [Vac07]

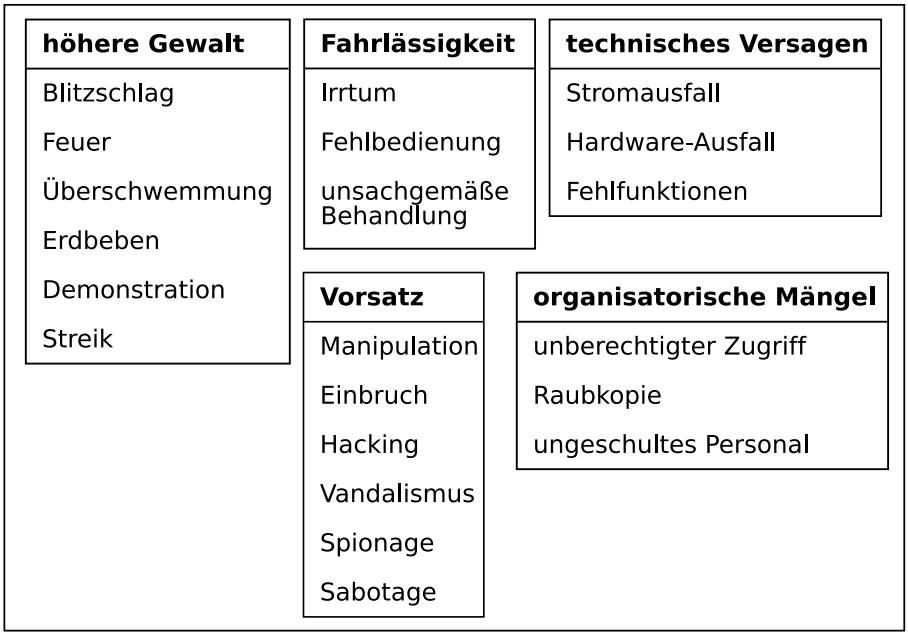

Abbildung 3.2: Gefährdungsfaktoren der IT-Sicherheit (nach [Eck06, S. 15])

3.1.4 Schwachstellen, Bedrohungen und Risiken eines Systems

Ein informationstechnisches System kann unterschiedliche Schwachstellen besitzen und einer Vielzahl von Bedrohungen und Risiken ausgesetzt sein. Diese Begriffe sollen daher nachfolgend erläutert werden.

Schwachstellen und Gefährdungsfaktoren Eine Schwachstelle bezeichnet einen Punkt, an dem ein System verwundbar ist. Umgekehrt kann man sagen, dass eine Verwundbarkeit eine Schwachstelle im System ist, die es ermöglicht, Sicherheitsmechanismen des Systems zu umgehen. Eine physische Schwachstelle mobiler Geräte ist beispielsweise ihre Anfälligkeit gegenüber Verlust oder Diebstahl. Typische Schwachstellen von Software oder Webseiten sind es dagegen, wenn durch mangelhafte Überprüfung von Benutzereingaben Pufferüberläufe (*Buffer Overflows*) möglich sind und fremder (schadhafter) Quelltext eingeschleust werden kann (*Remote Code Injection*). [Eck06]
Eine Analyse der Gefährdungsfaktoren eines Systems läßt auf dessen potentielle Schwachstellen schließen. In Abbildung 3.2 ist eine Klassifizierung der Gefährdungsfaktoren eines IT-Systems dargestellt, wie sie vom *Bundesamt für Sicherheit in der Informationstechnik* (BSI) vorgeschlagen wird.

Bedrohungen Die Bedrohung eines Systems verfolgt das Ziel, Schwachstellen oder Verwundbarkeiten des betreffenden Systems auszunutzen. Führt dies zum Erfolg, so könnten die in Abschnitt 3.1.3 genannten Eigenschaften eines sicheren Systems kompromittiert werden. Verschiedene Bedrohungen können unterschiedlich gewichtet werden. Dies

hängt von der Funktionalität und der Einsatzumgebung des Systems ab. [Eck06]

Risiko Unter dem Risiko einer Bedrohung versteht man die Wahrscheinlichkeit eines Schadensereignisses und die damit verbundene Höhe des Schadens. Zu einer Risikoanalyse gehört unter anderem die Analyse des Schadenspotentials, bei der die zu schützenden Werte und Objekte (die Schutzziele) und ihre Bedeutung innerhalb des Systems oder Unternehmens bekannt sein müssen. [Eck06]

3.1.5 Klassifizierung von Angriffen auf Computersysteme

In diesem Abschnitt sollen mögliche Angriffe auf informationstechnische Systeme dargestellt werden. Die im Folgenden genannten Angriffsarten wurden ursprünglich für stationäre Systeme bekannt. Bei der Darlegung soll aber insbesondere darauf geachtet werden, wieso und in wie fern diese Angriffsarten speziell für mobile Systeme eine Bedrohung darstellen.

Angriff Unter einem Angriff versteht man einen nicht authorisierten Zugriff oder Zugriffsversuch auf ein System. Man unterscheidet zwischen passiven und aktiven Angriffen. Passive Angriffe zielen auf eine unauthorisierte Informationsgewinnung und somit auf die Verletzung der Vertraulichkeit ab. Sie werden häufig durch das Abhören der Kommunikation durchgeführt. Aktive Angriffe zielen hingegen zusätzlich auf die Manipulation von Daten und Informationen sowie auf die Irreführung von Kommunikationspartnern, um diesen zu schaden. Sie richten sich somit gegen die Vertraulichkeit, Integrität, Authentizität und die Verfügkarkeit von Informationen, Daten, Objekten und Personen in informationstechnischen Systemen. [Eck06]

Lauschen

Als Lauschangriff bezeichnet man den Versuch, einen Kommunikationskanal abzuhören, um die darüber versandten Nachrichten mitzuhören und an ihren Informationsgehalt heran zu kommen. In mobilen Systemen ist der Schutz vor diesem Angriff eine besondere Herausforderung, da der Kommunikationskanal die Luft ist, und so im Prinzip von jedermann abgehört werden kann. Das Lauschen ist eine passive Angriffsmethode und bleibt im Regelfall unerkannt. Der Angreifer kann selbst bei verschlüsselter Kommunikation eine Verkehrsflußanalyse machen, um auf diese Weise Informationen zu gewinnen. Es wird dabei analysiert, wer, wann und wie lange mit wem kommuniziert hat. Wird der übertragene Verkehr protokolliert, so spricht man auch vom *Sniffing*. [ED06]

Spoofing

Als *Spoofing* bezeichnet man das Vortäuschen einer Identität, bzw. das Verschleiern der eigenen Identität. Spoofing ist ein Oberbegriff für alle Methoden, die rechtmäßige Au-

thentifizierung zu umgehen versuchen. Ein bekanntes Beispiel ist das ARP-Spoofing (*Address Resolution Protocol*), auch besser bekannt als *Man-in-the-Middle*-Angriff, bei dem ein Angreifer den beiden betroffenen Parteien jeweils die andere Identität vorspielt. Gelingt ihm dies, so kann er nach Belieben Nachrichten abfangen, verändern oder neu wiedereinspielen (*Replay*-Angriff). Der Man-in-the-Middle-Angriff läßt sich nicht einfach durch Verschlüsselung verhindern, da der Angreifer das Kommunikationsprotokoll auch kennt und sich durch den Angriff Schlüssel der kommunizierenden Parteien zu eigen machen kann. Auch recht populär ist das sogenannte IP-Spoofing, das Vortäuschen einer anderen IP-Adresse (*Internet Protocol*) als die, die vom *Internet Service Provider* vergeben wurde. Dies kann aber auch als eine Schutzmaßnahme zur Wahrung der Anonymität im Netz und gegen die Erstellung von Benutzerprofilen genutzt werden. [Eck06]

Denial-of-Service-Angriff

Der Denial-of-Service-Angriff ist ein Angriff auf die Verfügbarkeit eines Systems, indem dieses mit zu viel Anfragen überlastet wird und daher selbst rechtmäßige Anfragen nicht mehr bearbeiten kann. Durch die Analyse der durchschnittlichen *Requests* kann ein System aber bei deutlich erhöhtem Aufkommen das Bearbeiten dieser Requests unterlassen, um sicherzustellen, dass nicht das gesamte System blockiert. Allerdings werden damit auch seriöse Anfragen geblockt. In mobilen Systemen kann die Verfügbarkeit drahtloser Netze zusätzlich durch elektromagnetische Felder, welche die Frequenzen der Netze stören, angegriffen werden. [Sta02]

3.2 Kryptographische Verfahren

Die Sicherheit in informationtechnischen Systemen wird grundlegend durch einen sinnvollen Einsatz kryptographischer Verfahren realisiert. Die Kryptographie ist dabei die Wissenschaft vom Verschlüsseln von Informationen. Sie hat zum Ziel, die in Abschnitt 3.1.3 genannten sicheren Eigenschaften eines Systems wie Vertraulichkeit, Integrität, Authentizität und Nicht-Bestreitbarkeit zu ermöglichen. Die Kryptoanalyse bezeichnet hingegen die Analyse von kryptographischen Verfahren mit dem Ziel, ihre Sicherheit nachzuweisen oder zu widerlegen. Dazu versuchen Kryptoanalytiker unter anderem auch, kryptographische Verfahren zu brechen und ihre Schutzfunktionen zu umgehen. Die beiden Teilgebiete Kryptographie und Kryptoanalyse werden unter dem Oberbegriff der Kryptologie zusammengefasst. [Beu05]
Eine umfassende Darstellung aller Verfahren und Techniken der Kryptologie kann in diesem Abschnitt aus Platzgründen nicht gegeben werden. Deshalb werden nur die wichtigsten kryptographischen Verfahren und Algorithmen im Folgenden klassifiziert und benannt, da diese im weiteren Verlauf der vorliegenden Arbeit benötigt werden. Die Darstellungen sind vereinfacht und stellen daher nur die grundsätzlichen Ideen vor.

Das Kerckhoffs-Prinzip Eine der wichtigsten Anforderungen an praxisrelevante Krypto-Systeme ist das bereits im Jahr 1883 von Auguste Kerckhoffs formulierte und nach ihm benannte Kerckhoffs-Prinzip. Es besagt, dass das Maß an Sicherheit eines Krypto-Systems nicht von der Geheimhaltung der verwendeten Ver- und Entschlüsselungsverfahren abhängig sein darf, sondern einzig und allein von der Geheimhaltung der verwendeten Schlüssel. Wenn die Sicherheit eines kryptographischen Verfahrens hingegen nur auf der Geheimhaltung der verwendeten Algorithmen basiert, sprich man auch von *Security by Obscurity*. Die Veröffentlichung von Algorithmen hat den Vorteil, dass diese von anderen Experten begutachtet und eventuelle Schwachstellen schneller diagnostiziert und beseitigt werden können. [Sta02]

3.2.1 Symmetrische Verschlüsselung

Symmetrische Verschlüsselungsverfahren haben das Ziel, die Vertraulichkeit der Kommunikation zwischen zwei Kommunikationspartnern mittels des Wissens um ein gemeinsames Geheimnis zu gewährleisten. Die symmetrische Verschlüsselung einer Klartext-Nachricht m zu einer chiffrierten Nachricht m' erfolgt durch Anwendung eines symmetrischen Verschlüsselungsalgorithmus mit einem symmetrischen Schlüssel K. *Symmetrisch* bedeutet in diesem Zusammenhang, dass m' nur durch Verwendung des gleichen Schlüssels K wieder entschlüsselt werden kann, damit die originale Nachricht m wieder lesbar wird. Ohne den Schlüssel K kann ein Empfänger der Nachricht m' keine Rückschlüsse auf die ursprüngliche Nachricht m gewinnen. Die Sicherheit des Verfahrens hängt also zum großen Teil von der Geheimhaltung des Schlüssels ab. Sie ist aber auch von der kryptographischen Stärke des Schlüssels abhängig. Darunter ist einerseits die Länge des Schlüssels und andererseits die in ihm enthaltene Redundanz zu verstehen. Je länger ein Schlüssel ist und umso weniger redundante Informationen er enthält, desto sicherer ist die Verschlüsselung mit ihm. [Sch05]

Symmetrische Verschlüsselungsalgorithmen werden in Blockchiffren und Stromchiffren unterschieden. Bei Stromchiffren wird der Klartext bzw. Chiffre-Text Zeichen für Zeichen ver- bzw. entschlüsselt, während ein Blockchiffre den Klartext in Blöcke mit einer festen Blockgröße, die aus mehreren Zeichen besteht, unterteilt. Pro Schritt wird dann jeweils ein ganzer Block verarbeitet. Die grundlegenden Prinzipien, die angewendet werden, um einem unverschlüsselten Klartext in einen Chiffretext zu überführen, sind dabei die Prinzipien der Substitution und der Transposition. Da diese in der Regel auf einfachen Operationen wie z.B. *Shift* und *XOR* basieren, sind symmetrische Verfahren ressourcensparend im Hinblick auf die Prozessorleistung und den Arbeitsspeicher des Rechners, auf dem sie durchgeführt werden. Bekannte symmetrische Verschlüsselungsalgorithmen sind z.B. DES (*Data Encryption Standard*) und seine Weiterentwicklung TDES (*Triple DES*), *Blowfish* und der derzeit kryptographisch stärkste Vertreter AES (*Advanced Encryption Standard*). [Bis05]

Der Nachteil symmetrischer Verschlüsselung liegt in der Natur des Verfahrens selbst. Durch die Verwendung genau eines Schlüssels zu Ver- und zur Entschlüsselung einer Nachricht, müssen zwei Entitäten, die verschlüsselt über einen unsicheren Kanal kommunizieren wollen, beide über diesen Schlüssel verfügen. Daraus resultiert ein Schlüsselverteilungsproblem. Neben der verschlüsselten Nachricht muss nämlich auch der symmetrische Schlüssel übermittelt werden. Geschähe dies über den gleichen unsicheren Kommunikationskanal, so könnte ein Angreifer Nachricht und Schlüssel abfangen und würde so in den Besitz des Schlüssels gelangen, den er dann auf die Chiffre-Nachricht anwenden könnte. Vertraulichkeit wäre unter diesen Umständen nicht gegeben. Der Schlüssel muss daher selbst geschützt oder aber über einen anderen, sicheren Kanal übertragen werden. [Bis05]

Da mobile Geräte bereits durch ihre natürliche Beschaffenheit über begrenztere Ressourcen als stationäre Rechner verfügen (vgl. Abschnitt 2.1.5), bietet sich die Verwendung symmetrischer Verfahren in solchen Systemen besonders an. In hochdynamischen mobilen Ad-hoc-Systemen ohne feste Infrastrukturen stellt sich jedoch die Frage nach der Art der Schlüsselverteilung, da sich die Teilnehmer *a priori* fremd sind und ein gemeinsames Geheimnis erst noch etabliert werden muss. [SU06]
Seit der Entwicklung der *Public-Key-Kryptographie*, die im nächsten Abschnitt eingeführt wird, kann die asymmetrische Verschlüsselung eingesetzt werden, um einen symmetrischen Schlüssel zu verschlüsseln und ihn so über einen unsicheren Kanal übertragen zu können.

3.2.2 Public-Key-Kryptographie

Das Konzept der Public-Key-Kryptographie stellt ein asymmetrisches Verfahren dar und wurde erstmals im Jahre 1976 von Whitfield Diffie und Martin Hellman [DH76] vorgestellt. Das Neuartige dieses Konzepts ist die strikte Unterscheidung zwischen Schlüsseln zur Verschlüsselung und Schlüsseln zur Entschlüsselung von Nachrichten und ihre besondere Handhabung. Jede teilnehmende Entität besitzt ein eigenes Schlüsselpaar, welches aus einem öffentlichen, zum Verschlüsseln vorgesehenen Schlüssel und einem privaten, zum Entschlüsseln geeigneten Schlüssel besteht. Der Besitzer muss seinen privaten Schlüssel unbedingt geheim halten, während der öffentliche Schlüssel unter den Kommunikationspartnern verteilt wird. Der öffentliche Schlüssel kann auch auf einem dafür vorgesehenen *Key Server* publiziert werden. [Sch05]

Asymmetrische Verschlüsselung Der öffentliche Schlüssel ermöglicht es jedem, der ihn kennt, Daten für den Schlüsselinhaber zu verschlüsseln, die dieser dann nur mit seinem passenden privaten Schlüssel entschlüsseln kann. Niemand sonst kann die chiffrierte Nachricht entziffern, es sei denn er hätte den privaten Schlüssel des Inhabers

kompromittiert. Die beiden Schlüssel eines asymmetrischen Schlüsselpaares sind offensichtlich komplementär zueinander. Die wichtigste Anforderung an die beiden Schlüssel ist die Unmöglichkeit, einen privaten Schlüssel aus seinem öffentlichen Gegenstück ableiten zu können. Das der Public-Key-Kryptographie zugrundeliegende mathematische Schema basiert auf Erkenntnissen, dass die Faktorisierung einer großen Zahl (Primfaktorzerlegung) sehr aufwendig ist, während das Erzeugen einer Zahl durch Multiplikation zweier Primzahlen schnell berechnet werden kann. Eine Funktion, bei denen eine Richtung (Multiplikation) leicht, die andere (Faktorisierung) aber schwierig zu berechnen ist, bezeichnet man als Einwegfunktion. Um eine Verschlüsselung nach diesem Prinzip wieder rückgängig machen zu können, muss es sich um eine sogenannte *Trap-Door*-Einwegfunktion handeln, die mit Hilfe einer Zusatzinformation (nämlich dem entsprechenden asymmetrischen Schlüssel) auch rückwärts leicht zu berechnen ist. [Sch05]

Digitale Signaturen Um die Authentizität und Nicht-Bestreitbarkeit von Daten oder Nachrichten zu gewährleisten werden digitale Signaturen eingesetzt. Eine digitale Signatur soll den Urheber einer Nachricht eindeutig identifizieren. Dazu könnte dieser die Nachricht mit seinem privaten Schlüssel verschlüsseln. Andere Entitäten können die Nachricht mit Hilfe des öffentlichen Schlüssels des Urhebers wieder entschlüsseln und den Urheber dadurch eindeutig feststellen, da nur er im Besitz des privaten Schlüssels sein kann, der zur Verschlüsselung der Nachricht benutzt wurde. Um den Rechenaufwand zu verringern, wird eine digitale Signatur meist nur über den Hashwert einer Nachricht erstellt und dieser beigefügt. Der Empfänger der Nachricht verifiziert die Signatur, in dem er einen eigenen Hashwert der Nachricht berechnet und ihn mit dem Hashwert aus der entschlüsselten Signatur vergleicht. [Eck06]

Im Gegensatz zu einem symmetrischen Verfahren müssen die kommunizierenden Parteien bei der Public-Key-Kryptographie also keinen gemeinsamen geheimen Schlüssel kennen. Der Nachteil asymmetrischer Verfahren liegt jedoch im Aufwand der durchzuführenden Berechnungen, da die spezielle Beschaffenheit eines Schlüsselpaares und die damit verbundenen Operationen mathematisch komplex sind. Die bekanntesten asymmetrischen Algorithmen, die sowohl zur Public-Key-Verschlüsselung, als auch zur Erstellung digitaler Signaturen verwendet werden können, sind die nach ihren Entwicklern benannten Algorithmen *RSA* (*Rivest, Shamir, Adleman*, 1978) und *ElGamal* (1985). Ein asymmetrischer Algorithmus zur Etablierung eines geheimen symmetrischen Schlüssels zweier fremder Parteien über einen unsicheren Kanal ist beispielsweise der *Diffie-Hellmann*-Schlüsselaustausch. Allerdings ergibt sich hierbei noch keine Authentifikation der beiden Parteien, so dass der Algorithmus anfällig für einen Man-in-the-Middle-Angriff ist. [Sta03]

3.2.3 Kryptographische Prüfsummen

Kryptographische Prüfsummen sollen die Integrität von Nachrichten gewährleisten. Die Prüfsummen lassen sich in Hashwerte und *Message Authentication Codes* unterteilen.

Hashwert

Ein Hashwert ist eine kryptographische Prüfsumme, die von einer sogenannten *Hashfunktion* aus einem beliebig langen Datensatz berechnet wird. Der Hashwert hat dabei eine feste Länge (z.B. 128 Bit), die deutlich kleiner als die des Datensatzes ist und diesen vor unberechtigter Manipulation schützen soll. Der Einsatz von Hashfunktionen schützt dabei nicht unmittelbar den Datensatz selbst, ermöglicht es jedoch, eine Veränderung an ihm erkennbar zu machen. Theoretisch kann es zu einem gegebenen Hashwert mehrere passende Datensätze geben. [Sch05]
Für eine sichere Hashfunktion h müssen daher folgende Eigenschaften gelten:

- Die Hashfunktion h muss eine Nachricht m einer beliebigen Länge in einen Hashwert $h(m)$ fester Länge umwandeln. Dieser Vorgang soll leicht (effizient) zu berechnen sein. Ähnliche Nachrichten sollen zu völlig verschiedenen Hashwerten führen. Im Idealfall sollte das Invertieren eines Bits der Eingabe-Nachricht durchschnittlich die Hälfte aller Bits im resultierenden Hashwert verändern. [ED06]

- Einwegeigenschaft / Unumkehrbarkeit
  Es muss praktisch unmöglich sein, zu einem gegebenen Hashwert y einen Datensatz x zu finden, für den gilt $h(x) = y$. [Sch05]

- Schwache Kollisionsresistenz
  Es muss praktisch unmöglich sein, zu einem gegebenen Datensatz x einen zweiten Datensatz x' mit $h(x') = h(x)$ zu finden. [Sch05]

- Starke Kollisionsresistenz
  Es muss praktisch unmöglich sein, zwei beliebige Datensätze x und x' zu finden, die den gleichen Hashwert besitzen. [Sch05]

Praktisch unmöglich bedeutet dabei, dass die genannten Anforderungen trotz Einsatzes modernster Rechner nicht innerhalb eines sinnvollen Zeitrahmens erfolgreich durchgeführt werden können [Sch05]. An Stelle eines Hashwertes ist auch die Bezeichnung *Message Digest* üblich.

Wird eine Nachricht zusammen mit ihrem Hashwert an einen Empfänger verschickt, so berechnet dieser einen eigenen Hashwert über die Nachricht und vergleicht ihn mit dem empfangenen Wert. Bei einer Übereinstimmung sollte er sich idealerweise sicher sein können, dass die Nachricht beim Transport nicht manipuliert worden ist. Der Ablauf der Integritätskontrolle einer übertragenen Nachricht sieht wie folgt aus [Eck06]:

- Der Urheber einer Nachricht m berechnet den Hashwert $h(m)$. Sowohl die Nachricht m als auch ihr Hashwert $h(m)$ werden übertragen.

- Zur Kontrolle der Integrität einer Nachricht m' berechnet der Empfänger zunächst deren Hashwert $h(m')$ und vergleicht dieses Ergebnis mit dem Hashwert $h(m)$, der zusammen mit der Nachricht übertragen wurde.

- Gilt $h(m) = h(m')$, so kann aufgrund der kollisionsresistenten Eigenschaften kryptographisch sicherer Hashfunktionen angenommen werden, dass auch $m = m'$ gilt und m' somit eine nicht modifizierte, originale Nachricht ist.

Wenn die Nachricht und der Hashwert aber unverschlüsselt über einen unsicheren Kanal übertragen werden, so könnte ein Angreifer beide abfangen, die Nachricht manipulieren, einen neuen, selbst errechneten Hashwert beifügen und diesen manipulierten Datensatz an den Empfänger weiterleiten. Eine mögliche Lösung dieses Problems stellt die asymmetrische Verschlüsselung oder die digitale Signierung des Hashwertes für den Transportweg dar. Ein anderes Lösungkonzept ist der Einsatz eines *Message Authentication Codes*, welches im nächsten Abschnitt eingeführt wird.

SHA-1 Ein bekannter, lange Zeit als sicher geltender Hashalgorithmus ist SHA-1 (*Secure Hash Algorithm*), der im Jahre 1993 vom amerikanischen *National Institute of Standards and Technology* (NIST) veröffentlicht wurde und 160-Bit-lange Hashwerte erzeugt. Seit 2005 wurden jedoch mehrere praxisrelevante Angriffe bekannt, die die Zeit für die Berechnung von Kollisionen (Kollisionsangriff) stark reduziert haben, so unter anderem von Wang et al. [WYY05].
SHA-1 hat mittlerweile Nachfolger wie z.B. SHA-256, SHA-384 und SHA-512, wobei die beigefügte Zahl jeweils die Länge des erzeugten Hashwerts in Bit angibt. Die Gruppe der Nachfolger von SHA-1 wird auch unter dem Begriff SHA-2 zusammenfasst. [Sch05]

MD5 MD5 (*Message Digest Algorithm 5*) ist ein im Jahr 1990 von Rivest eingeführter Hashalgorithmus, der einen 128-Bit-langen Hashwert erzeugt. MD5-Hashwerte werden unter anderem von PGP (vgl. Abschnitt 4.4.2) verwendet und zur Integritätsprüfung von Dateien benutzt. Dabei wird der aktuelle MD5-Hashwert der Datei mit einem früher errechneten MD5-Wert verglichen. So kann festgestellt werden, ob die Datei verändert oder beschädigt wurde. [Eck06]
Beim MD5-Verfahren wurden bereits erfolgreiche Kollisionsangriffe vorgestellt. Diese erfolgten mittels einer erschöpfenden Suche, die parallel ausgeführt wurde. Vlastimil Klima gelang ein erfolgreicher Kollisionsangriff auf MD5 mit Hilfe eines Notebooks innerhalb weniger Stunden [Kli05].

Message Authentication Code

Zur Berechnung eines Message Authentication Codes (MAC) fließt sowohl die Nachricht selbst als auch zusätzlich ein geheimer symmetrischer Schlüssel mit ein. Man nennt sie daher auch Hashfunktion mit Schlüssel (englisch *Keyed One-Way Function*). Damit soll nicht nur die Integrität, sondern zusätzlich auch die Authentizität von Nachrichten gewährleistet werden, da ein MAC-Funktionswert nur bei Kenntnis des Schlüssels berechnet werden kann. Zur Verifikation des MAC muss der gleiche Schlüssel verwendet werden wie bei der Erstellung vor der Nachrichtenübertragung. Sender und Empfänger müssen also beide den gleichen geheimen Schlüssel kennen. Stimmt der vom Nachrichtenempfänger berechnete MAC mit dem empfangenen MAC überein, wurde die Nachricht nicht verändert. Ein Angreifer kann die unverschlüsselte Nachricht zwar ändern, da er aber den geheimen Schlüssel nicht kennt, kann er keinen gültigen MAC für die veränderte Nachricht erstellen. Dadurch lässt sich eine nachträgliche Änderung einer Nachricht erkennen. [Sch05]
Eckert betont jedoch, dass die durch MACs gewährleistete Authentizität schwach ist und sich auf den Urheber der Nachricht und nicht die Nachricht selbst bezieht. Handelt es sich bei der Nachricht beispielsweise um mobilen Code, so können aus dem MAC keine Informationen über dessen Funktionalität oder Schadhaftigkeit abgeleitet werden. [Eck06]

Ein bekanntes MAC-Verfahren ist der sogenannte HMAC-Algorithmus (*Keyed-Hashing for Message Authentication*), der als Hashfunktionen unter anderem auch SHA-1 oder MD5 benutzen kann. Der HMAC einer Nachricht m berechnet sich dabei wie folgt:

$$HMAC_K(m) = H\Bigg((K \oplus opad) \mid\mid H\Big((K \oplus ipad) \mid\mid m\Big)\Bigg)$$

Die Werte $opad$ und $ipad$ sind Konstanten, $\oplus$ steht für die bitweise XOR-Operation und $\mid\mid$ für die Konkatenation. [Eck06]

3.2.4 Public-Key-Infrastruktur

Als Public-Key-Infrastruktur oder kurz PKI bezeichnet man ein System mit zentralen Komponenten, welches es ermöglicht, *digitale Zertifikate* auszustellen, zu verteilen und zu prüfen. Ein digitales Zertifikat bindet einen öffentlichen Schlüssel an eine Identität. Es zertifiziert somit die Authentizität des Schlüssels und seine Gültigkeit. Das digitale Zertifikat selbst ist mit einer digitalen Signatur versehen, deren Echtheit mit dem öffentlichen Schlüssel des Ausstellers des Zertifikates geprüft werden kann. Um die Authentizität des Aussteller-Schlüssels überprüfen zu können, wird wiederum ein digitales Zertifikat benötigt. So läßt sich ein hierarchischer Validierungsbaum aufbauen, an dessen Wurzel ein Zertifikat stehen sollte, dem direktes Vertrauen entgegengebracht werden kann. Dieses

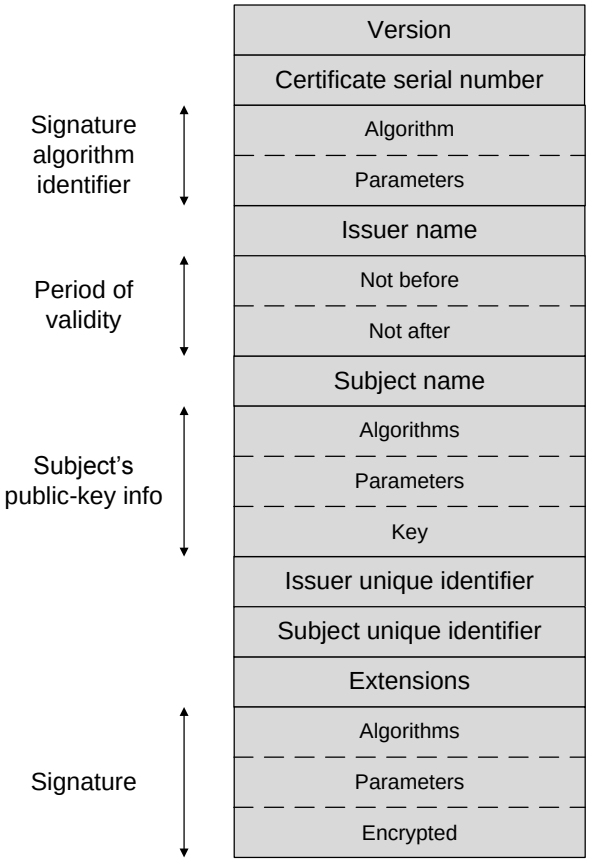

Abbildung 3.3: Aufbau eines X.509v3-Zertifikates (nach [Sta03, S. 420])

wird von einer sogenannten Root-Zertifizierungsstelle (*Root Certification Authority*) bereitgestellt. Eine PKI stellt weiterhin eine Zertifikatssperrliste (*Certificate Revocation List*, CRL) zur Verfügung, welche Zertifikate auflistet, die vor Ablauf der Gültigkeit zurückgezogenen wurden, weil beispielsweise das Schlüsselmaterial kompromittiert worden ist. Eine Zertifikatssperrliste muss daher regelmäßig aktualisiert werden. Zum Auffinden gesuchter Zertifikate wird meist ein Verzeichnisdienst (z.B. *LDAP*, *X.500*) bereitgestellt. [Beu05]

Der derzeit wichtigste PKI-Standard für digitale Zertifikate ist *X.509* in der aktuellen Version *X.509v3* von der *International Telecommunication Union* (ITU). In Abbildung 3.3 ist der grundlegende Aufbau eines X.509v3-Zertifikates dargestellt.
Um eine Public-Key-Infrastruktur auch in mobilen Umgebungen nutzbar zu machen, führte das *WAP-Forum* (*Wireless Application Protocol*) einen Sicherheitsstandard für drahtlose Netzwerke mit der Bezeichnung W-PKI (*Wireless PKI*) ein, bei dem sich digitale Zertifikate auf den SIM-Karten mobiler Geräte wie etwa Handys befinden. W-PKI wurde bislang allerdings nur in Pilotprojekten eingesetzt, da es noch technische Schwierigkeiten bei der Umsetzung gibt. [ED06]
Da Public-Key-Infrastrukturen zentrale Komponenten benötigen, sind sie für den Einsatz

in dezentralen Systemen mit Ad-Hoc-Charakter nicht unmittelbar geeignet. Geeignete Konzepte hierfür werden daher in Kapitel 4 untersucht.

3.3 Sicherheit im Internet

Für die Inanspruchnahme von Dienstleistungen im Internet werden Benutzer häufig aufgefordert, personenbezogene Informationen zu offenbaren, die jedoch schutzbedürftig sind und vertraulich behandelt werden müssen. Besonders bei Tätigkeiten wie Online-Banking oder dem Einkaufen im Internet müssen anwendungsnahe Mechanismen eingesetzt werden, die die Kommunikationspartner authentifizieren, ihre Kommunikation verschlüsseln und die Verwendung persönlicher Informationen regeln.

3.3.1 Authentifizierung und Vertraulichkeit: SSL/TLS

Da im Internet im Allgemeinen eine hohe Anonymität und ein damit verbundenes hohes Gefährdungspotential herrscht, sind Mechanismen der Authentifizierung und Vertraulichkeit in diesem Umfeld besonders wichtig. Diese Mechanismen sind in Form von Kommunikationsprotokollen realisiert und werden nachfolgend vorgestellt.

HTTP Das im Internet am meisten verbreitete Übertragungsprotokoll auf der Anwendungsschicht ist das zustandslose Protokoll HTTP (*Hypertext Transfer Protocol*). Es wird überwiegend dafür genutzt, Webseiten eines Servers in den Webbrowser eines Clients zu laden. Jede HTTP-Interaktion besteht aus der Anfrage eines Clients gefolgt von einer Antwort des angefragten Servers. Als zuverlässige Transportverbindung ist die Verwendung von TCP üblich, wird vom HTTP-Standard jedoch nicht zwingend gefordert. [Tan00]

HTTPS Da HTTP selbst keine Mechanismen zur Authentifizierung der beiden Kommunikationspartner und zur Verschlüsselung der kommunizierten Daten bereitstellt, diese Dienste aber bei sicherheitsbedürftigen Interaktionen wie beispielsweise dem Online-Banking benötigt werden, wurde HTTPS (*Hypertext Transfer Protocol Secure*) eingeführt. HTTPS ist dem HTTP-Protokoll syntaktisch gleich, ermöglicht jedoch unter der Verwendung des SSL/TLS-Protokolls zusätzlich die Authentifizierung der Kommunikationspartner und die Verschlüsselung ihrer Kommunikation. HTTPS ist heutzutage standardmäßig auf allen internetfähigen Computern vorhanden. Wie die genannten Protokolle in das TCP/IP-Referenzmodell eingebettet sind, ist in Abbildung 3.4 schematisch dargestellt. [Tan00]

SSL/TLS SSL (*Secure Sockets Layer*) ist ein hybrides Authentifizierungs- und Verschlüsselungsprotokoll für Datenübertragungen im Internet. Seit dem Jahr 1999 wird es als

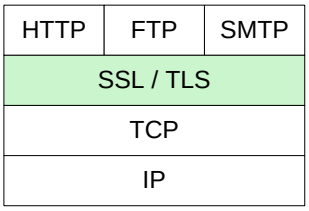

Abbildung 3.4: Einbettung des SSL-Protokolls (nach [Sta03, S. 530])

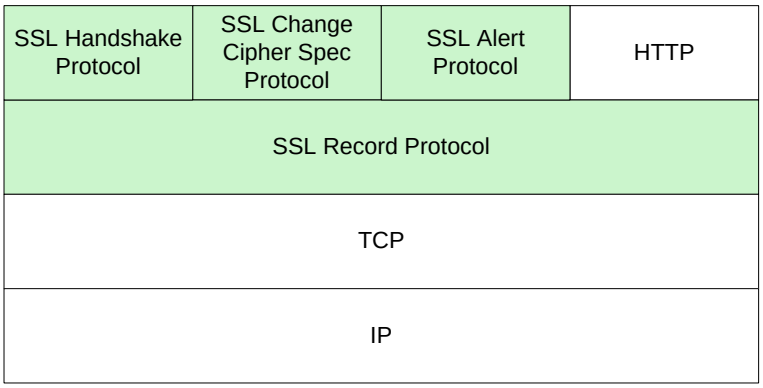

Abbildung 3.5: Der SSL-Protokollstapel (nach [Sta03, S. 531])

Standard unter dem Namen TLS (*Transport Layer Security*) von der IETF (*Internet Engineering Task Force*) weiterentwickelt. [Res00]
SSL besteht unter anderem aus den folgenden zwei Hauptkomponenten (vgl. Abbildung 3.5):

- *SSL Handshake Protocol*
  Das SSL-Handshake-Protokoll ist dafür zuständig, je zwei Kommunikationspartner auf Basis asymmetrischer Verfahren zu authentifizieren. Die gegenseitige Authentifizierung kann jedoch nicht erzwungen werden und ist optional. Die Authentifizierung kann mit Hilfe von X.509-Zertifikaten erfolgen. Desweiteren ermöglicht das Handshake-Protokoll eine Aushandlung kryptographischer Algorithmen und Schlüssel, die vom SSL-Record-Protokoll benötigt werden. [Res00]

- *SSL Record Protocol*
  Das SSL-Record-Protokoll verwendet symmetrische Algorithmen und einmalige Sitzungsschlüssel, um eine Ende-zu-Ende-Verschlüsselung der ausgetauschten Daten zwischen den beiden Kommunikationspartnern zu gewährleisten. Zusätzlich kann die Integrität und Authentizität einzelner Nachrichten mit Hilfe kryptographischer Prüfsummen abgesichert werden. [Res00]

Eine SSL-gesicherte Verbindung wird besonders bei Dienstleistungen im Bereich E-Commerce und Online-Banking verwendet, um Spoofing- und Lauschangriffe zu verhindern, damit kein wirtschaftlicher oder finanzieller Schaden entsteht.

Für mobile Geräte, die mittels des Protokolls WAP in der Version 1.x im Internet surfen, wird das Protokoll WTLS (*Wireless TLS*) eingesetzt, welches vom WAP-Forum eingeführt wurde und konzeptionell wie TLS arbeitet, jedoch auf die speziellen Bedürfnisse mobiler Geräte zugeschnitten ist. In der Version WAP 2.0 wurde WTLS jedoch durch HTTPS ersetzt. Damit wird auch die Interoperabilität zwischen mobilen und stationären Geräten verbessert. [ED06]

3.3.2 Schutz personenbezogener Daten: P3P

P3P (*Platform for Privacy Preferences*) wurde mit dem Ziel entwickelt, die Privatsphäre von Internetbenutzern zu schützen und ihnen einen schnellen Überblick darüber zu verschaffen, wie der Betreiber einer Webseite die von den Benutzern preisgegebenen personenbezogenen Daten weiterverarbeitet. Der Benutzer soll noch vor Preisgabe der persönlichen Informationen darüber aufgeklärt werden, in welchem Umfang die Daten gespeichert, verändert oder an Dritte weitergegeben werden. Diese Aufklärung erfolgt mittels einem standardisierten Austausch von Datenschutzinformationen, den sogenannten *Privacy Policies* mit den Betreibern. In den Privacy Policies wird angegeben, welche Informationen über Besucher gesammelt werden, wie lange sie gespeichert werden usw. [Cra02]

Um Datenschutz-Einstellungen, also die Handhabung personenbezogener Daten, abgleichen und aushandeln zu können, müssen diese formalisiert werden. P3P verfügt daher einerseits über ein Vokabular zur Beschreibung von Privacy Policies eines Betreibers sowie zur Beschreibung von Präferenzen eines Benutzers im Hinblick auf seine Privatsphäre. Andererseits wird ein Protokoll zum Anfordern, Übertragen und Aushandeln von Privacy Policies bereitgestellt. [Jen02]

Bevor P3P von einem Server verwendet werden kann, muss dieser zuerst seine eigenen Datenschutzrichtlinien als *Privacy Policy File* erstellen. Ein Privacy Policy File beschreibt die Regeln, nach denen personenbezogene Daten eines Benutzers gespeichert und weiterverwendet werden dürfen. Dies kann zunächst natürlichsprachlich formuliert werden. Darauf aufbauend wird dann mindestens ein Privacy Policy File in XML-Format generiert. Eine Policy kann verschiedene Geltungsbereiche haben. So kann sie beispielsweise für das gesamte Angebot des Servers gelten, oder aber nur für spezielle Dienste oder Seiten. Dies muss spezifiziert werden und wird in einem *Policy Reference File* festgehalten. Diese Datei kann beim Client angezeigt werden (z.B. im Webbrowser des Benutzers). Die P3P-Policies sind jedoch nicht zwingend gesetzeskonform. Sie beschreiben lediglich, welche Handlungen der Server vornimmt und welche nicht. [Cra02]

Will ein Client P3P nutzen, so muss die von ihm benutzte Software den P3P-Standard unterstützen. Der Client kann seine Präferenzen bezüglich Datenschutzeinstellungen fest-

legen und diese in seinen *Privacy Preferences* konfigurieren. Fordert er nun Seiten eines P3P-benutzenden Servers an, so werden die Privacy Preferences des Clients mit den Privacy Policies des Servers abgeglichen. Dem Client wird das Resultat mitgeteilt. Falls sich keine Konflikte ergeben haben, ist das Angebot des Servers uneingeschränkt nutzbar. Andernfalls können einige Optionen automatisch vom Server gesperrt werden, um den Wünschen des Clients gerecht zu werden, oder aber der Client kann die von ihm gewünschte Policy manuell aus dem Policy Reference File des Servers auswählen. Zur einfachen Handhabung von P3P und den damit verbundenen Protokollen gibt es Implementierungen des P3P-Standards als Plugin für bekannte Webbrowser wie z.B. *Mozilla Firefox* oder *Internet Explorer*. [Cra02]

Die Nachteile von P3P liegen hauptsächlich in der technischen Realisierung. So ist es beispielsweise nicht vorgesehen, dass die im P3P-Standard definierten Datenschutzerklärungen durch standardisierte technische Komponenten überprüft werden, um die im Hinblick auf die Handhabung der personenbezogenen Daten gemachten Aussagen zu validieren. Auch ist es nicht gewährleistet, dass gesetzlich vorgeschriebene Mindestanforderungen bezüglich des Datenschutzes eingehalten und kontrolliert werden. Da es keinen Erzwingungsmechanismus für die Einhaltung der von Servern präsentierten P3P-Richtlinien gibt, müssen die Benutzer den Dienstanbietern ein bestimmtes Maß an Vertrauen entgegenbringen und sie von anderen nicht vertrauenswürdigen Anbietern unterscheiden können, um sicher zu gehen, dass die Schutzmechanismen nicht doch umgangen werden. [Jen02]

Die Mechanismen von P3P zum Schutz der Privatsphäre von Benutzern lassen sich prinzipiell auch auf Servern, die Dienste für mobile Geräte anbieten, implementieren. Die Schwierigkeit liegt jedoch in der Realisierung benötigter P3P-Komponenten auf den mobilen Endgeräten selbst. Da die Nutzung eines mobilen Internetzugangs (z.B. über WAP) und ortsbasierten Diensten (*Location Based Services*) stetig zunimmt, wurde 2003 das Projekt PiMI (*Privacy in Mobile Internet*) ins Leben gerufen, welches in Kooperation mit der schwedischen Universität Karlstad, der Firma *Ericsson* und dem *Unabhängigen Landeszentrum für Datenschutz Schleswig-Holstein* durchgeführt wurde. In dem Projekt wurden die Möglichkeiten untersucht, wie P3P in mobile Geräte wie z.B. Handys eingebunden werden kann, um beispielsweise die Weitergabe von Ortsangaben zu unterbinden. Die P3P-Komponenten müssen so modelliert sein, dass sie den beschränkten Ressourcen und den kleinen Displays mobiler Geräte gerecht werden. [MÖ3]

In Abbildung 3.6 sind Screenshots einer möglichen P3P-Konfigurationssoberfläche dargestellt[2]. Das bereitstellende Handy verfügt dabei über ein überdurchschnittlich großes Display. Würde man die gleichen Interaktionen mit dem Benutzer eines Mobiltelefons

[2] mit freundlicher Genehmigung der Universität Karlstad, Schweden

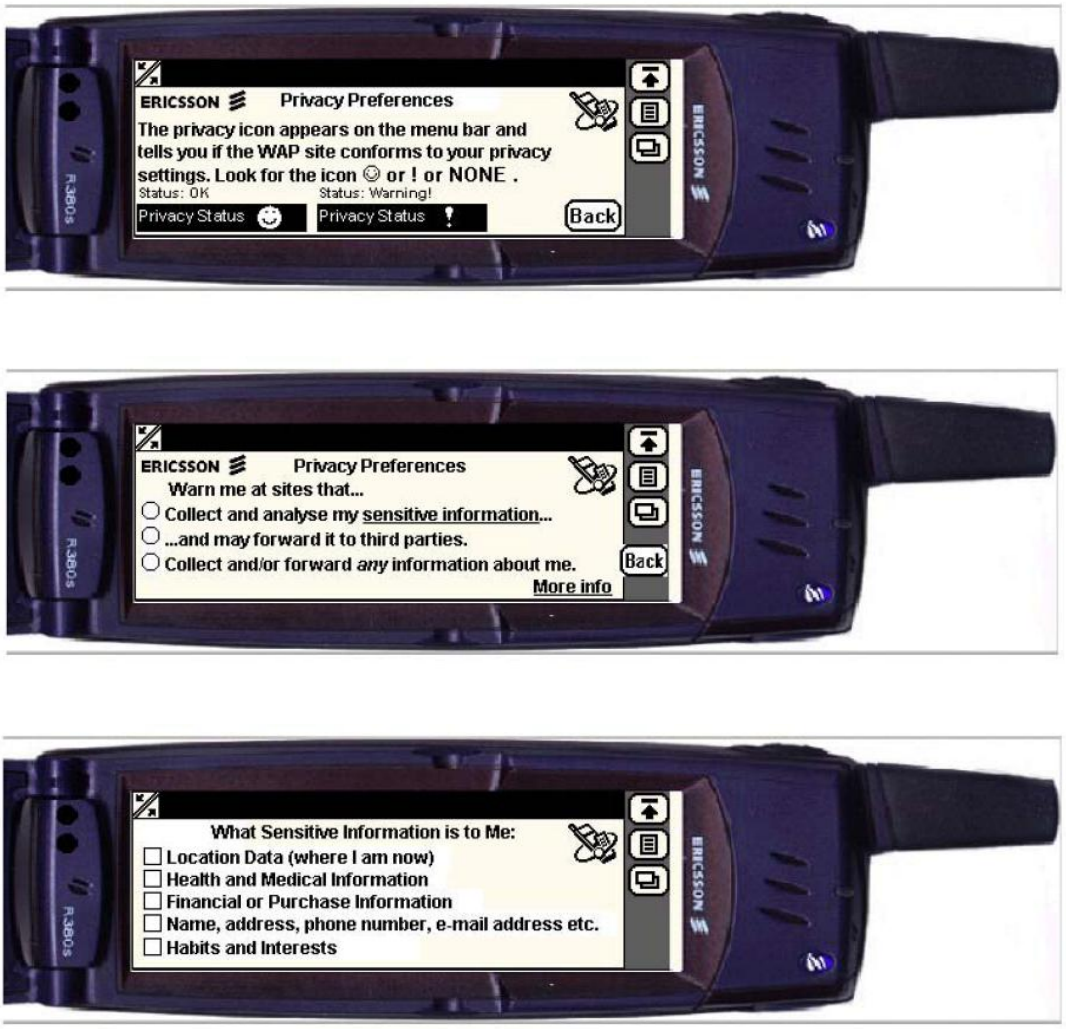

Abbildung 3.6: P3P in mobilen Geräten (aus [MÖ3, S. 32])

mit wesentlich kleinerem Display ausführen wollen, so würde die Benutzbarkeit des P3P-Standards auf diesem Gerät deutlich erschwert werden.

4 Trust Management in offenen, dezentralisierten und mobilen Systemen

Verteilte Systeme und Anwendungen, denen es an einer zentralen vertrauenswürdigen Autorität zur Überwachung des Systems, Authentifizierung von Teilnehmern und Validierung von Nachrichten und Informationen fehlt, werden als dezentralisiert bezeichnet. Ein dezentralisiertes System besteht demnach aus einer Menge von Entitäten oder Knoten (sogenannten *Peers*), die ohne das Vorhandensein einer zentralen Komponente miteinander interagieren. Die Kommunikation erfolgt dabei nachrichtenbasiert. Jeder Knoten versucht, seine eigenen individuellen Aufgaben und Ziele zu erfüllen, die nicht zwingenderweise einem globalen Systemziel dienlich sein müssen. Dies erreicht er durch Kooperation mit anderen Knoten. [Pac04]
In *offenen*, dezentralisierten Architekturen, in denen die teilnehmenden Knoten häufig und dynamisch wechseln können, kann die Teilnahme maliziöser Knoten nicht ausgeschlossen werden. Böswillige Knoten können absichtlich falsche oder irreführende Informationen publizieren. Da es keine zentrale Komponente (wie z.B. ein *Trust Center*) gibt, die in der Lage wäre, wahrheitsgetreue von verfälschten Informationen zu unterscheiden, fällt die Verantwortung zur Feststellung der Validität und Vertrauenswürdigkeit von Informationen und Kooperationspartnern daher lokal auf jeden einzelnen Knoten zurück. Dieses sogenannte *Trust Management* ist somit ein wesentlicher Aspekt offener, dezentralisierter Architekturen. Es ist eine besondere Herausforderung, die einzelnen Knoten so zu entwerfen, dass es jedem einzelnen von ihnen selbstverantwortlich möglich ist, die Authentizität der von anderen Knoten erhaltenen Informationen eigenständig bewerten zu können und somit die Vertrauenswürdigkeit der jeweiligen Knoten zuverlässig feststellen zu können. Dabei ist zu berücksichtigen, dass auch die Informationen, die den Einschätzungen zugrunde liegen, unvollständig oder verfälscht sein können. [Pac04]

In diesem Kapitel wird zunächst die Bedeutung des Begriffes *Vertrauen* für informationstechnische Systeme erörtert, da die Realisierung von Trust Management sowohl eine semantische, als auch eine syntaktische, quantifizierbare Modellierung von Vertrauen erfordert und das intuitive menschliche Verständnis dieses Begriffes hierfür nicht ausreichend ist. Aufbauend darauf werden bestehende Architekturen und Modelle zur Realisierung von Trust Management in dezentralisierten Systemen vorgestellt und auf ihre Tauglichkeit für den Einsatz in mobilen Umgebungen untersucht, da die Integration von Trust Management in eine Middleware für das Mobile Computing eines der Hauptziele der vorliegenden Arbeit ist.

4.1 Prinzipien des Vertrauens

Der Fortschritt in der drahtlosen Kommunikationstechnik im Hinblick auf eine flächendeckende Verfügbarkeit der Netze und höhere Datenraten ermöglicht mobilen Geräten eine drahtlose Anbindung nahezu an jedem Ort und zu jeder Zeit. Hierdurch sind Anwendungen auf mobilen Geräten in der Lage, dynamisch zahlreiche Dienste und Hosts ausfindig zu machen, mit denen sie interagieren könnten. Die Angst vor finanziellen, wirtschaftlichen oder sonstigen Schäden bei Durchführung bedeutender Transaktionen mit unbekannten, und somit potentiell schädlichen Entitäten, hemmt jedoch die Bereitschaft, Anwendungsfelder wie z.B. das *Mobile Commerce* oder mobiles Banking zu nutzen. Eine Inter- oder Transaktion kann riskant sein, wenn man sich über die Vertrauenswürdigkeit der Kooperationspartnernicht sicher ist. Dieses intrinsische Mißtrauen gilt sowohl für direkte Interaktionspartner, als auch für die Vermittler von Dienstleistungen und ist natürlich nicht nur auf mobile Systeme beschränkt. So muss man beispielweise bei der Bezahlung von Interneteinkäufen mit der eigenen Kreditkarte am heimischen PC seinem Dienstleister bereits vor dem Kauf vertrauen können, dass dieser die Kreditkartendaten nicht mißbraucht. In mobilen Ad-Hoc-Netzen ist das Risiko jedoch besonders hoch, da der Zugriff auf Informationen und Dienste einfacher ist und eine hohe Anonymität zwischen den einzelnen Kooperationspartnern durch eine höhere Fluktuation der verfügbaren Teilnehmer herrscht. Um eine flexible Ausführung nicht-trivialer Interaktionen zu ermöglichen, müssen die Risiken durch eine vertrauensbasierte Kooperation minimiert werden. Dies erfordert das Vorhandensein eines *Trust-Management-Frameworks*, welches in Abschnitt 4.4 eingeführt wird. [Cap04]

4.1.1 Definition von Vertrauen

Vertrauen ist ein Begriff, der in verschiedenen Wissenschaften mit teilweise abweichender Bedeutung verwendet wird. McKnight und Chervany [MC01] analysieren daher eine Reihe von Definitionen der Begriffe *Ver-* und *Mißtrauen* aus verschiedenen Disziplinen. Einige in der Literatur häufig zitierten Definitionen sollen hier erläutert werden:

- *"[Trust is] a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behavior of another."* [RBBC98, S. 393])

- *"[Trust is] the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party."* [MDS95, S. 710]

In beiden Definitionen von Vertrauen wird die (freiwillige) Bereitschaft einer Partei betont, das Risiko einer Verwundbarkeit durch eine zweite autonome Partei einzugehen,

in der Hoffnung, die Erreichung der eigenen Ziele durch Kooperation mit dieser Partei voranzutreiben. Besonders wichtig ist hierbei die positive Erwartungshaltung der vertrauenden Partei gegenüber dem Kooperationspartner, dass dieser keine schadhaften und böswilligen Absichten verfolgt. Wäre diese positive Erwartungshaltung nämlich nicht vorhanden, so wäre die Wahrscheinlichkeit eines tatsächlichen Schadensereignises zu hoch, um dieses Risiko einzugehen. Da der Kooperationspartner nicht gesteuert oder überwacht werden kann, kann er das in ihn investierte Vertrauen trotz positiver Erwartungshaltung der vertrauenden Partei enttäuschen.
Der Soziologe Diego Gambetta definiert Ver- und Mißtrauen wie folgt:

> *"[...] trust (or, symmetrically, distrust) is a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action. When we say we trust someone or that someone is trustworthy, we implicitly mean that the probability that he will perform an action that is beneficial or at least not detrimental to us is high enough for us to consider engaging in some form of cooperation with him. Correspondingly, when we say that someone is untrustworthy, we imply that that probability is low enough for us to refrain from doing so."* [Gam00, S. 217]

Die wichtigsten Kernaussagen aus Gambettas Definiton sollen hier nochmal hervorgehoben werden und mit den weiter oben angeführten Definitionen in Beziehung gebracht werden, um eine möglichst umfassende Vorstellung über den Begriff des Vertrauens zu erhalten:

- Vertrauen ist eine subjektive Wahrnehmung.
  Die Vertrauenswürdigkeit einer bestimmten Entität kann von jeder Partei anders eingestuft werden. Diese unterschiedlichen Einschätzungen können einerseits darauf beruhen, dass den Parteien voneinander abweichende Informationen über die besagte Entität bekannt sind, und sich andererseits auch auf die bislang gemachten, persönlichen Erfahrungen mit dieser Entität stützen. Der Aspekt der subjektiven Wahrnehmung von Vertrauen wird zwar in den weiter oben genannten Definitionen aus [RBBC98] und [MDS95] nicht berücksichtigt, ist aber besonders in dezentralisierten Trust-Management-Konzepten wichtig, da durch den Austausch subjektiver Einschätzungen über eine bestimmte Entität unter den Teilnehmern des Systems eine Approximation der tatsächlichen Vertrauenswürdigkeit dieser Entität erreicht werden kann. [Gam00]

- Wenn man eine Entität als vertrauenswürdig einstuft, glaubt man daran, dass das Verhalten und die Handlungen dieser Entität für einen selbst mit hoher Wahrscheinlichkeit vorteilhaft, zumindest aber nicht schädigend sein werden und somit die Voraussetzungen für eine eventuelle Kooperation geschaffen sind. Rousseau

[RBBC98] und Mayer [MDS95] beschreiben diesen Umstand als positive Erwartungshaltung.

- Das Verhalten und die Handlungen jener Entität, welcher Vertrauen entgegengebracht wird, können von der vertrauenden Partei nicht konstant beobachtet, geschweige denn überwacht oder gesteuert werden, da diese ja autonom ist. In diesem Punkt stimmen alle drei zitierten Definitionen überein. Wäre das Verhalten von Kooperationspartnern stets deterministisch oder der Kontrolle der kooperationsinitiierenden Partei unterworfen, so würde dies die Notwendigkeit von Vertrauen erübrigen. [Gam00], [RBBC98], [MDS95]

Capra [Cap04] erweitert Gambettas Definition um weitere Punkte, die besonders für Trust-Management-Konzepte in informationstechnischen Systemen wichtig sind:

- Vertrauen ist asymmetrisch.
  Zwei unterschiedliche Entitäten müssen sich gegenseitig nicht ähnlich oder gleich viel Vertrauen entgegenbringen. Asymmetrisches Vertrauen zwischen zwei Entitäten führt zum Aufbau asymmetrischer Vertrauensbeziehungen (siehe Abschnitt 4.1.2). [Cap04]

- Vertrauen ist kontextabhängig.
  Vertrauen in einer spezifischen Umgebung kann nicht bedingungslos auf andere Umgebungen übertragen werden. Vertrauen kann also an bestimmte Rahmenbedingungen geknüpft sein. [Cap04]

- Vertrauen ist dynamisch.
  Vertrauen wird aktuellen Wahrnehmungen und Erfahrungen angepasst. So wird das Vertrauen einer Entität in seine Interaktionspartner erhöht, wenn die Interaktion positive Effekte mit sich bringt. Analog wird das Vertrauen reduziert, wenn sich der Interaktionspartner als unzuverlässig oder böswillig erweist. [Cap04]

4.1.2 Vertrauensbeziehungen

Vertrauensbeziehungen (*Trust Relationships*) sollen einer Entität dabei behilflich sein, ihr Vertrauen, ihr Zutrauen und ihre Zuversicht in andere Entitäten modellieren und evaluieren zu können, um sich auf diese Weise so autonom wie möglich gegen maliziöse Entitäten absichern zu können und gegebenenfalls eine Kooperation mit ihnen zu verweigern. Vertrauensbeziehungen stützen sich auf die Vertrauenswerte, die den jeweiligen Entitäten zugeordnet werden. Um diese Vertrauenswerte maschinell verarbeitbar zu gestalten und ihre Vergleichbarkeit zu gewährleisten, ist es sinnvoll, Vertrauen durch numerische Werte zu beschreiben. Diese Quantifizierung von Vertrauen und eine Evaluation der erhaltenen Werte muss durch das verwendete Trust-Modell (siehe Abschnitt 4.3) ermöglicht werden. [SDET06]

Zusammenfassend kann man sagen, dass eine Vertrauensbeziehung formal gesehen zwischen genau zwei Entitäten besteht, asymmetrisch ist und nur bedingt transitiv (siehe Abschnitt 4.1.3) ist. Asymmetrische Vertrauensbeziehungen entstehen dann, wenn zwei Entitäten den eigens zugeordneten Vertrauenswert der jeweils anderen Entität durch die Metrik des verwendeten (gleichen) Trust-Modells unterschiedlich hoch einstufen. Dies kann dazu führen, dass eine Entität Kooperationsbereitschaft zeigt, da sie ihrem Gegenüber Vertrauen entgegen bringt, während die andere Entität die Kooperation aus Mißtrauen ablehnt. [ARH97]
In bestimmter Literatur zu kryptographisch orientierten Konzepten, die besonders in dezentralen Trust-Management-Systemen eingesetzt werden, wird der Aufbau einer Vertrauensbeziehung als gegenseitiger Austausch der öffentlichen Schlüssel der Kommunikationspartner verstanden. In diesem Fall ist mit dem Begriff *Vertrauen* der Glaube gemeint, mit welcher Wahrscheinlichkeit ein präsentierter Schlüssel tatsächlich zu einer angegebenen Identität gehört. Dieser Glaube wird in der Regel durch Zertifikate beeinflusst. Auf welche Weise die Verifizierung der Gültigkeit der Zertifikate erfolgen kann, wird durch die in Abschnitt 4.3 vorgestellten Trust-Modelle festgelegt. Vertrauen wird also in die Validität und Authentizität eines Schlüssel gesetzt und sagt nichts über das vergangene oder zukünftige Verhalten einer Entität aus. Die in den vorangegangenen Definitionen von Gambetta et al. betonte positive Erwartungshaltung ist vermutlich zwar implizit vorhanden, reicht aber ohne weiteres nicht unbedingt aus, um den Mißerfolg einer wichtigen Transaktion zu riskieren. In einem ganzheitlichen Ansatz können beide Konzepte einer Vertrauensbeziehung daher miteinander kombiniert werden. Dies wird beispielsweise in [SDET06] mit Hilfe der PACE-Architektur versucht, die in Abschnitt 4.4.1 vorgestellt wird.

4.1.3 Transitivität von Vertrauen

Vertrauen ist im Allgemeinen nicht transitiv [Jø96]. Die Annahme

(*Alice* vertraut *Bob*) & (*Bob* vertraut *Cathy*) => (*Alice* vertraut *Cathy*)

gilt nur, wenn bestimmte Bedingungen erfüllt sind. Abdul-Rahman und Hailes [ARH97] nennen dies *bedingte Transitivität* und beschreiben die notwendigen Bedingungen wie folgt:

- Bob teilt Alice sein Vertrauen in Cathy explizit mit. Er gibt Alice eine *Empfehlung* (*Recommendation*), Cathy ebenfalls zu vertrauen.
- Alice vertraut Bob als einer Entität, deren Empfehlungen zuverlässig sind und daher befolgt werden können.
- Alice ist in der Lage, die Qualität von Bobs Empfehlungen mit Hilfe eigener Informationen und Richtlinien zu beurteilen.

- Das Vertrauen ist nicht absolut. Alice kann Cathy aufgrund Bobs Empfehlung vertrauen, allerdings kann dieses Vertrauen geringer sein als Bobs Vertrauen in Cathy.

Das Konzept der Transitivität von Vertrauen wird besonders in Systemen mit einer großen Teilnehmerzahl und einer hohen Anonymität eingesetzt, um ein Netz des Vertrauens (*Web of Trust*) aufzubauen, da direktes Vertrauen zwischen je zwei Entitäten schlecht skalierbar ist.

4.2 Hindernisse beim Aufbau und Erhalt von Vertrauensbeziehungen

Um die Bedrohungen durch maliziöse Knoten im Netz neutralisieren zu können, müssen sogenannte *gute* Knoten angemessene Gegenmaßnahmen treffen. Eine potentiell wirksame Maßnahme ist es, Vertrauensbeziehungen zu seinen Kooperationspartnern aufzubauen und maliziöse Knoten von der Teilnahme auszuschliessen. Die Schwierigkeit hierbei liegt jedoch darin, zu erkennen, wer als schadhaft und wer als vertrauenswürdig eingestuft werden kann. Böswillige Knoten können versuchen, sich Vertrauen durch Manipulation zu erschleichen oder die Vertrauensbeziehungen anderer Knoten zu stören. Im folgenden sollen daher Gefährdungsfaktoren und Hindernisse beim Aufbau und Erhalt von Vertrauensbeziehungen dargestellt werden. Die Klassifizierung wurde aus [Pac04] übernommen.

Betrügerisches Auftreten

Böswillige Knoten können versuchen, ihre wahre Identität zu verschleieren und sich fälschlicherweise als andere Teilnehmer auszugeben, um sich bestehende Vertrauensbeziehung zwischen dem Knoten, dessen Identität sie vortäuschen und dem zu täuschenden Zielknoten zu Nutze zu machen. Der Zielknoten benötigt einen Mechanismus, um solche Vorfälle rechtzeitig aufdecken zu können. [SDET06]

Betrügerische Handlungen

Maliziöse Knoten können auch ohne Falschangaben bezüglich ihrer Identität oder ihrer Vertrauensbeziehungen in schadhafter Absicht handeln. So können sie anderen Teilnehmern Dienstleistungen anbieten, obwohl sie nicht in der Lage sind, diese tatsächlich zu erbringen. [SDET06]

Falschangaben über Vertrauensbeziehungen

Böswillige Knoten können falsche Angaben über ihre Ver- und Mißtrauensbeziehungen zu anderen Knoten machen. So könnte ein solcher Knoten seine ebenfalls böswilligen

Kooperationspartner nach außen hin als vertrauenswürdig deklarieren, um an wichtigen Transaktionen teilhaben zu können oder aber vertrauenswürdige Knoten gegenüber anderen Entitäten als nicht vertrauenswürdig einstufen, um ihnen zu schaden. Dies hätte nicht nur eine allgemeine Irreführung der Knoten zur Folge, sondern würde vor allem die Wahl der Kooperationspartner negativ beeinflussen. Dies kann sich wiederum negativ auf den Erfolg der Transaktion auswirken. [Pac04]

Betrügerische Absprache

Eine Gruppe maliziöser Knoten kann sich absprechen und zusammenschließen, um die Funktionsfähigkeit eines Systems zu stören. Dies könnte zum Beispiel in der Absicht geschehen, die eigenen Vertrauenswerte zu steigern und die der *guten* Knoten ausserhalb der böswilligen Gruppe zu senken. Ein Zusammenschluß von böswilligen Knoten kann ein Trust-Management-Konzept leichter umgehen oder manipulieren als ein einzelner böswilliger Knoten. [Pac04]

Angriff auf die Verfügbarkeit

Dieser Angriff wurde bereits in Abschnitt 3.1.5 erläutert. In dezentralisierten Systemen mit integriertem Trust-Management wird für diesen Angriff jedoch noch ein zusätzliches Motiv geliefert. Da man davon ausgehen kann, dass Knoten mit besonders hohen Vertrauenswerten auch die begehrtesten Kooperationspartner darstellen, könnten maliziöse Knoten diese mit dem Ziel angreifen, durch Störung ihrer Verfügbarkeit selbst an oberste Stelle der potentiellen Kooperationspartner zu rücken. Dieser Angriff ist besonders für kommerzielle Dienstanbieter schädlich, da ihnen viele Aufträge entgehen könnten, was wiederum zu wirtschaftlichem und finanziellem Schaden führen kann. [Pac04]

Systemstart und Eingliederung unbekannter Teilnehmer

Wenn ein System zum ersten Mal initialisiert wird oder wenn neue, bislang unbekannte Teilnehmer dynamisch beitreten können, so wird mit hoher Wahrscheinlichkeit der Umstand eintreten, dass keine Informationen bezüglich der Vertrauenswürdigkeit einiger Knoten vorliegen. Metriken und Algorithmen des verwendeten Trust-Management-Frameworks, die solch ein Vorwissen voraussetzen, würden in diesen Fällen blockieren. Daher sollte jeder Knoten in der Lage sein, sein eigenes Trust-Management-System initialisieren zu können (*Bootstrapping*), wobei unbekannte Knoten vorerst beispielsweise den niedrigsten Vertrauenswert zugewiesen bekommen, der später den Erfahrungen mit diesem Knoten angepasst wird. [SDET06]

Dynamische Anpassung von Vertrauenswerten

Das Verhalten und Benehmen eines Teilnehmers im System deutet auf den Grad seiner Vertrauenswürdigkeit hin. Werden gute Verhaltensweisen eines Teilnehmers erfahren, so

sollte sich das Vertrauen in ihn erhöhen, während es sich bei betrügerischem Verhalten entsprechend verringern sollte. Vertrauen sollte also den gegebenen Umständen und Erfahrungen dynamisch angepasst werden können, so dass möglichst aktuelle Vertrauenswerte als Entscheidungshilfe herbeigezogen werden können. Die Verwendung veralteter Werte kann die Auswahl unzuverlässiger oder böswilliger Kooperationspartner zur Folge haben. [Cap04]

Einbindung von Zusatzwissen

Wenn wichtige Informationen über systemexterne Kommunikationskanäle ausgetauscht werden können, so kann dies die Genauigkeit und Aussagekraft von Vertrauenswerten beeinflussen. Würde das Konzept zur Bestimmung von Vertrauenswerten nur durch die systeminternen Interaktionen bestimmt werden, so würden eventuell vorhandene, wertvolle externe Informationen einfach unberücksichtigt bleiben. Es sollte also möglich sein, externes Zusatzwissen in sinnvoller Weise in die Bestimmung der Vertrauenswürdigkeit von Knoten einfließen lassen zu können. [SDET06]

Auf welche Weise die hier genannten Hindernisse überwunden und wie vor allem die letzten drei Punkte realisiert werden, muss durch das verwendete Trust-Management-Konzept und das dazugehörige Trust-Modell geregelt werden, die im folgenden behandelt werden.

4.3 Trust-Modelle

Trust-Modelle oder auch Vertrauensmodelle geben vor, auf welche Art und Weise einer Entität Vertrauen zugeordnet werden kann, wie und zwischen welchen Entitäten Vertrauensbeziehungen entstehen können und wie die notwendigen Vertrauenswerte berechnet und evaluiert werden. Es muss ebenfalls spezifiziert werden, woher diese Informationen über die Vertrauenswürdigkeit einer Entität bezogen werden sollen, und es muss gegebenenfalls ein entsprechendes Protokoll zum Einholen von Meinungen, Reputationen und Vertrauensinformationen anderer Teilnehmer über eine bestimmte Entität bereitgestellt werden.

4.3.1 Klassifizierung

Es gibt drei verschiedene Vertrauensmodelle, mit denen beispielsweise Vertrauen in die Gültigkeit von Zertifikaten etabliert werden kann. Welches Modell wann eingesetzt wird ist abhängig von der Infrastruktur des betreffenden Systems oder Rechnernetzes und dem jeweiligen Anwendungsfall.

Direktes Vertrauen

Direktes Vertrauen ist das einfachste Vertrauensmodell. Ein Benutzer kann z.B. von der Gültigkeit eines Schlüssel überzeugt sein, weil er die Herkunft dieses Schlüssels genau kennt. Direktes Vertrauen tritt auch in komplexeren Vertrauensmodellen häufig auf. So vertraut ein Benutzer direkt auf die Authentizität der in seinem Webbrowser vorinstallierten Zertifikate einer Root Certification Authority, weil diese vom Hersteller mitgeliefert worden sind. [NA99]

Hierarchisches Vertrauen

In einem hierarchischen System gibt es eine Menge von Root-Zertifikaten, die auch als Vertrauensanker bezeichnet werden und öffentlich anerkannt sind (z.B. von der VeriSign-Zertifizierungsstelle). Diese Vertrauensanker stehen in der Vertrauenshierarchie an oberster Stelle. Ab hier entsteht ein sogenannter Vertrauensbaum. Mit den Schlüsseln der Root-Zertifikate können weitere Zertifikate signiert werden, deren Schlüssel ihrerseits hierarchisch untergeordnete Zertifikate signieren. So entsteht ein hierarchischer Vertrauenspfad, der zur Verifizierung solange rückwärts verfolgt werden muss, bis ein Root-Zertifikat gefunden wird, dem direktes Vertrauen entgegengebracht wird. [NA99]

Web of Trust

Ein Netz des Vertrauens, allgemein auch als *Web of Trust* bekannt, beschreibt transitive Vertrauensbeziehungen. Um abgestufte Vertrauenslevel festlegen zu können, speichert jeder Benutzer in seinem Public-Key-Ring zusätzlich zu jedem fremden öffentlichen Schlüssel einen Eintrag, der den Grad seines eigenen Vertrauens in die Authentizität des entsprechenden Schlüssels beschreibt (vgl. Abschnitt 4.4.2). Desweiteren können jedem dieser Schlüssel zusätzlich Signaturen anderer Benutzer zugeordnet sein, die ebenfalls das gewichtete Vertrauen dieser Benutzer in die Gültigkeit der Schlüssel widerspiegeln. In einem Web of Trust kann also jeder Teilnehmer die Rolle einer Zertifizierungsstelle übernehmen. Aus der Summe des eigenen Vertrauens in einen bestimmten Schlüssel sowie der jeweils angegebenen Vertrauenslevel anderer Benutzer in diesen Schlüssel und des eigenen Vertrauens in die zertifizierenden Benutzer kann dann mit Hilfe einer Evaluierungsfunktion ein Vertrauenswert ermittelt werden, der dem Schlüsseleintrag zugeordnet wird. Ob die einzelnen zertifizierenden Benutzer vertrauenswürdig sind, kann also durch Rekursion oder beispielsweise auch mit Reputationssystemen festgestellt werden. [Eck06]

4.3.2 Reputationssysteme

Reputationsbasierte Trust-Management-Systeme ermöglichen es einer Entität, die Vertrauenswürdigkeit eines anderen Teilnehmers mit Hilfe relevanter Reputationsinformationen zu evaluieren, die sie im Vorfeld gesammelt hat. Diese Informationen kann man

sich als Bewertungen aus und nach früheren Interaktionen vorstellen. Resnick et al. definieren Reputationssysteme wie folgt:

> *"A reputation system collects, distributes and aggregates feedback about participants' past behavior."* [RZFK00, S. 46]

Dafür gibt es verschiedene Herangehensweisen: In *XRep* muss ein Teilnehmer A die Reputationsinformationen eines Teilnehmers B aktiv von anderen Teilnehmer des Systems anfordern [DVP+02]. In *NICE* ist es hingegen so, dass jeder Teilnehmer seine von früheren Interaktionspartnern erhaltenen Bewertungen selbst als Cookies speichert und verwaltet, und sie einem potentiellen Interaktionspartner eigenständig als Beweis seiner Vertrauenswürdigkeit präsentiert. Dabei ist besonderer Wert darauf zu legen, dass die Cookies nicht manipuliert werden können [SLB06]. Andere Vertrauensmodelle wie beispielweise *NodeRanking* oder auch *REGRET* lassen zusätzlich die Einbindung bestehender sozialer Beziehungen zur Evaluierung der Reputation eines Teilnehmers zu [SS01]. Desweiteren kann unterschieden werden, ob nur positives oder nur negatives Feedback erstellt und gesammelt, oder aber eine Mischform verwendet werden soll.

4.3.3 Zugriffsbasierte Systeme

In zugriffsbasierten Systemen geht es darum, den Zugriff auf bestimmte Ressourcen und Dienste zu restriktieren und den Zugriff nur berechtigten Entitäten zu gewähren. In zugriffsbasierten Systemen wie z.B. *Policymaker* oder *KeyNote* werden daher sogenannte *Credentials* (Berechtigungsnachweise) und anwendungsspezifische Richtlinien benutzt, um den Zugriff auf Dienste und Ressourcen zu regulieren. Dabei muss der ressourcenanfordernde Benutzer dem Ressourcenverwalter seine Credentials präsentieren, während dieser die Credentials mit den Anforderungen der anwendungsspezifischen Policies abgleicht, um zu entscheiden, ob ein Zugriff auf die angeforderten Ressourcen genehmigt werden kann. [BFL96]

4.4 Dezentrale Trust-Management-Konzepte

Anders als in Systemen mit einer zentralen Autorität, die sich um die Koordination der Aktivitäten jeder Entität im System kümmert, ist das Verhalten von Entitäten in dezentralisierten Systemen völlig autonom. In solchen Systemen ist jede Entität auf die Interaktion mit anderen Entitäten angewiesen, um die eigenen individuellen Ziele erreichen zu können. Da auch böswillige Entitäten offenen Systemen jederzeit beitreten können, stellt dies ein potentielles Risiko für den Erfolg einer Kooperation dar. Die im folgenden vorgestellten dezentralen Trust-Management-Konzepte unterstützen die Entitäten dabei, Vertrauensbeziehungen untereinander aufzubauen und die Vertrauenswürdigkeit anderer Teilnehmer des System zu prüfen.

4.4.1 Die PACE-Architektur

Im Forschungsbereich des Trust Managements fehlt bislang ein ganzheitlicher Ansatz, der die Integration von Komponenten zur Modellierung und Evaluation von Vertrauen als festen Bestandteil der internen Architektur von Entitäten realisiert, die Teilnehmer eines dezentralisierten, offenen Systems sind. Die Bereitstellung eines Trust-Management-Modells, welches es ermöglicht, anderen Teilnehmern im System Vertrauenswerte zuzuordnen, sollte nicht nur Sache der Anwendungen sein, die auf den einzelnen Peers ablaufen. Vielmehr sollten die Vertrauensinformationen auf jeder internen Schicht eines Peers, also innerhalb seiner gesamten Architektur verfügbar sein, um als Entscheidungshilfe dienen zu können. Daraus resultieren Anforderungen bezüglich der Eingliederung von Trust-Management-Komponenten in die interne Architektur von Entitäten. Erenkrantz et al. haben versucht, diese Anforderungen eines dezentralisierten Trust Managements zu diagnostizieren und sie in einem geeigneten Modell, nämlich der von ihnen entwickelten PACE-Architektur, zur Erfüllung zu bringen. PACE stellt eine strukturierte und detaillierte Anleitung dar, wie Trust-Modelle innerhalb von Entitäten, die Teilnehmer eines dezentralisierten Systems sind, eingebettet werden können. Dabei ist die Architektur durch Kapselung der Komponenten vom Einsatz eines bestimmten Trust-Modells unabhängig und bietet daher ein hohes Maß an Flexibilität, da das Trust-Modell passend zum jeweiligen Anwendungsfall gewählt werden kann. [SDET06]

Die Grundideen

PACE ist das Akronym für *Practical Architectural approach for Composing Egocentric trust* und ist ein Modell, welches sich speziell damit beschäftigt, welche Komponenten in eine Entität integriert und wie sie angeordnet werden müssen, sowie wie sie untereinander interagieren sollten, um Trust Management in dezentralisierten Systemen zu ermöglichen. Die folgenden Punkte fassen daher die wichtigsten Entwurfsideen von PACE zusammen:

- PACE ist eine ereignisbasierte Architektur, die in Schichten organisiert ist. Wenn man davon ausgeht, dass die Anwendungsebene die unterste Schicht darstellt, so kennen die Komponenten einer Schicht nur Komponenten der grafisch jeweils höher angeordneten Schicht, nicht aber Komponenten der grafisch darunterliegenden Schichten (vgl. Abbildung 4.1). [SDET06]

- Die Komponenten kommunizieren untereinander asynchron unter Benutzung zweier Nachrichtentypen: Anfragen (*requests*) und Benachrichtigungen (*notifications*). [SDET06]

- PACE identitfiziert Entitäten im System mit Hilfe digitaler Identitäten. Dabei ist es prinzipiell erlaubt, dass ein Benutzer über mehrere elektronische Identitäten verfügt (z.B. eine Identität pro mobilem Gerät), und somit mehrere Entitäten in einem System darstellen kann. Die Vertrauensinformationen über jede vorhandene digi-

tale Identität werden in erster Linie einzeln festgelegt und verwaltet, unabhängig von der tatsächlichen physischen Identität, die repräsentiert wird. [SDET06]

- PACE unterscheidet strikt zwischen den internen Wissen und Glauben eines Peers und den Informationen, die von außen durch andere Peers herangetragen werden. Diese Unterscheidung ist wichtig, da externe Informationen von anderen Peers potentiell falsch oder unvollständig sein können. PACE unterteilt die interne Datenhaltung daher in eine interne und eine externe Speicherkomponente für Informationen. [Pac04]

- Um die Vergleichbarkeit von Vertrauenswerten zu erleichtern, fordert PACE eine numerische Repräsentation dieser Werte, ohne jedoch diesbezüglich semantische Beschränkungen zu machen. [Pac04]

- PACE sieht vor, dass vorhandenes Ver- oder Mißtrauen zwischen Peers nicht nur lokal in einer Komponente innerhalb eines Peers, sondern allen seinen internen Komponenten bekannt ist, damit jede dieser Komponenten bei lokalen Entscheidungen von diesem Wissen profitieren kann. Vertrauensbeziehungen sollten also einerseits nach außen hin für andere Peers veröffentlicht werden, können aber andererseits auch für interne Komponenten eines Peers wahrnehmbar sein. [SDET06]

Schichten-Architektur in PACE

PACE unterteilt die Architektur eines Peers in vier Schichten : *Communication*, *Information*, *Trust* und *Application*. In Abbildung 4.1 sind die einzelnen Schichten samt ihrer Komponenten dargestellt und werden nachfolgend erläutert. Die Communication-Schicht ist für die Kommunikation der Peers untereinander verantwortlich. Sie besteht aus den drei Komponenten *Protocol Handler*, *Communication Manager* und dem *Signature Manager*. Der Communication Manager ist für die dynamische Erstellung von Protocol Handlern zuständig, während der Signature Manager für die Signierung von Requests und die Verifizierung von Notifications verantwortlich ist. Ausgehende Nachrichten werden dabei mit dem lokalen privaten Schl üssel des Peers digital signiert, und die Signatur samt passendem öffentlichen Schlüssel in die Nachricht eingebettet, um unabhängig von der Unterstützung digitaler Signaturen durch die benutzten Transportprotokolle zu sein. Der Signature Manager des Empfängers der Nachricht kann den darin enthaltenen öffentlichen Schl üssel benutzen, um die Echtheit der Nachricht zu überprüfen. [SDET06]

Die Information-Schicht ist für die lokale Datenhaltung innerhalb eines Peers zuständig. Um zwischen internen und externen, also von anderen Peers gesammelten Informationen, unterscheiden zu können, besteht diese Schicht aus zwei Komponenten, nämlich der internen Informationskomponente und der externen Informationskomponente. Die interne Informationskomponente speichert unter anderem Request-Nachrichten, die

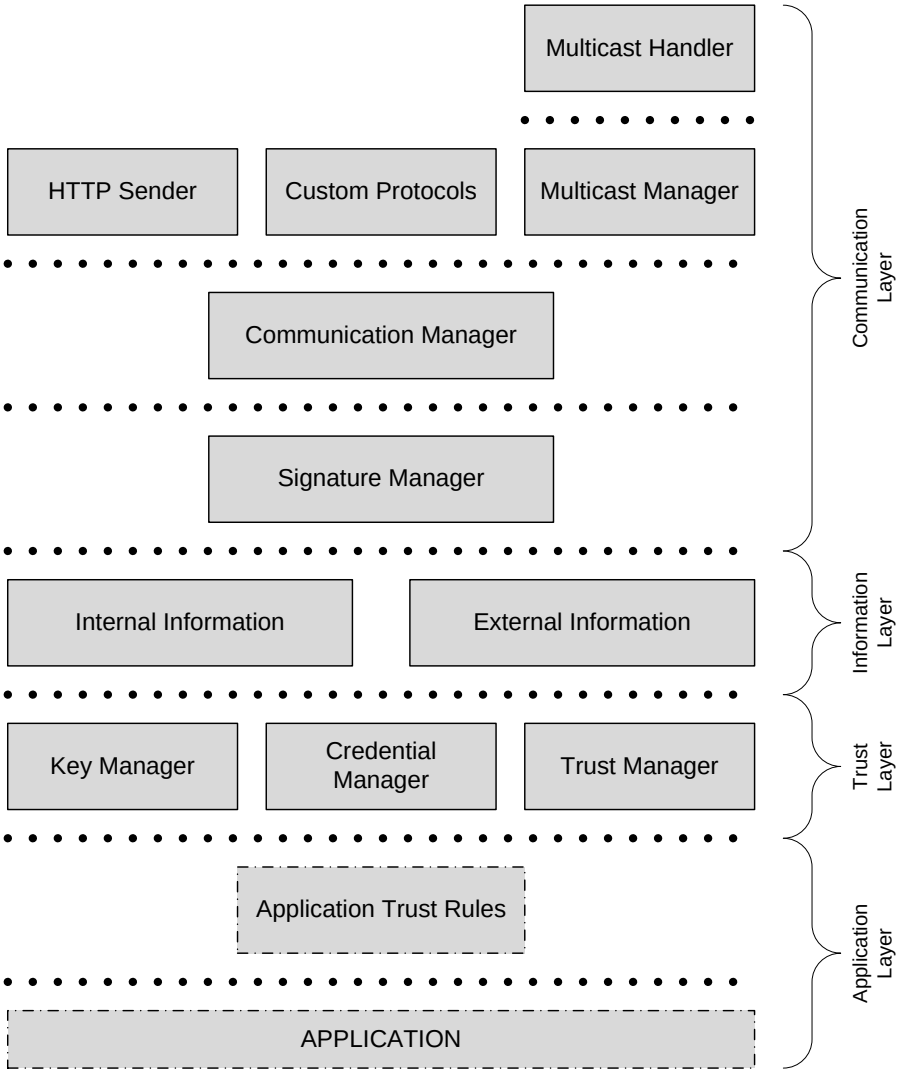

Abbildung 4.1: PACE-Architektur (nach [SDET06, S. 54])

von internen Komponenten des Peers stammen, während die externe Informationskomponente Notification-Nachrichten von anderen Peers speichert. Die Daten der internen Informationskomponente werden persistent gespeichert, um einen Überblick über frühere Aktionen und Informationen zu behalten. Diese Komponente kann dabei nur durch Requests modifiziert werden. Auch Datenanfragen sind nur über Request-Nachrichten möglich, um die internen Informationen vor einer ungewollten Verbreitung an andere Peers zu schützen. Die Information-Schicht ist auch dafür zuständig, Requests zur Übertragung an den Communication Manager der Communication-Schicht weiterzuleiten. [SDET06]

Der Trust Layer enthält Komponenten, die das eigentliche Trust Management ermöglichen sollen. Diese Schicht besteht aus dem *Key Manager*, dem *Credential Manager* und dem *Trust Manager*. Der Key Manager ist für die Erstellung von asymmetrischen Schl

üsselpaaren und zur Speicherung dieser in der internen Informationskomponente verantwortlich. Der Credential Manager verwaltet die Berechtigungsnachweise und Zertifikate anderer Peers, die in der internen Informationskomponente bekannt sind. Er ist zusätzlich dafür zuständig, fehlende notwendige öffentliche Schlüssel anzufordern, oder Schlüssel durch den Erhalt von entsprechenen Notifications von der weiteren Verwendung auszuschließen (*key revocation*). Der Trust Manager kann verschiedene Trust- und Reputation-Modelle kapseln, die in Abschnitt 4.3 vorgestellt werden. Er ist dafür zuständig, Vertrauenswerte zu berechnen, um die Vertrauenswürdigkeit erhaltener Nachrichten bzw. deren Absender abschätzen zu können. [SDET06]

Die Application-Schicht kapselt anwendungsspezifische Komponenten. Sie enthält die sogenannten *Application Trust Rules* und die eigentliche Anwendungskomponente. Die Application Trust Rules kapseln die Regeln, die verwendet werden sollen, um Nachrichten Vertrauenswerte auf Basis ihrer anwendungsspezifischen semantischen Bedeutung zuordnen zu können. Dabei unterstützen sie verschiedene Dimensionen von Vertrauensbeziehungen. Die Anwendungskomponente repräsentiert das lokale Verhalten eines Peers auf Anwendungsebene und kann auch Benutzerschnittstellen umfassen. [Pac04]

PACE beschäftigt sich nur mit den Vertrauensbeziehungen zwischen einzelnen Peers, nicht jedoch zwischen den einzelnen Komponenten eines Peers, da dieses Vertrauen als intrinsisch vorausgesetzt wird. PACE vertraut grundsätzlich nur den eigenen internen Informationen und Erkenntnissen und mißtraut vorerst externen Mitteilungen. PACE fordert nicht, dass alle interagierenden Peers, in einer dezentralisierten Anwendung das PACE-Modell implementieren. Auch Nicht-PACE-Entitäten können mit PACE -Entitäten interagieren. Desweiteren stellt PACE keine über die PACE-Funktionalität hinaus weiteren Einschränkungen für die interne Architektur eines Peers. [SDET06]

Da PACE nicht mit dem Anspruch entwickelt wurde, in mobilen Systemen eingesetzt zu werden und auch die Implementierung in Java den Anforderungen mobiler Geräte nicht gerecht wird, wird im Konzeptteil dieser Arbeit (Kapitel 5) geprüft werden müssen, in wie weit die PACE-Architektur oder einzelne PACE-Komponenten f ür den Einsatz innerhalb mobiler Middleware angepasst werden k önnen.

4.4.2 Pretty Good Privacy

Pretty Good Privacy oder kurz PGP ist ein von Phil Zimmermann entwickeltes Konzept zur Verschlüsselung und Authentifizierung von ausgetauschten Daten zwischen Entitäten, welches sich besonders beim Versand von E-Mail-Nachrichten über das Internet durchgesetzt hat. Das Besondere an PGP ist, dass es dezentralisiert nach dem Vorbild des Web-of-Trust eingesetzt werden kann und somit eine Alternative zu der als Public-Key-Infrastruktur bekannten Lösung mit Zertifikaten nach dem X.509v3-Standard darstellt,

bei denen zentrale Registrierungs- und Zertifizierungsstellen benötigt werden.
Anlass der Entwicklung von PGP war das vom US-Senat im Jahre 1991 erlassenes Gesetz Nr. 266, das vorsah, dass jede Verschlüsselungssoftware eine eingebaute Hintertür enthalten müsse, damit bei Bedarf ein staatlicher Zugriff erfolgen könne. Dieses Gesetz wurde zwar nicht verabschiedet, aber da die Regierung weiter an ähnlichen Vorhaben arbeitete, war dies eine Motivation für Zimmermann, die Rechte der Bürger in Bezug auf ihre Privatsphäre gegenüber dem Staat zu verteidigen, und privat ausgetauschte Nachrichten vor unbefugtem Zugriff[1] wie z.B. dem Geheimdienst zu schützen. [Sch05]

Da Middleware-Plattformen im Mobile Computing h äufig asynchron und nachrichtenbasiert durch sogenannte *Events* kommunizieren, und die Integration von Sicherheits- und Vertrauensaspekten in diese Middleware-Plattformen möglichst ohne die Notwendigkeit einer zentralen Komponente erfolgen soll, können einige Konzepte und Verfahren aus PGP eventuell übernommen und an die speziellen Anforderungen in mobilen Systemen angepasst werden. Daher soll die grundlegende Funktionsweise von PGP nachfolgend betrachtet werden. Die dem zugrundeliegenden mathematischen und kryptographischen Verfahren werden dabei nicht detailliert behandelt, es sei denn, es kommt ihnen eine besonders hervorzuhebende Bedeutung im Hinblick auf die Verwendbarkeit auf mobilen Geräten zu.

Die drei wichtigsten von PGP bereitgestellten Dienste sind Authentifizierung, Gewährleistung von Vertraulichkeit und Kompression. Diese werden durch ein hybrides Verfahren aus einer Kombination von asymmetrischen und symmetrischen Verschlüsselungsalgorithmen realisiert. Dafür erzeugt sich zunächst jeder Teilnehmer selbst ein eigenes asymmetrisches Schlüsselpaar, um die genannten Dienste in Anspruch nehmen zu können. [Sch05]

Authentifizierung

Die Authentifizierung geschieht durch die Erstellung digitaler Signaturen für Nachrichten auf Senderseite und der Verifikation dieser Signaturen auf Empfängerseite. Dabei müssen folgende Schritte durchlaufen werden [Sta03]:

1. Der Sender erstellt eine Nachricht.

2. Er berechnet einen Hashwert über diese Nachricht, z.B. mit SHA-1 (vgl. Abschnitt 3.2.3).

3. Der Hashwert wird mit dem RSA-Algorithmus (vgl. Abschnitt 3.2.2) und dem privaten Schlüssel des Senders verschlüsselt. Das Ergebnis wird als digitale Signatur

[1] Aus Zimmermanns Sicht ist jede Entität, die weder Sender noch Empfänger der Nachricht ist, zum Zugriff auf die darin enthaltenen Informationen unbefugt.

bezeichnet und wird der originalen Nachricht vorangestellt. Die Nachricht kann nun versendet werden.

4. Der Empfänger entschlüsselt die digitale Signatur, indem er RSA mit dem öffentlichen Schlüssel des Senders verwendet und erhält so den Hashwert der empfangenen Nachricht.

5. Der Empfänger berechnet einen eigenen Hashwert für die Nachricht unter Verwendung des gleichen Hash-Algorithmus wie der Sender und vergleicht diesen Wert mit dem entschlüsselten Hashwert. Stimmen beide überein, so wird die Nachricht als authentisch betrachtet.

Die Kombination aus SHA-1 und RSA ist ein in der Praxis h äufig anzutreffendes Schema, trotz der Tatsache, dass bereits erfolgreiche Kollisionsangriffe gegen SHA-1 durchgef ührt wurden. PGP unterstützt auch sogenannte alleinstehende, losgelöste digitale Signaturen, die unabhängig von den Nachrichten, die sie signieren, gespeichert und übertragen werden können. Besonders nützlich ist diese Eigenschaft, wenn mehrere Entitäten ein Dokument oder ein Zertifikat signieren wollen, und jede Entität ihre Signatur auf das reine Dokument anwenden kann und keine bereits vorhandenen Signaturen anderer Entitäten mitsignieren muss. Dies vermeidet eine komplizierte Verschachtelung der Signaturen. [Sta03]

Vertraulichkeit

Vertraulichkeit wird bei PGP durch Verschlüsselung von Nachrichten erreicht. Dies kann sowohl für Nachrichten genutzt werden, die zur Übertragung gedacht sind, als auch für solche, die zur lokalen Speicherung als Datei vorgesehen sind. In beiden Fällen wird ein symmetrischer Verschlüsselungsalgorithmus benutzt. Damit hierbei nicht das typische Problem der Schlüsselverteilung entsteht, wird in PGP jeder symmetrische Schlüssel nur ein einziges Mal zur Ver- und Entschlüsselung genau einer Nachricht verwendet. Es wird pro Nachricht also ein zufälliger symmetrischer Schlüssel vom Sender erzeugt, der zusammen mit der symmetrisch verschlüsselten Nachricht versendet wird. Vor der Übertragung muss der symmetrische Schlüssel jedoch noch geschützt werden. Dies geschieht, indem er mit dem öffentlichen Schlüssel des Empfängers verschlüsselt wird. Der Ablauf sieht also wie folgt aus [Sta03]:

1. Der Sender erstellt eine Nachricht und generiert einen zufälligen symmetrischen Schlüssel zur einmaligen Verwendung für diese Nachricht.

2. Die Nachricht wird mit Hilfe des Schlüssels symmetrisch verschlüsselt. Als Algorithmen kommen dabei z.B. CAST-128, 3DES und AES in Frage.

3. Der symmetrische Schlüssel wird mit RSA und dem öffentlichen Schlüssel des Empfängers verschlüsselt und der Nachricht beigefügt.

4. Der Empfänger benutzt RSA und seinen privaten Schlüssel zur Gewinnung des symmetrischen Schlüssels.

5. Die Nachricht kann nun mit dem symmetrischen Schlüssel und dem verwendeten symmetrischen Algorithmus wieder entschlüsselt werden.

Als Alternative zu RSA können auch asymmetrische Algorithmen wie Diffie-Hellmann oder ElGamal verwendet werden. Der Einsatz einer Kombination aus symmetrischer und asymmetrischer Algorithmen hat zwei wichtige Vorteile:

- Der zeitliche und rechnerische Aufwand bei der Verschlüsselung wird minimiert. Dies liegt daran, dass die symmetrische Verschlüsselung um ein Vielfaches schneller und weniger rechenaufwendig ist als die asymmetrische Verschlüsselung, und somit weniger Ressourcen in Anspruch nimmt. Dies ist vor allem beim Einsatz von Verschlüsselung auf mobilen Geräten ein wichtiges Auswahlkriterium für Algorithmen. [Sch05]

- Der Einsatz von Public-Key-Algorithmen mit privaten und öffentlichen Schlüsselpaaren löst das Schlüsselverteilungsproblem für die symmetrische Verschlüsselung. Das erspart den Einsatz eines *Session-Key-Exchange*-Protokolls und die dadurch wegfallende Kommunikation. [Sch05]

In PGP können die beiden oben beschriebenen Funktionen der Authentifizierung und der Vertraulichkeit auch sequentiell auf die gleiche Nachricht angewandt werden, um sich gegenseitig zu ergänzen. Als erstes wird dabei die digitale Signatur für die Klartext-Nachricht generiert und dem Klartext beigefügt. Dieses Ergebnis wird dann symmetrisch verschlüsselt. Der verwendetete symmetrische Schlüssel wird nun selbst asymmetrisch verschlüsselt und an die Nachricht angehängt. Um die korrekte Interpretation der Nachrichtenfelder zu gewährleisten, stellt PGP ein eigenes Nachrichtenformat zur Verfügung (siehe Abbildung 4.2). [Sta03]

Kompression

Per Standardeinstellung komprimiert PGP Nachrichten mit dem ZIP-Algorithmus. Dies hat eine ressourcensparende Wirkung sowohl bei der Übertragung der Nachrichten, als auch bei einer eventuellen Speicherung. Die Kompression geschieht nach der Erstellung der digitalen Signatur, jedoch vor der Verschlüsselung der Nachricht. Diese Reihenfolge hat unter anderem den Vorteil, dass die kryptographische Sicherheit der Verschlüsselung verstärkt wird, da die komprimierte Nachricht über weniger redundante Informationen als das Original verfügt. [Sta03]

Kryptographische Schlüssel in PGP

PGP verwendet vier Arten von Schlüsseln: symmetrische Einweg-Schlüssel für jede Nachricht, asymmetrische öffentliche und private Schlüssel sowie Passwort-basierte symme-

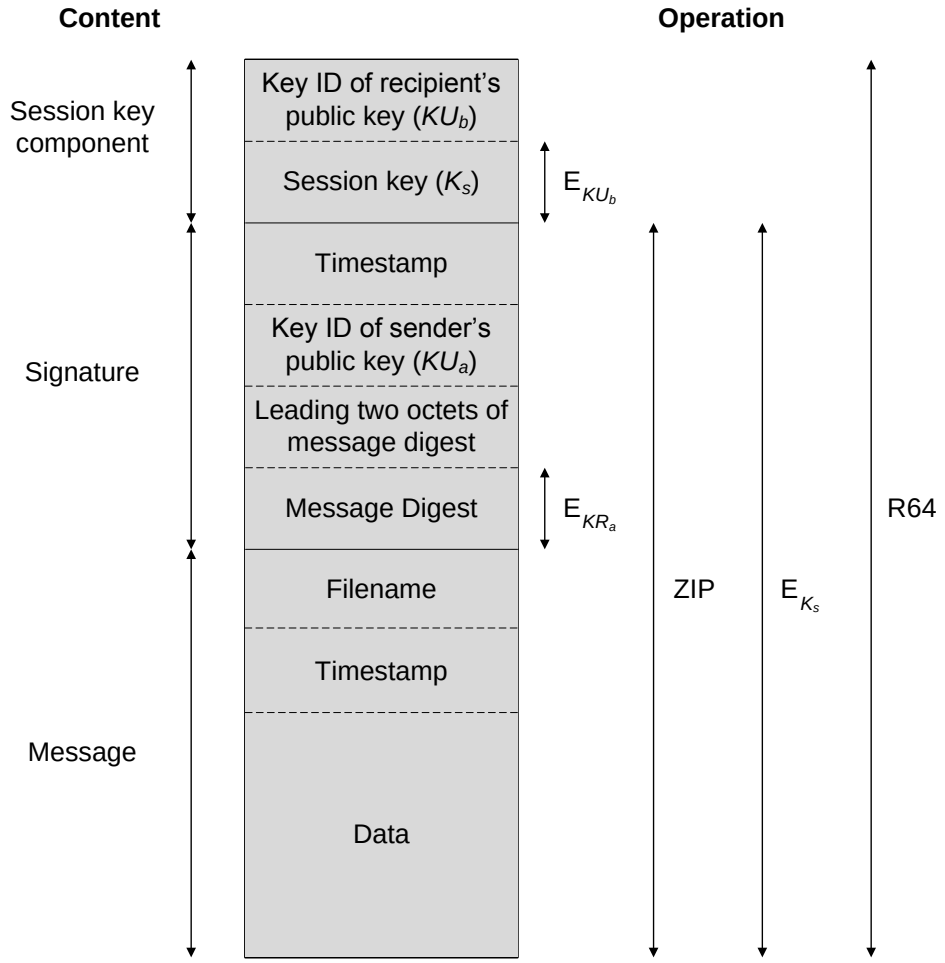

Abbildung 4.2: Das PGP-Nachrichtenformat (nach [Sta03, S. 446])

trische Schlüssel. Letztere sind zur sicheren verschlüsselten Speicherung des privaten Schlüssels eines Benutzers gedacht, der das Passwort bei Zugriff auf diesen eingeben muss. Diese Schlüssel müssen drei unterschiedlichen Anforderungen gerecht werden [Sta03]:

- Die Zufallsgenerierung der symmetrischen Schlüssel muss unvorhersehbar sein.
- Einem Benutzer soll es möglich sein, über mehrere asymmetrische Schlüsselpaare zu verfügen. So könnte er beispielsweise bei der geschäftlichen Kommunikation mit seinem Arbeitgeber einen anderen Schlüssel verwenden als bei der Kommun-

kation mit seinen Vereinskameraden. Da es keine 1:1-Zuordnung eines einzigen Schlüsselpaares zu einem Benutzer gibt, muss es einen Erkennungs- oder Benachrichtigungsmechanismus geben, welcher Schlüssel wann einzusetzen ist.

- Jeder PGP-Teilnehmer muss sich selbst um das Schlüsselmanagement kümmern. Dabei müssen sowohl eigene Schlüsselpaare als auch die öffentlichen Schlüssel anderer Entitäten verwaltet werden.

Key Identifier Um den symmetrischen Schlüssel zur Entschlüsselung einer Nachricht wiederzugewinnen, muss der Empfänger unbedingt wissen, welcher seiner eigenen öffentlichen Schlüssel zur Verschlüsselung des symmetrischen Schlüssels benutzt worden ist. Eine einfache Lösung, ihm dies mitzuteilen, wäre es, wenn der Sender den verwendeten Public Key des Empfängers zusammen mit der Nachricht verschickt. Der Empfänger könnte dann überprüfen, ob es sich tatsächlich um einen seiner eigenen Schlüssel handelt und mit der Verarbeitung der Nachricht fortsetzen. Diese Lösung ist allerdings sehr aufwendig, wenn man bedenkt, dass ein öffentlicher RSA-Schlüssel 1024 bit und mehr lang sein kann. Die in PGP gefundene Lösung verwendet daher sogenannte *Key Identifier*. Dabei wird jedem öffentlichen Schlüssel eine *Key ID* zugeordnet, die aus seinen 64 niedrigwertigsten Bits besteht und mit hoher Wahrscheinlichkeit eindeutig in Kombination mit der *Key-Owner*-Identität ist. Die Key ID eines Schlüssel K berechnet sich also aus $(K \bmod 2^{64})$. [NA99]
Key Identifier werden auch zur Verifikation von digitalen Signaturen benötigt, da ein Sender mehrere private Schlüssel zur Verfügung haben kann und dem Empfänger daher mitteilen muss, welchen öffentlichen Schlüssel dieser verwenden soll. Dieses Konzept der relativ kurzen Key Identifier im Vergleich zur originalen Schlüssellänge eignet sich auch gut zum Einsatz in mobilen Umgebungen, um den Kommunikationsoverhead so gering wie möglich zu halten und Speicherplatz zu sparen. Alternativ zu Key Identifiern können auch *Fingerprints* verwendet werden, die jedem Schlüssel einen Hashwert zuordnen, über den der betreffende Schlüssel identifiziert werden kann. [Sch05]

Schlüsselhaltung Um einen effizienten Zugriff auf benötigte Schlüssel zu ermöglichen, müssen Datenstrukturen zur systematischen Speicherung der Schlüssel bereitgestellt werden. In der als Private-Key-Ring bezeichneten Komponente speichert jede Entität ihre eigenen asymmetrischen Schl üsselpaare, während in dem sogenannten Public-Key-Ring die öffentlichen Schlüssel anderer Teilnehmer gespeichert werden. [Sch05]

Bindung von Identitäten an öffentliche Schlüssel

Obwohl PGP konzeptionell ein kryptographisch starkes Produkt darstellt, gibt es eine Schwäche im System, die typisch für alle Public-Key-Management-Systeme ist. In der PGP-Dokumentation findet sich eine treffende Formulierung für das Problem:

> *"This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. It is the "Achilles heel" of public key cryptography, and a lot of software complexity is tied up in solving this one problem."* [NA99, S. 46]

Es geht also darum, eine zuverlässige Methode zu finden, mit der man feststellen kann, ob ein öffentlicher Schlüssel tatsächlich zu der angegebenen Identität gehört. Damit soll das Risiko minimiert werden, dass sich solche falschen oder kompromittierten Schlüssel im Public-Key-Ring einer Entität befinden. Angenommen, Bob möchte einen authentischen öffentlichen Schlüssel von Alice erhalten. Es existieren mehrere Lösungsansätze:

- Übergabe des Schlüssels ausserhalb der normalen Kommunikation. Alice könnte ihren Schlüssel auf einem Speichermedium sichern und das Medium per Post an Bob schicken. Diese Methode skaliert jedoch sehr schlecht bei vielen Kommunikationspartnern und ist zeitaufwendig. [Tyn05]
- Verifikation des Schlüssels über das Telefon. Wenn Bob die Stimme von Alice am Telefon kennt, könnte er sie anrufen und sie um ein Diktat des Schlüssels bitten. Eine praktischere Alternative wäre das Versenden des Schlüssels als E-mail oder das Hochladen auf einen Key-Server und die Verifikation des Hashwertes (Fingerprint) des Schlüssels über das Telefon. Auch diese Methode ist aufwendig. [Tyn05]
- Bob könnte Alice's Schlüssel auch von einer vertrauenswürdigen Quelle beziehen, bei der Alice ihren Schlüssel persönlich publiziert hat. Dies kann ein Teilnehmer sein, dem Bob vertraut oder aber eine vertrauenswürdige Zertifizierungsstelle. Diese vertrauenswürdige Quelle erstellt ein Zertifikat, welches die Identität von Alice an ihren Schlüssel bindet. Auch Zusatzinformationen wie z.b. ein Gültigkeitsdatum des Schlüssels können mit aufgenommen werden. Dieses Zertifikat wird von der vertrauenswürdigen Quelle digital signiert. [Tyn05]

Key Signing Party Um sich vorhandes direktes Vertrauen zwischen PGP-Teilnehmern zu Nutze zu machen, damit der Aufbau eines Web of Trust beschleunigt werden kann, gibt es sogenannten Key Signing Parties. Dabei treffen sich PGP-Teilnehmer, um sich gegenseitig die Zugehörigkeit ihrer PGP-Schlüssel zu ihrer Identität zu signieren. Alle Teilnehmer senden ihren öffentlichen Schlüssel vor der eigentlichen Key Signing Party an den Koordinator der Veranstaltung oder einen dafür vorgesehenen Key-Server. Um seine eigene Identität zu bestätigen, wird während der Veranstaltung meist die Vorlage amtlicher Dokumente (z.B. Personalausweis oder Reisepass) verlangt. [Tyn05]

4.5 Trust Management in mobilen Ad-Hoc-Netzen

Das Fehlen einer zentralen Autorität in sich selbstorganisierenden mobilen Ad-Hoc-Netzen stellt besondere Herausforderungen an das Trust Management in solchen Systemen. Zu

diesem Schluß kommen auch Savola und Uusitalo:

> *"Arguably, trust management is the most critical security issue in mobile ad hoc networks. If nodes do not have any prior knowledge of each other, the trust establishment becomes complicated. In these kinds of situations the nodes themselves should be responsible for their own security."* [SU06, S. 1]

Der Aufbau von Vertrauen zwischen den Knoten basiert dabei grundlegend auf den eingesetzten Authentifizierungs- und Sicherheitsmechanismen:

> *"Trust establishment in this kind of environment requires new solutions for authentication and security management in general. The responsibility for security management can be transferred naturally to individual nodes themselves."* [SU06, S. 1]

Unter Authentifizierung wird dabei der Prozess verstanden, der zur Feststellung der Identität einer Entität dient. Die Authentifizierung basiert dabei auf Credentials, die kryptographischer Natur sind (z.B. Zertifikate) unter Einbezug der individuellen Eigenschaften der Entitäten (z.B. Geräteadresse). Da der Einsatz zentraler Zertifizierungsstellen in mobilen Ad-Hoc-Netzen nicht in Frage kommt, gibt es in solchen Umgebungen zwei Ansätze zur Realisierung von Authentifizierung mittels Zertifikaten:

- Der Einsatz selbsignierter Zertifikate wie z.B. in *SelfOrgPKM*.
  Diese Ansatz ähnelt dem Konzept von PGP. Ein selbstsignierte Zertifikat ist dabei kein unmittelbarer Nachweis der Identität der signierenden Entität, es ist jedoch ein Nachweis über den Besitz eines bestimmten Public-Private-Schlüsselpaares. Da die Verantwortung über die eigene Sicherheit jeder Entität selbst zukommt, wird dieser Ansatz auch als *Peer Architecture* bezeichnet und funtioniert nach dem Prinzip des Web-of-Trust. [SU06]

- Eliminierung einer zentralen Zertifizierungsstelle und Verteilung ihrer Funktion (*Distributed Certification Authority*) auf mehrere Knoten, die diese Funktion nur gemeinsam ausführen können. Dieser Ansatz geh ört zum Forschungsbereich der Schwellwert-Kryptographie (*Threshold Cryptography*). [SU06]

Beide Ansätze sollen nachfolgend näher betrachtet werden, da es sich um aktuelle Beiträge handelt, deren Konzepte sich potentiell f ür eine Anwendung innerhalb mobiler Middleware eignen (vgl. Kapitel 5).

4.5.1 SelfOrgPKM

SelfOrgPKM steht für *Self-Organized Peer-to-Peer Key Management* und ist ein Key-Management-Schema für vollständig selbstorganisierte mobile Ad-Hoc-Netze (MANETs). Aus diesem Grunde wurde es einerseits mit der Anforderung einer geringen Komplexität zwecks Ressourcenschonung und andererseits dem Ziele der Eliminierung einer *Trusted Third Party* (TTP) entwickelt.

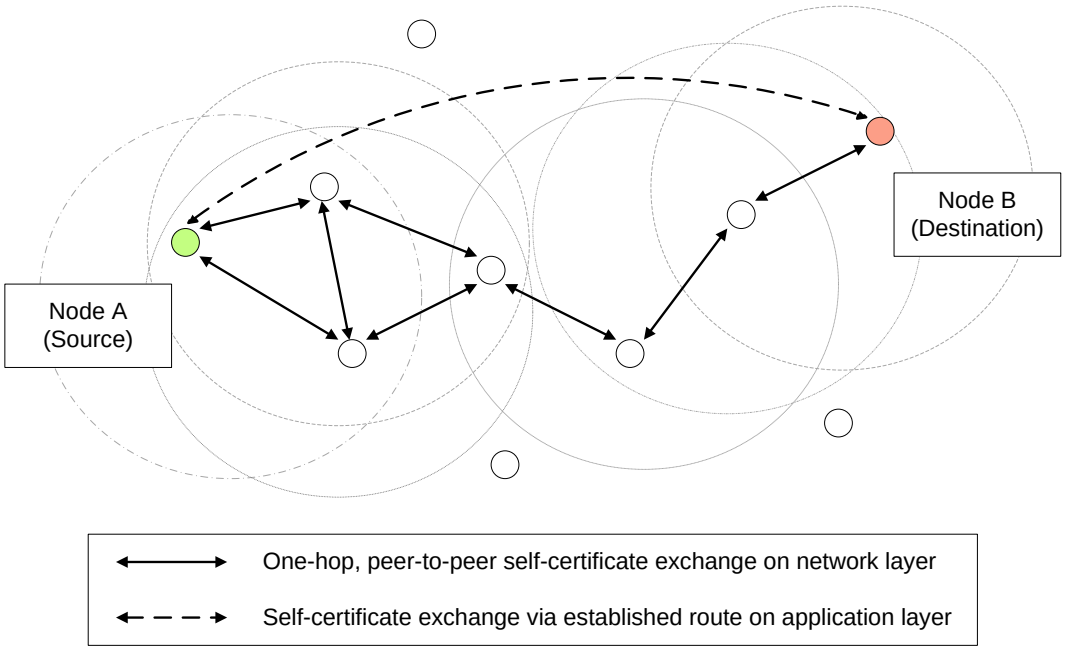

Abbildung 4.3: Zertifikatsaustausch in SelfOrgPKM (nach [MDM05, S. 24])

In SelfOrgPKM ist jede beteiligte Entität ihre eigene Autorität. Sie generiert ihr Schlüsselmaterial selbst, so dass auf diese Weise das Schlüsselmanagement gleichermaßen auf alle Knoten aufgeteilt wird. Der Aufbau von Vertrauensbeziehungen kann dabei auf zwei Ebenen geschehen: Erstens, auf Ebene der Vermittlungsschicht zwischen direkt benachbarten Knoten bei der Formierung des MANETs (z.B. während des Routings), und zweitens auf der Anwendungsebene auf einer Need-To-Know-Basis, also nur bei Bedarf. Dabei können Ketten von gegenseitigem Zertifikatsaustausch entstehen, die in Abbildung 4.3 schematisch dargestellt werden. Eine bidirektionale Vertrauensbeziehung zwischen zwei Knoten A und B entsteht dabei dadurch, dass die Knoten ihre Tupel $[K_A, ID_A]$ und $[K_B, ID_B]$ untereinander austauschen, wobei K_i das asymmetrische oder symmetrische Schlüsselmaterial und ID_i einen eindeutigen Netzwerkbezeichner oder eine eindeutige Netzwerkadresse eines Knoten i darstellt. Die Bindung zwischen dem Schlüssel K_i und dem Netzwerkbezeichner ID_i muss dabei öffentlich verifizierbar sein. Eine weitere Bindung des Tupels $[K_i, ID_i]$ an einen eindeutigen Benutzernamen wird dabei nicht von SelfOrgPKM übernommen, sondern benutzergesteuerten Anwendungen überlassen. [MDM05]

Key Management Das in SelfOrgPKM vorgestellte Key-Management-Schema benutzt eine Variante des asymmetrischen ElGamal-Algorithmus, abgeleitete öffentliche Schlüssel und krypto-basierte Netzwerkbezeichner. Jeder Knoten generiert sich entsprechend dem Algorithmus zunächst sein eigenes Schlüsselmaterial, welches aus einem privaten und einem öffentlichen Schlüssel besteht. Dieses Basis-Schlüsselpaar bleibt jedoch geheim und wird nicht zur Verschlüsselung der Kommunikation oder ähnlichem verwen-

det. Nur die Generierung weiterer (abgeleiteter) Private-Public-Schlüsselpaare aus dem Basis-Schlüsselpaar ist zulässig. Es gelten folgende Beziehungen zwischen dem Basis-Schlüsselpaar und den abgeleiteten Schlüsseln [MDM05]:

- Ein gültiger abgeleiteter öffentlicher Schlüssel kann nur generiert werden, wenn der Eigentümer den privaten Basis-Schlüssel kennt.
- Ein Benutzer kann sich abgeleitete öffentliche Schlüssel so oft generieren, wie er es für erforderlich hält.
- Der abgeleitete private Schlüssel muss statistisch unabhängig vom privaten Basis-Schlüssel und anderen bereits abgeleiteten privaten Schlüsseln sein. Ein kompromittierter abgeleiteter privater Schlüssel enthüllt somit keine Informationen über den privaten Basis-Schlüssel und zukünftige abgeleitete private Schlüssel.
- Es muss eine verifizierbare Bindung zwischen dem öffentlichen Basis-Schlüssel eines Benutzers und seinem jeweiligen abgeleiteten öffentlichen Schlüssel existieren, um Nicht-Bestreitbarkeit zu gewährleisten.

Netzwerkbezeichner Die Idee, krypto-basierte Bezeichner zu wählen, um öffentliche Schlüssel an Knoten bzw. an deren Netzwerk- oder Geräteadresse zu binden, wurde bereits in [MC04] vorgestellt.
Das ursprüngliche Konzept, einen partiellen Hashwert des öffentlichen Schlüssels eines mobilen Knotens zu benutzen, um einen Netzwerkbezeichner daraus zu formen, wird in [OR01] vorgestellt und wurde als Authentifizierungsprotokoll zur Vermeidung von gefälschten Adressen für das mobile IPv6 realisiert.
In SelfOrgPKM berechnet jeder Knoten einen Hashwert über seinen öffentlichen Schlüssel mit Hilfe einer kollisionsresistenten Einweg-Hashfunktion. Dieser Hashwert fungiert dann als Netzwerkbezeichner bzw. Geräteadresse dieses Knotens.

Selbstsignierte Zertifikate Ein selbstsigniertes Zertifikat $SelfCert_i$ sieht in SelfOrgPKM folgendermaßen aus:

$$SelfCert_i \quad = \quad [KI_i \,||\, y_i \,||\, \alpha_i \,||\, \beta_i]$$

mit

$$KI_i \quad = \quad [ID_i \,||\, y_i \,||\, r_i \,||\, SerNo \,||\, IssueDate \,||\, ValPeriod \,||\, ExtInfo]$$

In KI_i befinden sich zusätzliche Informationen zu einem bestimmten abgeleiteten Schlüsselpaar eines Knotens i, unter anderem z.B. die Gültigkeitsdauer des Schlüsselpaares, eine Seriennummer und der Erstellungszeitpunkt des zugehörigen Zertifikates $SelfCert_i$. Das in $SelfCert_i$ vorkommende Symbol y_i bezeichnet dabei den abgeleiteten öffentlichen Schlüssel des Knotens i, für den das Zertifikat ausgestellt wurde. Weiter ist das

Tupel (α_i, β_i) eine angehängte digitale Signatur über $[KI_i \mid\mid y_i \mid\mid \beta_i]$, damit Empfänger des Zertifikates die Zugehörigkeit des darin enthaltenen Schlüssels y_i zum Netzwerkbezeichner ID_i des Knotens i verifizieren können. [MDM05]

4.5.2 Distributed Certification Authority (Threshold Cryptography)

In dezentralisierten kabelgebundenen Rechnernetzen, in denen es keine zentrale Zertifizierungsstelle oder Trusted Third Party gibt, wird das Konzept der Threshold Cryptography, im deutschen Sprachraum manchmal auch als Schwellwert-Kryptographie bezeichnet, bereits eingesetzt, um Sicherheitsziele wie Verfügbarkeit, Vertraulichkeit und eine sichere Schlüsselverteilung zu erreichen. Die zugrundeliegende Idee ist, einen kryptographischen Schlüssel so auf n voneinander unabhängige Knoten aufzuteilen, dass mindestens t aus n Knoten mit $(t < n)$ notwendig sind, um das gemeinsame Geheimnis wiederherzustellen. Weniger als t Knoten können dabei mit ihrem Teil des Schlüssels alleine nichts bewirken. Ein Vorteil dieses Verfahrens ist beispielsweise, dass $(n-t)$ Knoten ausfallen dürfen oder kompromittiert sein können und das Schema dennoch funktioniert. Man kann nun einerseits ein (t, n)-Threshold-Verschlüsselungsschema definieren, indem ein öffentlicher Schlüssel auf die Knoten aufgeteilt wird, oder aber man definiert ein (t, n)-Threshold-Signaturschema, bei dem ein privater Schlüssel auf die Knoten aufgeteilt wird. [EC05]

Ertaul und Chavan [EC05] untersuchen auch die Verwendbarkeit einer RSA-Threshold-Cryptography-Implementierung in MANETs. Da der Beitrag aber sehr kryptographisch und formelreich gestaltet ist, soll hier auf die Darlegung der mathematischen Zusammenhänge verzichtet werden. Das Konzept der Schwellwert-Kryptographie wird in dieser Arbeit nicht weiter verfolgt, da es sich nicht so gut für offene mobile Systeme eignet, bei der die Kommunikation zwischen oft wechselnden Teilnehmern nur spontan bei Bedarf erfolgen soll und durch häufige Verbindungsabbrüche gekennzeichnet ist.

5 Konzept zur Erweiterung mobiler Middleware um Sicherheits- und Vertrauensaspekte

Dieses Kapitel dient zunächst der Definition von Anforderungen an eine Komponente für Sicherheit und Vertrauen, welche für den Einsatz innerhalb einer kontextbezogenen, mobilen und dezentralen Middleware entwickelt werden soll. Dabei steht vor allem das Konzept der (sicheren) Kooperation zwischen autonomen Systemteilnehmern mobiler Umgebungen im Vordergrund. Anhand dieser Anforderungen werden die vorhandenen konzeptionellen Vorschläge, die bislang aus dem DEMAC-Projekt hervorgegangen sind, untersucht. Aufbauend auf diesen Erkenntissen und den Ergebnissen der vorangegangenen Kapitel wird die Architektur der entworfenen Komponente, ihre Funktionalität sowie mögliche Interaktionsabläufe vorgestellt.

5.1 Anforderungsermittlung

Die im folgenden genannten Anforderungen erwachsen einerseits aus den Rahmenbedingungen des Mobile Computing (vgl. Kapitel 2) und andererseits aus den Untersuchungen der Sicherheit informationstechnischer Systeme (vgl. Kapitel 3) sowie von Trust-Management-Konzepten in offenen, dezentralisierten und mobilen Systemen (vgl. Kapitel 4). Die Anforderungen sind somit nicht von einem konkreten Projekt, wie z.B. der DEMAC-Zielplattform abhängig, wohl aber auf diese anwendbar. Da die DEMAC-Zielplattform über eine serviceorientierte Architektur verfügt (vgl. Abbildung 2.3), werden die Anforderungen für eine generische *Security-Service*-Komponente formuliert.

A1 Funktionalität:
Der Security Service soll die in Abschnitt 3.1.3 genannten Eigenschaften eines sicheren Systems, wie Vertraulichkeit, Integrität, Authentifizierung und Nicht-Bestreitbarkeit gewährleisten und schützen. Unter dem Oberbegriff des Trust Managements zusammengefasst, ist die Bereitstellung von Diensten zur Authentifizierung von Teilnehmern, zur Gewährleistung von Vertraulichkeit und Integrität ihrer Kommunikation und zum Aufbau von Vertrauensbeziehungen untereinander eine grundlegende Aufgabe des Security Service.

A2 Dezentralität:
Alle Prozesse und Operationen sowohl innerhalb des Security Service als auch logisch zwischen den Security Services verschiedener Knoten sollen dezentral ohne

die Notwendigkeit einer zentralen Instanz ablaufen. Durch die Selbstverantwortlichkeit jedes Knotens für seine eigene Sicherheit wird eine hohe Toleranz bezüglich des Ausfalls von Knoten erreicht.

A3 Leichtgewichtigkeit:
Um den Ansprüchen mobiler Umgebungen und Geräte in Bezug auf eine allgemeine Ressourcenknappheit, kleine Displays, häufige Verbindungsabbrüche, eine geringe Datenrate usw. gerecht zu werden, sollte der Security Service, sein Speicherbedarf und alle seine Operationen und Berechnungen leichtgewichtig und kompakt sein. Der Security Service soll die Grundfunktionalität des Systems absichern, darf diese jedoch nicht beeinträchtigen. Genau an dieser ist der Benutzer schließlich vorrangig interessiert. Die Sicherheit an sich ist dementsprechend als nichtfunktionaler Aspekt zu betrachten.

A4 Autonomie und generische Verwendbarkeit:
Um generisch zu sein, und so für viele Middleware-Plattformen einsetzbar zu sein, sollte der Security Service möglichst autonom und unabhängig von anderen Middleware-Komponenten funktionieren und von diesen nur benutzt werden. Aus diesem Grunde eignet sich besonders eine Anordnung oberhalb des logischen Adressierungs- und Transportmechanimus der Middleware und unterhalb komplexerer Anwendungskomponenten. Diese Anordnung ermöglicht es dem Security Service zusätzlich, von den zugrunde liegenden tatsächlichen Übertragungstechnologien wie Bluetooth, WLAN etc. zu abstrahieren.

A5 Flexibilität und Erweiterbarkeit:
Die Architektur eines Security Service sollte so gewählt werden, dass seine inneren Module gekapselt sind, um sie unabhängig voneinander austauschen zu können. Eine lose Kopplung der inneren Module und eine abstrakte Schnittstellendefinition ermöglichen eine bessere Austauschbarkeit von Vertrauensmodellen und Algorithmen sowie eine einfachere Einführung neuer Module.

A6 Unaufdringlichkeit und Benutzbarkeit:
Alle erforderlichen Operationen des Security Service sollen möglichst im Hintergrund ablaufen und die Aufmerksamkeit des Benutzers so wenig wie möglich beanspruchen. Dies ist einerseits in Mark Weisers Sinne vom *Verschwinden des Computers* [Wei91], und hat andererseits auch den Vorteil, Fehlkonfigurationen durch den Benutzer vermeiden zu können. Sollten Interaktionen mit dem Benutzer notwendig sein, müssen diese leichtverständlich, dem Display des Gerätes angepasst und dem Benutzer zumutbar sein. Beispielsweise wäre eine manuelle Verifikation von 1024-Bit-langen Schlüsseln nicht zumutbar. Nur benutzbare Systeme und Komponenten finden langfristig Verwendung und Anklang bei Benutzern.

5.2 Sicherheit in DEMAC - Analyse bisheriger Vorschläge

Die abstrakte Architektur der DEMAC-Middleware wurde bereits in Abschnitt 2.4 eingeführt (vgl. Abbildung 2.3). Da die Sicherheit eines Systems nicht losgelöst von dessen Architektur betrachtet werden kann, soll zunächst ein Überblick darüber verschafft werden, auf welchen Schichten der DEMAC-Middleware bereits Vorschläge zur Integration von Sicherheitsmechanismen geäußert wurden. Zusätzlich soll analysiert werden, welchen Beitrag die Realisierung dieser Vorschläge zur ganzheitlichen Sicherheit des Systems leisten würde. Darauf aufbauend soll dann ein eigenes Sicherheitsmodell für DEMAC entwickelt werden.

In bisherigen Arbeiten im Rahmen des DEMAC-Forschungsprojektes wurde einerseits vorgeschlagen, eine Sicherheitskomponente auf Ebene des DEMAC Process Service und andererseits auf Ebene des DEMAC Context Service zu realisieren. Im Folgenden sollen beide Ansätze kritisch diskutiert werden.

5.2.1 Sicherheit auf Ebene des Process Service

Da die Prozesse, die an die Ausführungsumgebung des DEMAC Process Service übergeben werden, komplexe Aufgaben samt schutzbedürftigen Informationen enthalten können, müssen diese Prozesse im Allgemeinen vertraulich behandelt und vor Manipulation bewahrt werden [Zap05]. Dabei ist in erster Linie die Vertraulichkeit des Prozesses auf dem Kommunikationswege zu betrachten. Auch die physische Vertraulichkeit der Prozessdaten durch eine verschlüsselte Speicherung auf dem Gerät ist denkbar, allerdings birgt die drahtlose Übertragung von Prozessen grundsätzlich ein höheres Risiko der Offenlegung von Informationen, da der Angreifer sich nur in Reichweite des anzugreifenden Gerätes befinden muss, anstatt es dem Besitzer zu entwenden. Bei besonders schutzbedürftigen Prozessen geschäftlicher Art dürfen zusätzlich nur bestimmte Personen involviert werden, so dass eine zuverlässige Authentifizierung der Prozessteilnehmer notwendig ist [Zap05].

Der Grad der Schutzbedürftigkeit eines Prozesses könnte generell von speziellen Anwendungsszenarien abhängig gemacht werden, allerdings ist es einem Prozessinitiator möglich, seine ursprünglichen Intentionen zusätzlich durch Definition nicht-funktionaler Kriterien zu wahren, die während der gesamten Zeit der Prozessausführung auch auf fremden Geräten berücksichtigt werden [Zap05]. Zaplata schlägt daher vor, die Absicherung der Prozessbeschreibungen auf Ebene des DEMAC Process Service zu realisieren und eine Sicherheitskomponente zu entwickeln, die optional als Zusatzkomponente (*Extension Module*) an die Basiskomponente (*Base Module*) angeschlossen werden kann. Diese soll sich um die Verschlüsselung der Kommunikation und um die Authentifizierung von Geräten und Prozessteilnehmern kümmern. Dem Prozessinitiator steht es dabei offen,

Strategien (nicht-funktionale Aspekte) bezüglich Sicherheit für den gesamten Prozess global festzulegen oder aber für einzelne Aktivitäten samt vertraulichen Informationen zu definieren. [Zap05]

Da diese Strategien (*DPDL Strategies*) innerhalb der Kernkomponente des Process Service ausgewertet werden, ist es nachvollziehbar, eine Sicherheitskomponente innerhalb der DEMAC Ausführungsumgebung integrieren zu wollen, sofern die Strategien die Sicherheit des Prozesses in Bezug auf Vertraulichkeit und Integrität der Prozesse sowie in Bezug auf die Authentifizierung der Prozessteilnehmer betreffen.

Da es nicht Ziel in Zaplatas Arbeit [Zap05] war, eine oder mehrere Sicherheitskomponenten für die DEMAC-Middleware zu modellieren und zu realisieren, die Notwendigkeit hierfür aber festgestellt wurde, ist der Vorschlag der Realisierung einer Sicherheitskomponente als optionale Zusatzkomponente der Basiskomponente konzeptioneller Natur und wurde bislang nicht umgesetzt. Da es aber auch Geräte geben kann, die nur über die Kernkomponente der Ausführungsumgebung verfügen, wären diese von vornherein von der Teilnahme an einer abgesicherten Prozessausführung mit Hilfe von Trust Management, Authentifizierung und Verschlüsselung ausgeschlossen, falls diese Mechanismen innerhalb einer optionalen Zusatzkomponente realisiert sind. Gleiches gilt für Geräte mit Kern- und Basiskomponente ohne weitere optionale Zusatzkomponenten. Die von Zaplata [Zap05] geäußerte Idee, solche Geräte trotzdem nur zur Weiterleitung eines Prozesses an einen geeigneten Prozessteilnehmer einzubeziehen, indem die nichtfunktionalen Aspekte des Prozesses sowie weitere, zur Prozessverwaltung benötigten Daten lesbar bleiben und der Rest der Prozessbeschreibung bereits verschlüsselt oder anonymisiert ist, erscheint in der vorliegenden Arbeit nicht anwendbar. Dies gründet sich einerseits in der Annahme, dass ein gemeinsames Grundwissen zwischen der verschlüsselnden Partei und der entschlüsselnden Partei etabliert sein muss, damit ein Klartext und ein Geheimtext einander korrekt zugeordnet werden können und dieses Wissen auf einer Peer-to-Peer-Basis bei Bedarf etabliert werden soll. Andererseits gründet es auch in der Tatsache, dass es dem Gerät ohne Sicherheitskomponente, welches einen schutzbedürftigen Prozess weiterleiten soll, nicht möglich wäre, den potentiellen Empfänger der Prozessbeschreibung zu authentifizieren, da die Mechanismen der Authentifizierung ja gerade im Funktionsumfang der fehlenden Zusatzkomponente für Sicherheit implementiert sind.

Als weiteren notwendigen Sicherheitsmechanismus nennt Zaplata [Zap05] den Schutz eines Ausführers fremder Prozesse vor dessen potentiell maliziösen Auswirkungen. Es sollte also möglich sein, vor Ausführung eines fremden Prozesses diesen auf seine Schadhaftigkeit und sonstige Auswirkungen auf die Ressourcen des eigenen Gerätes zu prüfen. Die Notwendigkeit dieses Aspektes ist unumstritten, allerdings fokussiert sich die vor-

liegende Arbeit weniger auf die Prüfung einzelner Aktivitäten eines DPDL-Prozesses, sondern vielmehr auf die Beziehungen zwischen den jeweiligen Kooperationspartnern. Wenn es möglich wäre, ein hohes Maß an Vertrauen zwischen dem Prozessinitiator und dem tatsächlichen Ausführer des Prozesses zu etablieren, so könnte eine aufwendige Überprüfung potentiell negativer Auswirkungen der Ausführung entfallen.

Ein weiterer Kritikpunkt grundsätzlicher Art, der unmittelbar nichts mit der Integration der Sicherheitskomponente auf einer bestimmten Ebene zu tun hat, ist die Tatsache, dass die Möglichkeit zur Definition sicherheitsrelevanter nicht-funktionaler Kriterien durch die Prozessinitiatoren Freiraum für Fehler, Nachlässigkeiten und somit auch für Sicherheitslücken schafft. So hat die Erfahrung aus der Praxis gezeigt, dass optionale oder standardardmäßig deaktivierte Parameter und Funktionen von Benutzern häufig ignoriert oder falsch verwendet werden. Dies kann unter anderem aus Mangel an Verständnis oder aus Bequemlichkeit geschehen und betrifft sowohl private als auch geschäftliche Anwender. Belegt wird dieser Umstand durch zahlreiche Studien: Laut einer Studie nutzen über 50% aller Unternehmen mit WLAN-Zugang nicht einmal die schwache WEP-Verschlüsselung[1], die aber immer noch besser als gar keine Verschlüsselung ist, da sie zumindest die Personen fernhält, die zufällig auf das Unternehmensnetz stossen [Eck06].
Desweiteren widerspricht es Mark Weiser's Vorstellung des Ubiquitous Computing vom Verschwinden des Computers aus dem Bewußtsein des Menschen [Wei91], wenn der Benutzer mit zu vielen technischen Details und diesbezüglichen Entscheidungen konfrontiert wird (vgl. Anforderung A6). Prinzipiell ist es daher besser, wenn vorhandene Sicherheitsmechanismen erst einmal standardardmäßig aktiviert sind und so automatisiert wie möglich ohne Benutzerinteraktionen ablaufen können. Wenn dies funktioniert, dann kann man sich immer noch überlegen, ob man eine Möglichkeit zur manuellen Modifikation oder gar Abschaltung der Mechanismen durch den Benutzer zuläßt, um damit in Ausnahmefällen Abhilfe durch Ressourcenfreigabe zu leisten. Dies sollte konzeptionell gesehen aber nicht die Regel sein.

5.2.2 Sicherheit auf Ebene des Context Service

In der ebenfalls im Rahmen des DEMAC-Forschungsprojekts entstandenen Arbeit von Turjalei [Tur06] geht es um die Integration von Context-Awareness in eine Middleware für mobile Systeme. Der Autor kommt dabei zu dem Schluss, dass Kontextinformationen vertrauliche Informationen enthalten können wie z.B. Aufenthaltsorte konkreter Personen, Identitäten von Personen in der Umgebung eines Benutzers, Datumsangaben, Uhrzeiten etc. Zum Schutz der Privatsphäre des einzelnen Benutzers soll der nichtautorisierte Zugriff auf diese Informationen und ihre unkontrollierte Verteilung vermie-

[1] Mit mathematischen Methoden wurde die Zeit zum Hacken eines per WEP geschützten WLAN auf wenige Minuten reduziert.

den werden. Hierzu schlägt Turjalei [Tur06] vor, einen *SecurityPrivacyService* innerhalb der Kontext-Management-Komponente auf Ebene des Context Service zu realisieren, die sich um eine Ver- und Entschlüsselung der vertraulichen Kontextinformationen kümmert. Als weitere Aufgabe des SecurityPrivacyService neben der Verschlüsselung von Kontextinformationen wird noch die Erfordernis eines Verfahrens genannt, mit dessen Hilfe entschieden werden kann, welche der verfügbaren Kontextdaten auf einem Gerät an andere Teilnehmer preisgegeben werden dürfen und wie diese Teilnehmer bestimmt werden sollen. Genauer werden beide Punkte jedoch nicht erläutert, da die betreffende Arbeit sich hauptsächlich mit der Modellierung und Präsentation von Kontext und Context-Awareness beschäftigt und nicht mit der Entwicklung eines Sicherheitskonzepts.

Zunächst einmal muss geprüft werden, an welcher Stelle eine Verschlüsselung von Kontextinformationen in Frage kommen könnte. Es könnte sowohl die Verschlüsselung der Kontextdaten auf dem Übertragungswege in Betracht kommen, als auch die verschlüsselte Speicherung der Kontextdaten auf dem mobilen Gerät eines Benutzers, wie dies von Zaplata [Zap05] bereits analog für die Prozessbeschreibungen auf Ebene des Process Service vorgeschlagen wurde. Der Hauptnachteil bei einer verschlüsselten Speicherung von Daten auf einem mobilen Gerät ist die Tatsache, dass der Prozess der Ver- und Entschlüsselung bei jedem Zugriff auf diese Daten eine Benutzerinteraktion wie beispielsweise die Eingabe einer PIN, eines Passwortes oder einer Stimmprobe erfordert. Eine automatische Ver- und Entschlüsselung ohne Benutzerinteraktion wäre jedoch nur möglich, wenn der Schlüssel auf dem Gerät selbst gespeichert ist. Würde ein Angreifer ein solches Gerät stehlen, hätte er somit auch Zugriff auf den Schlüssel und das ganze Verfahren hätte keinen Gewinn eingebracht. In diesem Falle kann man die Verschlüsselung auch unterlassen und so wertvolle Rechenressourcen und Energie einsparen.

Der Schutz vertraulicher Kontextinformationen soll in der hier vorliegenden Arbeit in erster Linie dadurch sichergestellt werden, dass ihre Weitergabe nur an authentifizierte und vertrauenswürdige Entitäten unter Benutzung einer gesicherten Kommunikationsverbindung weitergegeben wird. Da jeder Benutzer autonom ist, kann er auf diese Weise nach eigenem Ermessen entscheiden, wem er seine Informationen preisgeben möchte und wem nicht.

5.3 Entwurf der Middleware-Komponente: der Security Service

Da die DEMAC-Middleware von konkreten Übertragungsprotokollen abstrahiert, kann man sich nicht auf eventuell vorhandene Sicherheitsmechanismen der unteren Transportschichten verlassen. Darüber hinaus hat die Analyse bisheriger Vorschläge für die Sicherheit der DEMAC-Middleware in Abschnitt 5.2 ergeben, dass auf unterschiedlichen Ebenen, wie z.B. auf Ebene des Process Service oder aber auf Ebene des Context Service,

ähnliche Aufgaben bezüglich der Authentifizierung von Entitäten und der Vertraulichkeit von Daten und Informationen anfallen. Es ist daher sinnvoll, eine eigene zentrale Komponente für Sicherheit und Vertrauen innerhalb der Middleware zu entwerfen, die Dienste wie Authentifizierung, Verschlüsselung und Trust-Management für alle internen Komponenten anbietet. Um der A4-Anforderung nach Autonomie und generischer Verwendbarkeit sowie der serviceorientierten Architektur der DEMAC-Middleware gerecht zu werden, stellt diese Komponente nach außen hin eine Sevice-Einheit, den Security Service, dar, um ihre innere Komplexität zu verbergen. Damit die Komponente generell von vielen anderen Middleware-Komponenten bei Bedarf benutzt werden kann, soll sie so tief wie möglich in der Schichtenarchitektur der entsprechenden Middleware angeordnet werden. Innerhalb der DEMAC-Middleware bietet sich hier besonders die Ebene des DEMAC Asynchronous Transport Service an (vgl. Abbildung 2.3). Der Transport Service verwendet ein eigenes logisches Adressierungsschema, welches sich der Security Service ebenfalls zu Nutze machen kann. Eine Nachricht wird nämlich erst nach Verlassen des Transport Service an der Schnittstelle nach unten (in Richtung des physischen Übertragungskanals) in übliche Paketformate wie z.B. die der Protokolle Bluetooth oder WLAN transformiert und die logischen Adressen von Sender und Empfänger in physische Netzwerkadressen übersetzt [KZL07].

5.3.1 Architektur des Security Service

Da der Security Service für einen dezentralisierten Einsatz ausgelegt ist (Anforderung A2), muss jeder DEMAC-Teilnehmer diesen selbst implementieren. Der Security Service ist dabei in drei Schichten organisiert (Abbildung 5.1).

Trust Layer

Auf dem *Trust Layer* befinden sich alle Komponenten, welche die zum Aufbau von Vertrauensbeziehungen notwendigen Daten, wie Schlüssel, Zertifikate und Reputationen, anfordern, erstellen, prüfen und verwalten. Dies umfasst den *Key Manager*, den *Credential Manager* und den *Reputation Manager*.

Key Manager Der Key Manager ist für die selbstorganisierte Generierung des eigenen asymmetrischen Schlüsselmaterials zuständig. Er erstellt ebenfalls die dazugehörigen selbstsignierten Zertifikate und die zur einmaligen symmetrischen Ver- und Entschlüsselung von Nachrichten benötigten symmetrischen Schlüssel.
In einem selbstsignierten Zertifikat ist unter anderem der öffentliche Schlüssel und der Netzwerkbezeichner des Besitzers enthalten. Das selbstsignierte Zertifikat weist die Authentizität eines öffentlichen Schlüssels nach, da eine Signatur mit dem entsprechenden privaten Schlüssel über den Hashwert des gesamten Zertifikates erstellt wird und bei erfolgreicher Verifikation den Nachweis des Ausstellers über den Besitz des privaten

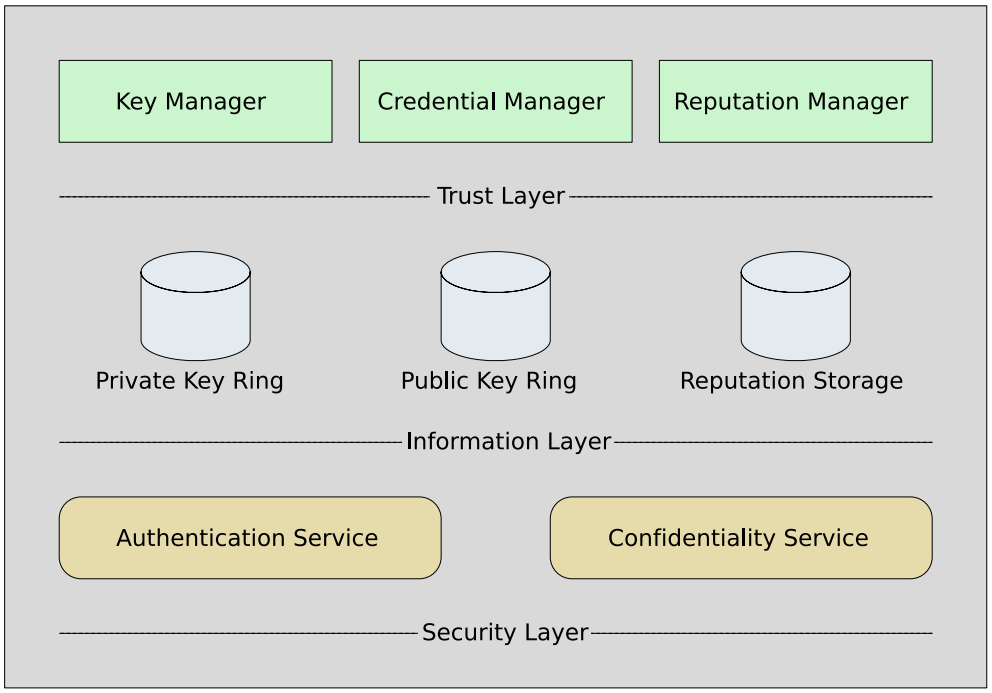

Abbildung 5.1: Architektur des Security Service

Schlüssels darstellt. Eine tatsächliche Bindung des im Zertifikat angegebenen Netzwerkbezeichners an den öffentlichen Schlüssel ist durch das selbstsignierte Zertifikat allein noch nicht gegeben. Da ein fehlender öffentlicher Schlüssel jedoch direkt vom betroffenen Kooperationspartner angefordert wird, dessen Netzwerkbezeichner dementsprechend bereits bekannt ist, bietet ein Vergleich des im Zertifikat angegebenen Netzwerkbezeichners mit dem Sender des Zertifikates eine ausreichende Sicherheit. Die Bindung eines konkreten real vorhandenen Personennamens an einen Netzwerkbezeichner ist damit noch nicht gegeben.
Eine sogenannte *Out-of-Band*-Verteilung von Schlüsselmaterial ausserhalb von Kommunikationsprotokollen und eigenständiger Generierung ist in einer dezentralen und dynamischen Anwendungsumgebung mit Ad-Hoc-Charakter nicht sinnvoll, da einerseits kein Administrator verfügbar ist und diese Methode andererseits auch schlecht skaliert. Sollte der Fall eintreten, dass die Middleware beispielsweise innerhalb eines kleinen Unternehmensnetzes mit weniger als 15 Geräten verwendet wird, so wäre eine manuelle Eintragung von Schlüsseln in jedes Gerät denkbar. Da aber hierfür ebenfalls ein Administrator benötigt wird, könnte man stattdessen auch gleich eine klassische Public-Key-Infrastruktur aufbauen. Ausserdem wäre ein solches mobiles Netz dann ziemlich statisch, da unternehmensfremde Geräte nicht als Teilnehmer fungieren dürfen.

Credential Manager Der Credential Manager ist für die Anforderung fehlender öffentlicher Schlüssel und Zertifikate verantwortlich. Das hierfür benötigte Kommunikationsprotokoll wird in Abschnitt 5.3.2 eingeführt. Dieses Protokoll wird nur gestartet,

wenn das Nicht-Vorhandensein benötigter Credentials festgestellt wurde. Hierzu sucht der Credential Manager unter Angabe des betroffenen Netzwerkbezeichners nach Einträgen im Public Key Ring. Der Credential Manager prüft die erhaltenen Zertifikate auf ihre Validität (Verifikation). Der Credential Manager sorgt auch für das Zurückziehen des eigenen Schlüsselmaterials unter anderen Teilnehmern, falls dieses z.B. kompromittiert worden ist.

Reputation Manager Der Reputation Manager verwaltet eigens erhaltene Reputationen, fordert bei Bedarf die Reputationen anderer Teilnehmer an und evaluiert diese. Diese Evaluation soll Rückschlüsse aus vergangenen Inter- oder Transaktionen im Hinblick auf das zukünftige Verhalten eines potentiellen Kooperationspartners ermöglichen. In der vorliegenden Arbeit erscheint es am sinnvollsten, wenn jeder Teilnehmer seine eigens erhaltenen Bewertungen selbst speichert und diese bei Bedarf seinen potentiellen Kooperationspartnern präsentiert, um stets möglichst autonom handeln zu können und nicht von der Verfügbarkeit der Bewerter abhängig zu sein. Es bietet sich folglich an, ein Reputationssystem zu verwenden, in dem ausschließlich positive Bewertungen gespeichert werden, da negative Bewertungen andernfalls einfach verschwiegen oder gelöscht werden könnten (vgl. Abschnitt 4.3.2). Wie eine vom Reputation Manager generierte Reputation aussehen könnte, ist in Abbildung 5.2 dargestellt. Die Reputation enthält Angaben über ihren Aussteller sowie den Empfänger, der durch die Reputation bewertet wird. Die eigentliche Bewertung besteht dabei aus dem Ergebnis der Interaktion und daraus resultierenden Empfehlungen und wird zusammen mit dem Zeitpunkt und der Art der bewerteten Interaktion beschrieben. Zum Schutz vor Manipulation oder Fälschungen ist eine Reputation mit der digitalen Signatur des Ausstellers versehen. Eine Reputation enthält daher auch stets die Key ID des öffentlichen Schlüssels des Ausstellers, damit der benötigte Schlüssel zur Verifikation der Signatur eindeutig zugeordnet werden kann. Unter Umständen müssen fehlende öffentliche Schlüssel und Zertifikate der Aussteller für die Verifikation der digitalen Signaturen angefordert werden, welches weiteren Kommunikations- und Berechnungsaufwand verursacht. Wenn es jedoch beispielsweise aus Gründen der Nicht-Verfügbarkeit von Ausstellern nicht möglich sein sollte, die von potentiellen Kooperationspartnern präsentierten Reputationen zu verifizieren, so können diese nicht verifizierbaren Reputationen nur mit einem geringeren Vertrauenswert als verifizierte Reputationen in die Evaluation mit einfließen.

Information Layer

Den soeben vorgestellten Komponenten des Trust Layer stehen geeignete Behälter zur Speicherung der entsprechenden Daten zur Verfügung. Diese sind auf der Ebene des *Information Layer* angesiedelt (vgl. Abbildung 5.1). Da die Schlüssel und Zertifikate grundlegend für den Betrieb der Authentifizierungs- und Vertraulichkeitsdienste sind, müssen sie systematisch gespeichert und organisiert sein, um einen schnellen Zugriff zu ermög-

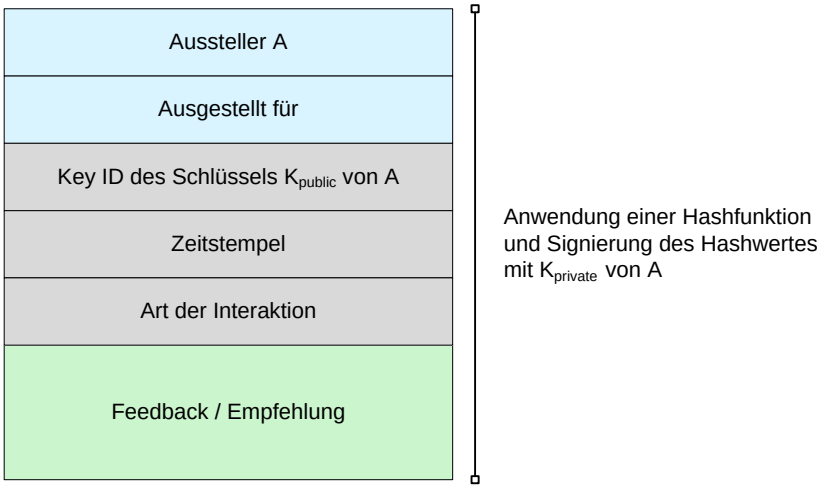

Abbildung 5.2: Format einer Reputation

lichen.

Private Key Ring Im *Private Key Ring* verwaltet jeder Knoten seine eigenen Schlüssel. Hierzu gehört das durch den Key Manager generierte asymmetrische Schlüsselpaar (unter Umständen auch mehrere), selbstsignierte Zertifikate sowie alle weiteren erforderlichen symmetrischen Schlüssel, die kurzzeitig gespeichert werden müssen. Sie werden in Datenstrukturen gespeichert, in denen sie um weitere Parameter wie beispielsweise Zeitstempel oder *Key Identifier* (vgl. Abschnitt 4.4.2) erweitert werden können.

Public Key Ring Im Public Key Ring werden die öffentlichen Schlüssel und Zertifikate anderer Knoten gespeichert, die an vergangenen Interaktionen beteiligt gewesen sind oder für zukünftige Interaktionen potentielle Kooperationspartner darstellen. Auch sie werden um weitere Angaben wie beispielsweise ihre Gültigkeitsdauer oder einen Vertrauenswert in die Authentizität des Schlüssels ergänzt. Da mobile Geräte eine begrenzte Speicherkapazität aufweisen, müssen nicht mehr benötigte Credentials wieder regelmäßig aus dem Public Key Ring entfernt werden.

Reputation Storage Im Reputation Storage werden sowohl eigene erhaltene Reputationen als auch die angeforderten Reputationen von Interaktionspartnern gespeichert. Da alle Reputationen das gleiche Format haben, wird der Reputation Storage als ein einzelner Behälter modelliert. Da ein Teilnehmer über mehrere Geräte und somit auch über verschiedene Netzwerkbezeichner verfügen kann, wird der Reputationssatz eines bestimmten Teilnehmers über eine eigene ID indexiert, die alle verfügbaren Reputationen dieses Teilnehmers zusammenfasst, selbst wenn sie von verschiedenen Geräten stammen. Eine langfristige Speicherung ist nur für die Reputationen vorgesehen, die für den jeweiligen Besitzer des implementierenden Gerätes selbst ausgestellt sind. Reputationen,

die der Bewertung der Zuverlässigkeit potentieller Kooperationspartner dienen, können nach einer Prüfung wieder entfernt werden.

Security Layer

Die dritte Schicht ist der *Security Layer* (vgl. Abbildung 5.1), der die tatsächlichen Funktionen des Ver- und Entschlüsselns, des Signierens und der Verifikation von Signaturen ausübt. Diese Schicht gliedert sich in den *Authentication Service* und den *Confidentiality Service*, die jeweils eigenständige Funktionseinheiten darstellen, in der Regel aber in einem hybriden Modell kombiniert verwendet werden.

Authentication Service Dieser Service überprüft Signaturen eingehender Nachrichten und erstellt Signaturen für ausgehende Nachrichten. Dafür müssen benötigte Schlüssel im jeweiligen Key Ring nachgesehen werden. Dieser Dienst soll die Authentizität von Nachrichten bzw. der Nachrichtenquelle nachweisen oder widerlegen. Die Zuverlässigkeit des Authentication Service basiert somit grundlegend auf der Authentizität der verwendeten asymmetrischen Schlüssel.

Confidentiality Service Der Confidentiality Service ist für die Gewährleistung der Vertraulichkeit von Nachrichten zuständig. Diese werden auf der Senderseite symmetrisch verschlüsselt und auf der Empfängerseite wieder entschlüsselt. Hinreichende Vertraulichkeit ist dann gegeben, wenn neben der sicheren Übermittlung von Nachrichten auch die Identität der Kommunikationspartner nachgewiesen ist.

Der Einsatz einer hybriden Kombination aus symmetrischer und asymmetrischer kryptographischer Verfahren und das spezielle Nachrichtenformat des Security Service, welches im nächsten Abschnitt eingeführt wird, erfordert die Anwendung beider Dienste auf jede zu schützende Nachricht, um Angriffsmöglichkeiten so gering wie möglich zu halten.

5.3.2 Kommunikation zwischen Security Services

Eine Kommunikation zwischen den jeweiligen Security Services zweier Knoten ist immer dann erforderlich, wenn fehlende *Credentials* zur Inanspruchnahme der Sicherheitsdienste erst noch angefordert werden müssen (vgl. Abbildung 5.3), oder aber wenn die Authentifizierungs- und Verschlüsselungsdienste des Security Layers bereits einseitig in Anspruch genommen wurden, um den Sicherheitsanforderungen einer Transaktion oder eines Prozesses gerecht zu werden. Um von dem Nachrichtenformat und den Nachrichten-Transportmechanismen der entsprechenden Middleware zu abstrahieren und damit die A4-Anforderung nach Autonomie und generischer Verwendbarkeit zu erfüllen, stellt der Security Service ein eigenes Nachrichtenformat zur Verfügung, um die Übertragung

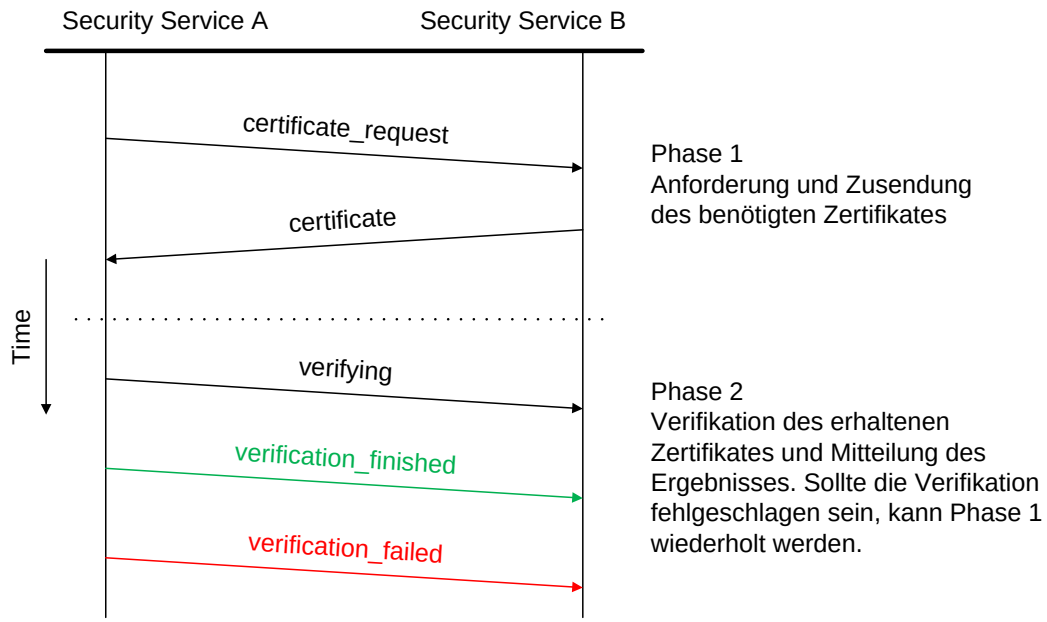

Abbildung 5.3: Protokoll zum Anfordern benötigter Zertifikate

verschlüsselter Nachrichten und den Austausch der verwendeten Schlüssel zu ermöglichen. Der bestehende Nachrichten-Transportmechanismus der Middleware, der auch sogenannte Management-Nachrichten verschickt, die keiner Sicherung bedürfen oder Teil von Initialisierungsprotokollen sind (z.B. beim Routing), kann somit vor Veränderungen bewahrt bleiben. Die von einem Security Service generierten Nachrichten können daher als *Payload* (Nutzdaten) im tatsächlichen Nachrichtenformat des Transportmechanismus getunnelt werden.

In Abbildung 5.4 ist das Nachrichtenformat des Security Service dargestellt. Um das Nachrichtenformat generisch verwendbar zu gestalten (Anforderung A4) und hierfür von konkret implementierten Algorithmen und Schlüssellängen abstrahieren zu können, benennt der Header die speziell für diese Nachricht eingesetzten kryptographischen Verfahren selbst. Dies ermöglicht eine flexible Erweiterung des Security Service um neue Verfahren (Anforderung A5), deren Parameter auch wieder übermittelt werden können. Desweiteren enthält der Header neben Empfänger und Sender der Nachricht auch die Key Identifier der verwendeten öffentlichen Schlüssel der Kommunikationspartner. Ein Zeitstempel garantiert die *Frische* der Nachricht. Dies bedeutet, dass die Nachricht zeitnah erstellt worden ist und keine ältere, wieder eingespielte, Nachricht darstellt (vgl. Abschnitt 3.1.5, Replay-Angriff). Damit dieser Zeitstempel nicht manipuliert werden kann, wird er mit dem privaten Schlüssel des Senders signiert. Die schutzbedürftigen Nutzdaten der Nachricht werden symmetrisch verschlüsselt. Der dabei eingesetzte symmetrische Schlüssel wird eigens für diese Nachricht generiert und für keine weiteren Nachrichten verwendet. Dies schützt vor Kompromittierung des symmetrischen Schlüssels

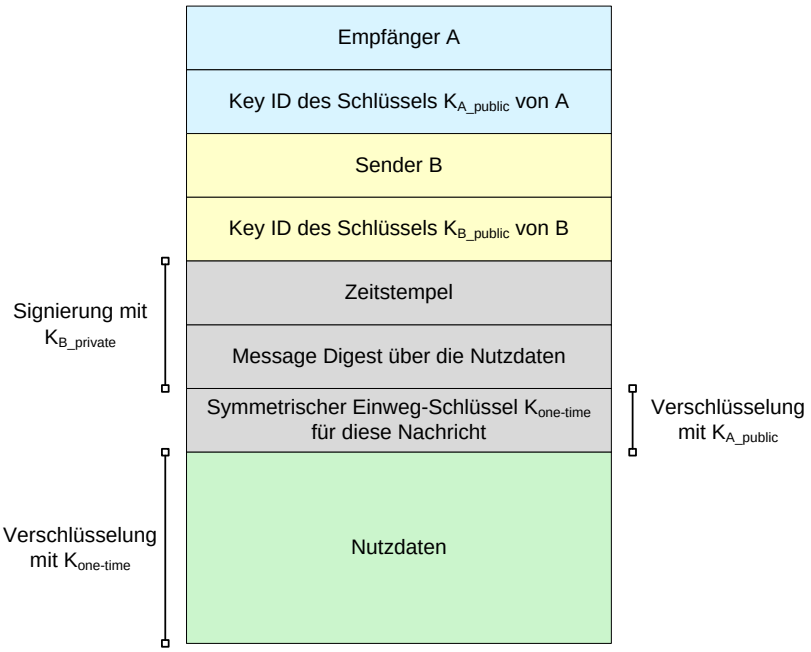

Abbildung 5.4: Das Nachrichtenformat des Security Service

und unterstützt eine asynchrone, nachrichtenbasierte Kommunikation mit wechselnder Verfügbarkeit der Kommunikationspartner. Damit der symmetrische Schlüssel nicht im Klartext übertragen wird, was das gesamte Verfahren nutzlos machen würde, wird er asymmetrisch mit dem öffentlichen Schlüssel des Empfängers verschlüsselt. Dies gewährleistet, dass nur der Empfänger den symmetrischen Schlüssel wieder rekonstruieren kann, um die Nutzdaten zu entschlüsseln, da nur er im Besitz des passenden privaten Schlüssels ist. Um die Integrität der Nutzdaten zu sichern, wird ein Message Digest darüber berechnet. Um eine eventuelle Manipulation der Nachricht erkennbar zu machen, wird der Message Digest mit dem privaten Schlüssel des Senders signiert. Dies weist bei einer verifizierbaren Signatur gleichzeitig die Authentizität der Nachricht dar, da aufgrund der zu verwendenden kollisionsresistenten Hashfunktion nur der Sender über den privten Schlüssel verfügt, mit dem der korrekte Hashwert signiert worden ist. Eine solche Nachricht des Security Service kann generell in eine beliebige Middleware-spezifische Transportnachricht eingebettet werden.

6 Implementierung

Der in Kapitel 5 konzeptionell entworfene Security Service wurde im Rahmen der vorliegenden Arbeit prototypisch für die mobile Middleware-Plattform DEMAC (vgl. Abschnitt 2.4) implementiert. In diesem Kapitel wird das Ergebnis dieser Implementierung sowie die verwendeten Mittel und Methoden vorgestellt.

6.1 Verwendete Plattform und Werkzeuge

Zum besseren Verständnis der Vorgehensweise bei der Implementierung soll die verwendete Java-Plattform sowie nützliche Hilfsmittel im Folgenden kurz eingeführt und erklärt werden.

6.1.1 Java Micro Edition

Die DEMAC-Middleware ist in der Programmiersprache Java in der Version *Java Micro Edition*, der Konfiguration *Connected Device Configuration* und dem Profil *Personal Profile* implementiert. Folglich wurde auch der speziell auf mobile Bedürfnisse abgestimmte Security Service in dieser Form realisiert.

Die Java Micro Edition ist speziell für mobile Geräte ausgerichtet, die über begrenztere Ressourcen als stationäre Desktop-Rechner verfügen, z.B. in Bezug auf Rechenleistung, Speicherkapazität und Netzwerk-Konnektivität. Da sich die Ressourcen und die Leistungsfähigkeit mobiler Geräte auch untereinander stark unterscheiden können, wird die Java Micro Edition in zwei mögliche Konfigurationen unterteilt. Die bereits oben genannte Konfiguration *Connected Device Configuration* (CDC) ist für mobile Geräte vorgesehen, die mindestens einen 32-Bit-Prozessor und 2 MB Speicherplatz vorweisen können. Dagegen ist die *Connected Limited Device Configuration* (CLDC) für mobile Geräte geeignet, die mit 16-Bit- bis 32-Bit-Prozessoren und Speicherkapazitäten von 128-512 KB ausgestattet sind. Die virtuelle Java-Maschine für CLDC wird daher auch als Kilobyte Virtual Machine (KVM) bezeichnet, im Gegensatz zur der Java Virtual Machine für CDC.

Profile stellen eine Erweiterung der genannten Konfigurationen dar, und spezifizieren die Gruppe der unterstützten Geräte genauer in Bezug auf Eigenschaften wie Schnittstellen für die Benutzerinteraktion sowie andere gerätespezifische Merkmale. Profile werden jeweils genau einer Konfiguration zugeordnet. Das hier verwendete *Personal Profile* eignet sich für leistungsfähige mobile Geräte wie PDAs, die auch über eine vielfältige Benutzerschnittstelle verfügen. Auf den einzelnen Profilen können dann noch optionale Pakete aufgesetzt werden, welche die Benutzung von Bluetooth, Web Services oder die Anbin-

dung von Datenbanken ermöglichen. Zusammen bilden die Java Micro Edition, die Java Standard Edition, die Java Enterprise Edition sowie Java Card für Smartcards die weit verbreitete Java 2 Entwicklungsplattform.

6.1.2 Die Kryptographie-Bibliothek Bouncy Castle

Bouncy Castle ist eine frei verfügbare Java-Klassenbibliothek, die eine Implementierung verschiedenster kryptographischer Algorithmen für eine Vielzahl von Java-Versionen zur Verfügung stellt. Sie wird in der vorliegenden Arbeit verwendet, da sie auch eine, speziell für mobile Geräte geeignete, sogenannte *Bouncy Castle Lightweight API* bereitstellt, die sich auf der Plattform Java Micro Edition in den Konfigurationen CLDC und CDC einsetzen läßt. Darüber hinaus ist sie auch mit allen anderen Java-Plattformen bis einschließlich *Java Development Kit 1.6* kompatibel. Benötigte Bouncy-Castle-Klassen können dabei direkt importiert und benutzt werden. Die Dokumentation der Programmierschnittstellen ist jedoch nicht sehr ausführlich. [Hoo05]

6.1.3 JUnit-Testumgebung

Zum Testen der im Rahmen dieser Arbeit geschriebenen Klassen wurde das Framework JUnit verwendet, welches automatisierte Testläufe zur Prüfung der Korrektheit von Java-Programmen ermöglicht. Hierfür ist das Einbinden einer JUnit-Systembibliothek sowie das Schreiben von Testklassen notwendig, die das Soll-Verhalten der zu testenden Klassen mit Hilfe von sogenannten *Assertions* (Zusicherungen) spezifizieren. Diese Vorgehensweise wird *testgetriebene Entwicklung* genannt und zählt zu den *agilen Methoden*. Das grundlegende Ziel dahinter ist es, Fehlerfreiheit dadurch zu gewährleisten, dass nichts implementiert wird, was nicht auch getestet wird (Vollständigkeit der Testfälle). Werden Testklassen erst nach dem eigentlichen Quelltext entwickelt, so besteht die Gefahr, Testfälle zu übersehen. Möchte ein Entwickler fremden Quelltext ändern, so ruft er zuerst alle JUnit-Tests auf und vergewissert sich auf diese Weise über die Fehlerfreiheit des Quelltextes. Treten dann nach eigenen Änderungen Fehler bei erneuten JUnit-Testläufen auf, so weiß der Entwickler, dass er selbst Fehler in die Software eingebaut hat. [Lin05]

Das Testen mit JUnit ist besonders für die Klassen, welche die kryptographische Basisfunktionalität zur Verfügung stellen, von besonderer Wichtigkeit, um sicherzustellen, dass die entwickelten Methoden zur Ver- und Entschlüsselung, zur Berechnung von Hashwerten, zum Padding und zur Generierung von Schlüsselmaterial zuverlässig und konsistent arbeiten. Da die Dokumentation der verwendeten Bouncy Castle Lightweight API nicht den gewünschten Detaillierungsgrad mit sich bringt, stellen wohldefinierte Testmethoden die korrekte Verwendung der Bibliotheksklassen sicher. Darüber hinaus wäre eine mauelle Verifikation der Ergebnisse kryptographischer Algorithmen für Menschen nicht zumutbar. Wie eine JUnit-Testmethode z.B. für die symmetrische Verschlüsselung aussieht, ist in Listing 6.1 dargestellt. Das Listing zeigt eine Methode, die zusichert, dass

ein Klartext, der mit dem gleichen symmetrischen Schlüssel symmetrisch verschlüsselt und wieder entschlüsselt wurde, mit dem Originaltext im unberührten Zustand übereinstimmen muss. Diese Zusicherung muss zu *wahr* ausgewertet werden, damit der JUnit-Test gelingt. Mißlingt der JUnit-Test mit der Meldung *Assertion Failed Error*, so deutet dies auf einen Programmierfehler im getesteten Quelltext hin, in diesem Falle beispielsweise in den Methoden aus den Zeilen 9 und 13.

```
1  public void testSymmetricEncryption()
2  {
3    byte[] symmetricKey =
4      SymmetricKeyGenerator.generateSymmetricKey();
5    byte[] plaintext = "Geheime Nachricht".getBytes();
6
7    //fuehrt die Verschluesselung durch
8    byte[] ciphertext =
9      SymmetricEncryption.symmEncrypt(symmetricKey, plaintext);
10
11   //fuehrt die Entschluesselung durch
12   byte[] decrypted =
13     SymmetricEncryption.symmDecrypt(symmetricKey, ciphertext);
14
15   //testet Originaltext und entschluesselten Text auf Gleichheit
16   assertTrue(plaintext.length == decrypted.length);
17   for(int i = 0 ; i < plaintext.length ; i++)
18   {
19     assertTrue(plaintext[i] == decrypted[i]);
20   }
21 }
```

Listing 6.1: JUnit-Test für symmetrische Verschlüsselung

JUnit ist ein Werkzeug, welches es Entwicklern erleichtert, die Spezifikation ihrer Programme zu verifizieren. Um alle Testklassen und ihre Testmethoden auf einmal aufrufen zu können, um die Tests so automatisiert wie möglich ablaufen lassen zu können, benutzt man die sogenannte *JUnit Testsuite*, welche eine Java-Klasse darstellt, in der alle aufzurufenden Testklassen einzeln eingetragen werden.

6.2 Der DEMAC Security Service

Im Konzeptteil dieser Arbeit (vgl. Kapitel 5) wurde ein Modell für einen dezentralen und selbstorganisierten Securiy Service vorgestellt, welcher für den Einsatz innerhalb

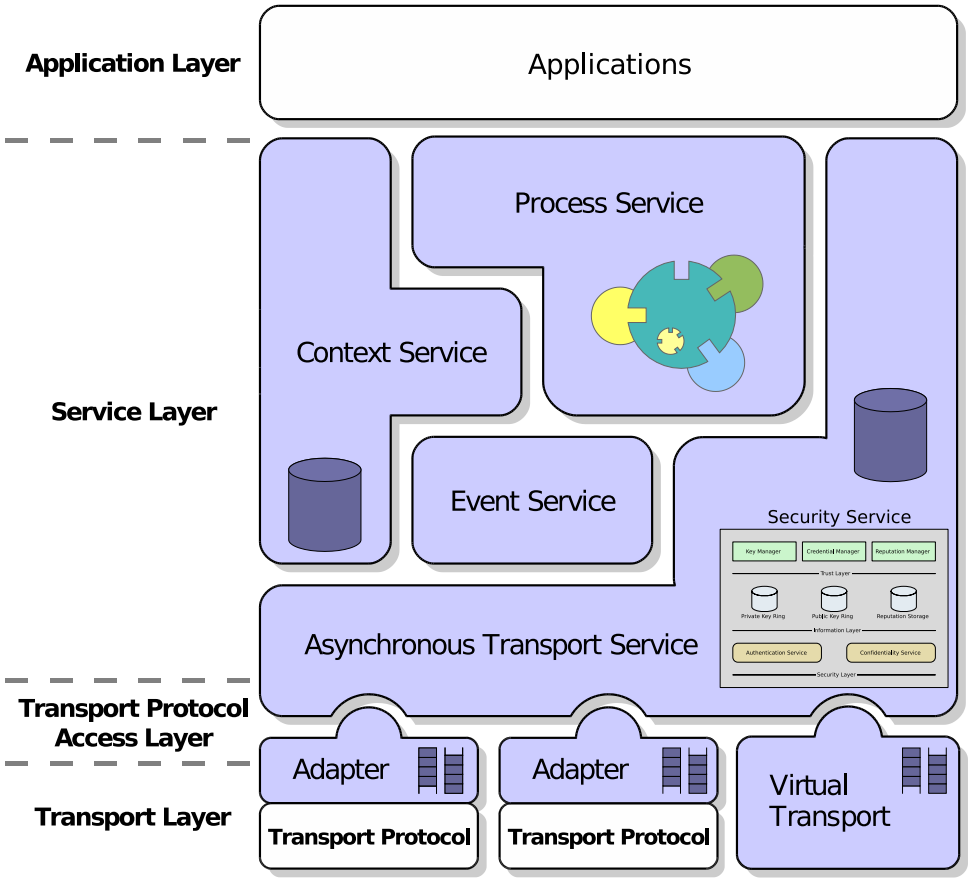

Abbildung 6.1: Einbettung des Security Service in DEMAC

mobiler Middleware entwickelt wurde. In diesem Abschnitt wird die, im Rahmen dieser Arbeit durchgeführte, konkrete Implementierung eines Security Service beschrieben. Dies umfasst einerseits die Erläuterung der internen Programmkomponenten des Security Service und andererseits die Initiierung und den Ablauf der Sicherheits- und Vertrauensdienste zur Laufzeit. Ein grafischer Überblick über die Einbettung des abstrakten Security Service aus Abbildung 5.1 in die abstrakte DEMAC Service-Architektur aus Abbildung 2.3 wird in Abbildung 6.1 dargestellt.

6.2.1 Software-Architektur des DEMAC Security Service

Alle Klassen, die zur Realisierung des Security Service geschrieben werden mußten, sind in Pakete zusammengefasst und in das bereits vorhandene DEMAC-Projektverzeichnis `org.demac.impl.j2me.transport` eingeordnet, da sich dort bereits alle relevanten Klassen der Kommunikationsinfrastruktur, nämlich dem Asynchronous Transport Service, befinden und dies der Schicht entspricht, die im Konzeptteil für die Einbettung des Security Service ausgewählt worden ist und sich daher auch für die Implementierung eignet. Der Quelltext für den DEMAC Security Service gliedert sich in drei Java-Pakete.

Das Paket security.tests

Das Paket `security.tests` umfasst alle JUnit-Testklassen, die im Rahmen der vorliegenden Arbeit zum Testen der Implementierung geschrieben worden sind. Das Paket trägt also nichts zur eigentlichen Funktionalität des Security Service bei und kommt daher auch nicht auf mobilen Endgeräten zur Ausführung, sondern nur auf Rechnern, die zur Entwicklung der DEMAC-Middleware verwendet werden. Erfolgreich durchgeführte Tests gewährleisten jedoch die Zuverlässigkeit des Security Service und sind dementsprechend ein wichtiger Bestandteil der Implementierung. So ist es beispielsweise nach einem Austauch verwendeter kryptographischer Algorithmen sinnvoll, sicher zu stellen, dass weiterhin alle Arbeitsabläufe korrekt ausgeführt werden können.

Das Paket security.crypto

Die von dem Security Service benötigten Klassen für die Verwendung kryptographischer Algorithmen sind in dem Paket `security.crypto` zusammengefasst. Dabei werden in dem Paket keine eigenen kryptographischen Algorithmen implementiert. Vielmehr stellt diese Paket Wrapper-Klassen zur Verfügung, welche die Schnittstellen der Bouncy-Castle-Kryptographie-Bibliothek benutzen und deren spezifische Methoden von der Initialisierung der Verschlüsselungs-*Engines* und Schlüsselgeneratoren bis hin zum Abarbeiten einzelner Datenblöcke und dem Einsatz von Paddingverfahren für die richtige Größe einer Nachricht in eigene, intuitive Methoden kapseln und ihre Benutzbarkeit vereinfachen. Diese Wrapper-Klassen lassen den Security Service von konkreten kryptographischen Algorithmen und Verfahren abstrahieren, da der Security Service die tatsächlich zugrundeliegenden Algorithmen nicht kennt. Diese sind daher jederzeit austauschbar. So ist man auch nicht auf die Benutzung der Bouncy-Castle-Kryptographie-Bibliothek angewiesen, sondern kann jederzeit andere Kryptographie-Provider benutzen, die sich für den Einsatz auf mobilen Geräten eignen. Ein Überblick über die Klassen und Methoden des Pakets ist in Abbildung 6.2 zu sehen und gliedert sich grob in die symmetrische und die asymmetrische Verschlüsselung mit den entsprechenden benötigten Komponenten wie Schlüsselgeneratoren, Verschlüsselungsalgorithmen, Padding- und Prüfsummenverfahren.

Das Paket security

Das Paket `security` bildet die Kernfunktionalität des DEMAC Security Service. Die Architektur richtet sich dabei genau nach dem Modell, welches im Konzeptteil entworfen worden ist (vgl. Abbildung 5.1). Daher wurde für dieses Paket keine eigene Klassenübersicht erstellt. Um jedoch die interne Struktur des implementierten Security Service und seine Einbettung innerhalb des Transport Service verständlich zu machen, werden die wesentlichen Merkmale und Komponenten des Security Serivce in eigenen Abschnitten behandelt.

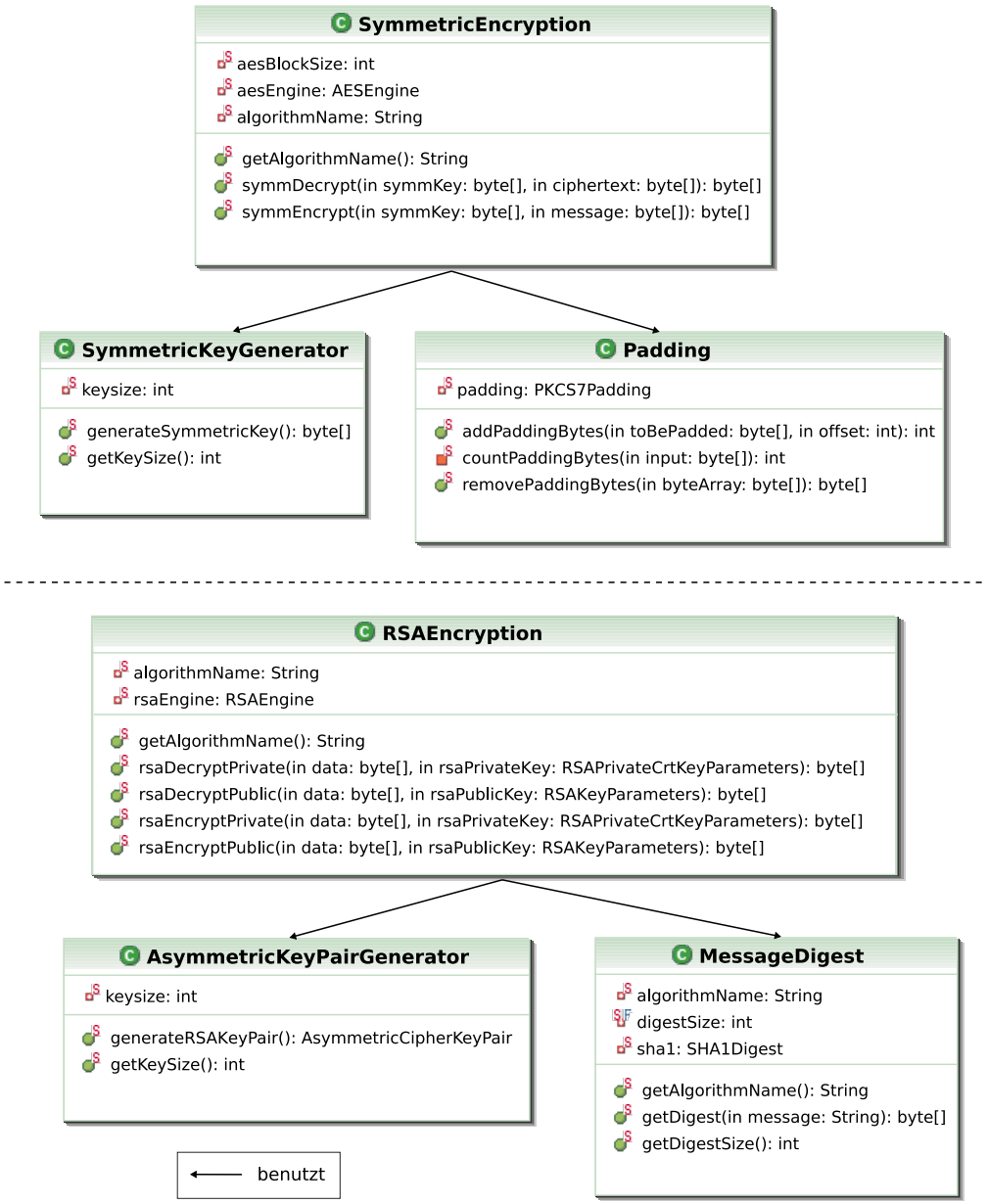

Abbildung 6.2: Die kryptographischen Basisklassen

6.2.2 DEMAC Secure Message

Das im Konzeptteil dieser Arbeit beschriebene spezielle Nachrichtenformat des Security Service (vgl. Abbildung 5.4) wird innerhalb des DEMAC Security Service durch *DEMAC Secure Messages* realisiert. Das Ermöglichen des Sendens und Empfangens einer Secure Message erfordert dabei keine Veränderung der bestehenden darunterliegenden Transportmechanismen des DEMAC Transport Service. Eine Secure Message wird bei gewünschter Verwendung der Sicherheitsmechanismen innerhalb der Nutzdaten einer DEMAC Transport Message, die eine Transportnachricht des Transport Service darstellt,

getunnelt. In Listing 6.2 ist eine gewöhnliche Transportnachricht des DEMAC Transport Service in XML-Format zu sehen, deren Nutzdaten im Klartext (Zeile 6) übertragen werden und auf diese Weise für jeden lesbar sind, der die Nachricht abfängt.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <TransportMessage>
3   <sender>C5828C4A35494CC4B7D5821CA8DAF420</sender>
4   <receiver>88EC727792FD46C28D24A9FDE29F182B</receiver>
5   <MessageHandle>180160E0C3A3424AA3D8A61806731675</MessageHandle>
6   <body>This message should be confidential.</body>
7 </TransportMessage>
```

Listing 6.2: Eine DEMAC Transport Message

Der Security Service ist in der Lage, die Nutzdaten einer solchen Transportnachricht abzusichern. Hierzu benötigt er einen symmetrischen Einmal-Schlüssel für die Nachricht, das eigene asymmetrische Schlüsselpaar und den öffentlichen Schlüssel des Empfängers. Diese Parameter werden dem Security Service von seinen inneren Komponenten wie z.B. dem Key Manager und dem Credential Manager bereitgestellt, so dass die Generierung einer sicheren Transportnachricht erfolgen kann. Um die Funktionalität des Security Service innerhalb des Transport Service zu kapseln und vor anderen Komponenten wie beispielsweise dem DEMAC Process Service zu verbergen, aber dennoch die eben eingeführte DEMAC Secure Message einsetzen zu können, wurde die Klasse *Secure Transport Message* eingeführt, die von der Klasse *Transport Message* erbt. Enthält ein Prozess beispielsweise einen nicht-funktionalen Aspekt, der Verschlüsselung und Authentifizierung beim Versenden vorschreibt, so wird einfach eine Secure Transport Message statt einer gewöhnlichen Transport Message generiert, die vom Transport Service aber ansonsten aufgrund der Vererbung wie eine gewöhnliche Transport Message gehandhabt werden kann. Der Vorteil ist jedoch, dass der Transport Service vor dem Versand seiner Transportnachrichten überprüfen kann, ob es sich um eine Nachricht vom Typ Secure Transport Message handelt. Ist dies der Fall, so wird die Nachricht an den im Transport Service eingebetteten Security Service weitergereicht, der dann seine eigene spezifische Nachricht, die DEMAC Secure Message im Body der Secure Transport Message tunnelt. Listing 6.3 zeigt eine solche Nachricht. Das ursprüngliche Nachrichtenformat der Transport Message aus Listing 6.2 ist dabei erhalten geblieben. Damit der Transport Service des Empfängers der Nachricht jedoch erkennt, dass er diese zunächst an seinen eigenen Security Service zur Entschlüsselung weiterreichen muss, wurde dem originalen Message Handle der ungesicherten Transport Message aus Listing 6.2 der Kennzeichner `demac:security:encrypted` vorangestellt. Neu hinzugekommen ist vor allem die eingebettete Secure Message im Body der Transport Message (Zeilen 9-49), die unter anderem dafür sorgt, dass der Klartext `"This message should be confidential."` aus Listing 6.2 nicht mehr lesbar ist.

Die Zeilen 10-20 benennen die verwendeten kryptographischen Algorithmen und Schlüssellängen zur Absicherung der Nachricht. Der DEMAC Security Service unterstützt im Rahmen dieser Implementierung zwar jeweils nur einen Algorithmus aus den Bereichen der Public-Key-Kryptographie (RSA), der symmetrischen Verschlüsselung (AES) und kryptographischer Prüfsummen (SHA-1), jedoch ermöglicht die Unabhängigkeit und Abstraktion des Nachrichtenformats einer Secure Message von konkreten Algorithmen jederzeit die Erweiterung des Security Service um weitere Algorithmen, deren Parameter bei Benutzung einfach wieder innerhalb der Secure Message mit übermittelt werden müssen. Diese Vorgehensweise wird einerseits der A4-Anforderung nach Autonomie und generischer Verwendbarkeit des Security Service gerecht und erfüllt andererseits auch die A5-Anforderung nach Flexibilität und Erweiterbarkeit (vgl. Abschnitt 5.1).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <TransportMessage>
3    <sender>C5828C4A35494CC4B7D5821CA8DAF420</sender>
4    <receiver>88EC727792FD46C28D24A9FDE29F182B</receiver>
5    <MessageHandle>
6      demac:security:encrypted:180160E0C3A3424AA3D8A61806731675
7    </MessageHandle>
8    <body>
9      <SecureMessage>
10       <Algorithms>
11         <PublicKeyCryptographyParams>
12           <AlgorithmName>RSA</AlgorithmName>
13           <KeySize>512</KeySize>
14         </PublicKeyCryptographyParams>
15         <SymmetricEncryptionParams>
16           <AlgorithmName>AES</AlgorithmName>
17           <KeySize>128</KeySize>
18         </SymmetricEncryptionParams>
19         <MessageDigestAlgorithm>SHA-1</MessageDigestAlgorithm>
20       </Algorithms>
21       <receiver>88EC727792FD46C28D24A9FDE29F182B</receiver>
22       <receiversPublicKey>
23         <RSAKeyParameters>
24           <Exponent>13</Exponent>
25           <Modulus>1614976003</Modulus>
26         </RSAKeyParameters>
27       </receiversPublicKey>
28       <sender>C5828C4A35494CC4B7D5821CA8DAF420</sender>
```

```
29        <sendersPublicKey>
30          <RSAKeyParameters>
31            <Exponent>17</Exponent>
32            <Modulus>363935797</Modulus>
33          </RSAKeyParameters>
34        </sendersPublicKey>
35        <encryptedSymmetricKey>
36          2d53e06731289110e1cda3ffc83b7cf8c2b9ca67b89d59b11aa9b90
37          98ff0e7c65cdc5ecb44b1780cecb5a32ab87425050edb94ef1eb9d3
38          5f06d409a1b74e54d0
39        </encryptedSymmetricKey>
40        <SignatureOfMessageDigest>
41          2cf2753e275fa245992bba52bfaf95c1b33389ffec18927f049319f
42          a36443eb9f554afe0e2e32e7158058de109f2978d2739d3bba7b5b2
43          537996dd097c88eac9
44        </SignatureOfMessageDigest>
45        <encryptedPayload>
46          4fdf7724769bf5c5eb229445eca2e78bd9288f12bfffeb37288569f
47          2a7dc693d
48        </encryptedPayload>
49      </SecureMessage>
50    </body>
51  </TransportMessage>
```

Listing 6.3: Eine eingebettete DEMAC Secure Message

Die Zeilen 22-27 enthalten die notwendigen Parameter zur Rekonstruktion des verwendeten öffentlichen Schlüssels des Empfängers und nicht das Schlüsselobjekt selbst. Laut Programmierschnittstelle der Rekonstruktionskomponente der Bouncy-Castle-API bestehen diese Parameter aus dem Exponenten und dem Modulus des asymmetrischen Schlüssels sowie genau genommen auch aus der Angabe, ob es sich um einen privaten Schlüssel handelt. Da der Security Service aber nicht vorsieht, private Schlüssel zu kommunizieren, und es sich bei übertragenen asymmetrischen Schlüsseln in jedem Falle um öffentliche Schlüssel handeln muss, kann diese Angabe innerhalb einer Secure Message entfallen und wird vom Security Service automatisch angenommen, um die Größe einer Secure Message generell so gering wie möglich halten zu können, um somit der A3-Anforderung nach Leichtgewichtigkeit gerecht zu werden. Die Parameter für den öffentlichen Schlüssel des Senders sind analog in den Zeilen 29-34 enthalten.

Die Zeilen 35-48 spiegeln die Ergebnisse aus einer hybriden Anwendung symmetrischer und asymmetrischer Verschlüsselungsalgorithmen unter Einbezug von Padding- und

Hashing-Verfahren wider. Die hexadezimalen Ziffern bilden dabei die Werte von Byte-Arrays ab, die die verschlüsselten Daten enthalten. Da die Nutzdaten der originalen Nachricht aus Gründen der Performanz mit einem schnellen symmetrischen Algorithmus verschlüsselt werden, muss der verwendete symmetrische Schlüssel ebenfalls an den Empfänger übermittelt werden. Zur Sicherheit wird dieser daher mit dem öffentlichen Schlüssel des Empfängers verschlüsselt und ist in den Zeilen 35-39 enthalten. Über die Nutzdaten wird ein Message Digest erstellt, der die Integrität der Daten sichern soll und hierfür mit dem privaten Schlüssel des Senders signiert wird (Zeilen 40-44). In den Zeilen 45-48 sind dann die mit dem symmetrischen Schlüssel verschlüsselten Nutzdaten enthalten.

Zur besseren Lesbarkeit und Übersicht ist die Formatierung des Listings 6.3 großzügig gewählt. Die Transportnachricht mit eingebetteter Secure Message wirkt dadurch unverhältnismäßig groß im Vergleich zu der ungesicherten Transportnachricht aus Listing 6.2. In der Tat ist eine Secure Transport Message deutlich größer als eine gewöhnliche Transport Message und benötigt sowohl länger für ihre Generierung und Weiterverarbeitung, als auch für die Übertragung über das Netz. Dies sind jedoch Kosten, die man akzeptieren muss, wenn man eine sichere Kommunikationsinfrastruktur in einem System etablieren möchte.

Wie und wann es zum Versand einer DEMAC Secure Message kommt, wird in Abbildung 6.3 schematisch dargestellt. Erfordern beispielsweise die nicht-funktionalen Aspekte eines Prozesses eine Verschlüsselung der Kommunikation zwischen den Kooperationspartnern A und B, so befragt der Security Service A zunächst seinen eigenen Credential Manager A nach dem öffentlichen Schlüssel von B. Ist dieser noch nicht vorhanden, so fordert der Credential Manager A den entsprechenden Schlüssel vom Security Service B an. Der Security Service B sieht seinen eigenen öffentlichen Schlüssel durch den dafür zuständigen Key Manager nach und sendet ihn daraufhin an den anfordernden Credential Manager A zurück. Der Credential Manager speichert den öffentlichen Schlüssel von B in seinem Public Key Ring und leitet den Schlüssel dann an den Security Service A weiter. Nun ist die Basis für den Versand einer Demac Secure Message vorhanden. Wie bereits oben genauer beschrieben, generiert der Security Service A eine solche Nachricht unter Benutzung des erhaltenen öffentlichen Schlüssels von B. Diese Nachricht wird logisch gesehen direkt an den Security Service B gesendet. Der Security Service B benutzt daraufhin seinen eigenen privaten Schlüssel, um die erhaltene Nachricht dem Entschlüsselungsverfahren zu unterziehen. Wurde die originale Nachricht wiederhergestellt, so kann diese an die entsprechende Anwendung, welche die Daten benötigt, weitergereicht werden. Wie in der Abbildung gut zu erkennen ist, findet der komplexe Ablauf des Schlüsselaustausches und der Ver- und Entschlüsselung von Nachrichten verdeckt vor den übergeordneten Akteuren A und B statt, d.h. der Ablauf erfordert keine weitere Benutzerinteraktion.

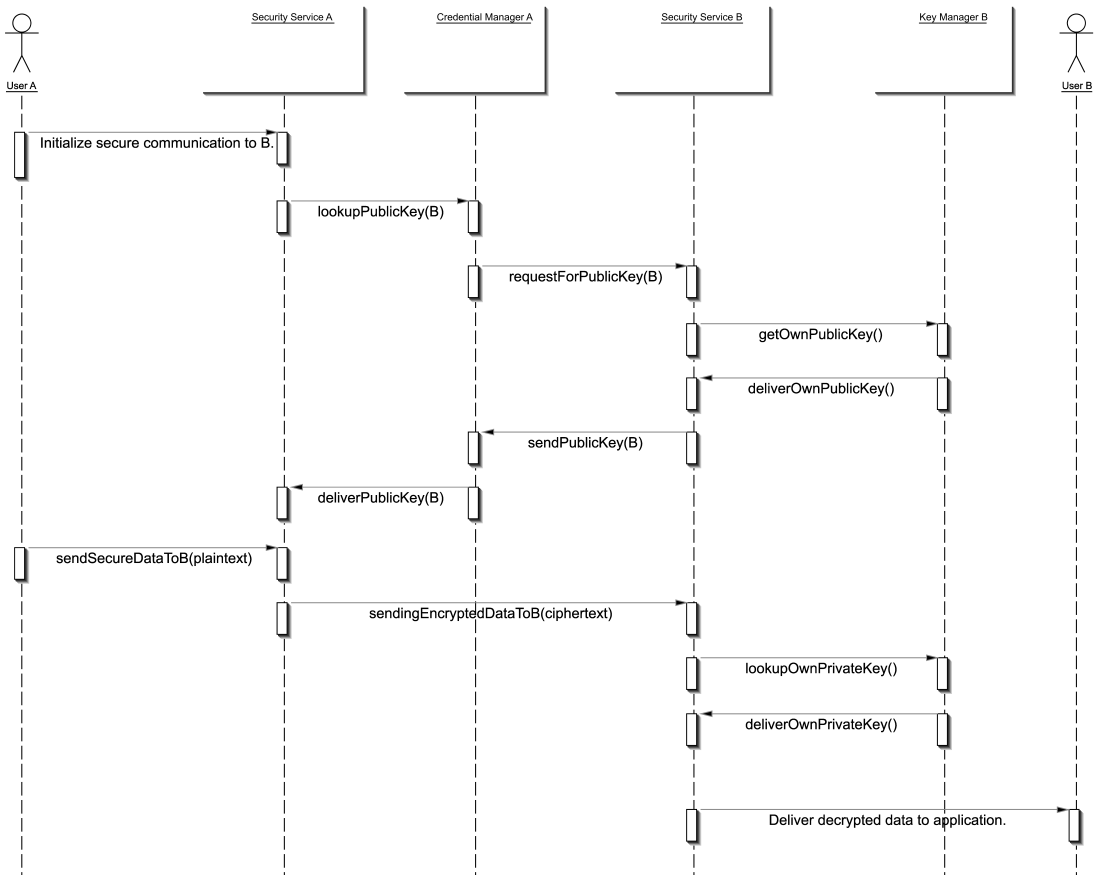

Abbildung 6.3: Sequenzdiagramm für eine vertrauliche Kommunikation

6.2.3 DEMAC Reputation

Das in Abschnitt 5.3.1 vorgestellte Modell eines Reputation Managers wurde als Grundlage für die Implementierung innerhalb des DEMAC Security Service genommen. Der Reputation Manager ist in der Lage, auf Anfrage Reputationen für frühere oder bestehende Kooperationspartner zu generieren. Das eigentliche Feedback wird dabei vom Benutzer auf Anwendungsebene ähnlich dem *ebay*-Bewertungssystem festgelegt. Um den bestehenden Nachrichen-Transportmechanismus des Transport Service erneut nicht verändern zu müssen, wurde die Klasse *Reputation Transport Message*, die analog zu der Klasse Secure Transport Message von der Klasse Transport Message erbt, implementiert. Als Unterscheidungsmerkmal wird dem Message Handle der Reputation Transport Message der Prefix `demac:security:reputation` vorangestellt. Listing 6.4 zeigt eine Reputation Transport Message, wie sie durch den Transport Service versendet werden kann. Im Body der Nachricht befindet sich die eigentliche Reputation (Zeilen 9-30).

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <TransportMessage>
3   <sender>889A489109DE4234A42D909C2D5733EA</sender>
```

```
4   <receiver>68EC727792FD46C28D24A9FDE29F182B</receiver>
5   <MessageHandle>
6     demac:security:reputation:35416CE0C3A34BE443D8A61806731675
7   </MessageHandle>
8   <body>
9     <Reputation>
10      <issuer>889A489109DE4234A42D909C2D5733EA</issuer>
11      <issuedTo>68EC727792FD46C28D24A9FDE29F182B</issuedTo>
12      <issuersPublicKey>
13        <RSAKeyParameters>
14          <Exponent>17</Exponent>
15          <Modulus>363935797</Modulus>
16        </RSAKeyParameters>
17      </issuersPublicKey>
18      <signatureAlgorithm>RSA</signatureAlgorithm>
19      <messageDigestAlgorithm>SHA-1</messageDigestAlgorithm>
20      <signatureOnReputationDigest>
21        25008768f2194229680ea9aeca7e261473229441fa7ec0a39b106af
22        377dc4cff15cd2279c5714f5b2e5cfcc99d712f7986db253e2f052b
23        c26d4b3b9336b3b455
24      </signatureOnReputationDigest>
25      <typeOfInteraction>General</typeOfInteraction>
26      <recommendation>
27        Process execution suceeded. Recommended and
28        trustworthy co-operation partner.
29      </recommendation>
30    </Reputation>
31  </body>
32 </TransportMessage>
```

Listing 6.4: Eine eingebettete DEMAC Reputation

In Zeile 10 wird der Aussteller der Reputation angegeben, während Zeile 11 denjenigen benennt, für den die Reputation ausgestellt worden ist. Die Art der Interaktion, die zwischen beiden stattgefunden hat, wird in Zeile 25 angegeben. Es folgt die eigentliche Bewertung in den Zeilen 26-29. Der Aussteller erstellt eine Signatur über den Hashwert der Reputation zum Schutz ihrer Integrität, die er mit seinem privaten Schlüssel verschlüsselt (Zeilen 20-24). Die Zeilen 18-19 geben dabei die verwendeten Verfahren an, nämlich den RSA-Algorithmus und das SHA-1-Hashverfahren. Sollte die Authentizität einer Reputation aufgrund des fehlenden öffentlichen Schlüssels des Ausstellers nicht verifizierbar sein, so kann sie gegenüber verifizierten Reputationen nur mit geringerer Gewichtung und Glaubwürdigkeit in die Evaluierung mit eingehen.

7 Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde ein dezentraler Security Service für mobile Middleware entwickelt, der speziell für den Einsatz in selbstorganisierten, hochdynamischen und kontextsensitiven mobilen Umgebungen, die zum Zweck der Kooperation bestehen, ausgerichtet ist und die Anforderungen bezüglich der Sicherheit mobiler Interaktionen und Transaktionen in solchen Systemen berücksichtigt. Bei der Entwicklung des Konzeptes wurde besonderer Wert auf eine dezentralisierte Architektur des Security Service gelegt, um den Einsatz in verteilten mobilen Umgebungen zu ermöglichen und zentrale Komponenten möglichst zu eliminieren. Dies ermöglicht eine hohe Verfügbarkeit der Dienste und vermeidet einen *Single-Point-of-Failure*. Dabei wurden die Rahmenbedingungen des Mobile Computing, wie beispielsweise die begrenzte Prozessorleistung, geringe Speicherkapazitäten und häufige Verbindungsabbrüche beachtet. Diese Rahmenbedingungen sind der Grund dafür, weshalb Sicherheitsmechanismen in mobilen Systeme kryptographisch gesehen meist schwächer sind als auf stationären Rechnern und in fest verkabelten Netzen.

In der vorliegenden Arbeit wurden dabei keine Sicherheitsmechanismen oder Verschlüsselungsalgorithmen neu erfunden. Vielmehr wurden bereits bestehende Konzepte und Verfahren bezüglich Sicherheit und Vertrauen aus stationären und mobilen Systemen untersucht, miteinander kombiniert und für den speziellen Einsatz in dezentralisierten, mobilen und kontextsensitiven Systemen, die für eine langfristige Kooperation der Teilnehmer zur Erreichung jeweils eigener Ziele ausgelegt sind, angepasst. Da diese Kooperation unter anderem eine Ansammlung und einen Austausch sensibler Daten und Informationen bedeutet, müssen solche sicherheitsrelevanten Interaktionen unter Teilnehmern abgesichert werden. Das heißt wiederum, dass grundlegende Dienste wie die Verschlüsselung der Kommunikation, die Authentifizierung von Kooperationspartnern und der Aufbau von Vertrauensbeziehungen zur Gewährleistung von Vertraulichkeit, Integrität, Authentizität und Nicht-Bestreitbarkeit vorhanden sein müssen.

Intrinsische Probleme kryptographischer Verfahren, wie beispielsweise die sichere Schlüsselverteilung über unsichere Kanäle, eine verifizierbare Bindung von realen Personen an öffentliche Schlüssel, der Einsatz zu kurzer Schlüssellängen sowie der Berechnungs- und Kommunikationsaufwand im Allgemeinen, bleiben in mobilen Systemen bestehen und sind in ihrer Ausprägung noch gravierender, da nur stark beschränkte Ressourcen zur Verfügung stehen, diesen Problemen entgegen zu wirken. Der Aufbau von Vertrauen zwischen zwei fremden Entitäten im Hinblick auf eine positive Erwartungshaltung bezüglich des Verhaltens des Gegenübers, über das es keinerlei Vorwissen gibt (z.B. in mobilen Ad-Hoc-Netzen), ist ein schwieriger Punkt, da man ohne Erfahrungswerte keine

treffende Einschätzung und Bewertung des Kooperationspartners machen kann.

Der konzipierte Security Service adressiert die genannten Punkte. Um die Realisierbarkeit des entwickelten Security Service zu demonstrieren, wurde dieser im Rahmen des DEMAC-Forschungsprojekts prototypisch implementiert. Die DEMAC-Middleware verfügt nun über ein grundlegendes Gerüst zum Absichern ihrer Interaktionen mit gleichartigen Entitäten. Ausblickend sind jedoch noch Verfeinerungen und Erweiterungen der Funktionalität denkbar. So können beispielsweise verschiedene Profile entwickelt werden, die je nach Sicherheitsanforderung eines Prozesses oder abhängig von der Anwendung, die ausgeführt wird, greifen. Auch Konzepte zur Evaluation, Verwendung und Darstellung von Reputationen *auf Anwendungsebene*, die einem Benutzer einen schnellen, aber zuverlässigen Eindruck eines zweiten Benutzers vermitteln, könnten in zukünftigen Arbeiten auf diesem Gebiet vertieft werden.

Abschließend kann man sagen, dass es für die Sicherheit eines Systems nicht ausreichend ist, diese einmal zu implementieren, und dann sich selbst zu überlassen. Ein ständiges *Monitoring*, die regelmäßige Überprüfung von neu erwachsenen Anforderungen sowie die Aufrüstung mit aktuelleren, besseren Verfahren und die Schulung der Benutzer tragen zur ganzheitlichen Sicherheit eines Systems bei.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[Abd06] ABDALLA, Samer: *Standards und Risiken drahtloser Kommunikation. Risikoanalyse des IEEE 802.11 Standards (WLAN)*. VDM Verlag, 2006

[ADB+99] ABOWD, Gregory D. ; DEY, Anind K. ; BROWN, Peter J. ; DAVIES, Nigel ; SMITH, Mark ; STEGGLES, Pete: Towards a Better Understanding of Context and Context-Awareness. In: *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag, 1999. – ISBN 3–540–66550–1, S. 304–307

[ARH97] ABDUL-RAHMAN, Alfarez ; HAILES, Stephen: A distributed trust model. In: *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*, ACM Press, 1997, S. 48–60

[BCG04] BASAGNI, Stefano ; CONTI, Marco ; GIORDANO, Silvia: *Mobile Ad Hoc Networking*. John Wiley & Sons, 2004

[Beu05] BEUTELSPACHER, Albert: *Kryptologie: Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen*. Wiesbaden : Vieweg-Verlag, 2005

[BFL96] BLAZE, Matt ; FEIGENBAUM, Joan ; LACY, Jack: Decentralized Trust Management. In: *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*. Washington, DC, USA : IEEE Computer Society, 1996. – ISBN 0–8186–7417–2, S. 164

[Bis05] BISHOP, Matt: *Introduction to Computer Security*. Pearson Education, 2005

[BS03] BERESFORD, Alastair R. ; STAJANO, Frank: Location Privacy in Pervasive Computing. In: *IEEE Pervasive Computing* 2 (2003), Nr. 1, S. 46–55

[Cap04] CAPRA, Licia: Engineering human trust in mobile system collaborations. In: *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*. New York, NY, USA : ACM Press, 2004. – ISBN 1–58113–855–5, S. 107–116

[Cra02] CRANOR, Lorrie F.: *Web Privacy with P3P*. O'Reilly Media, 2002

[DH76] DIFFIE, Whitfield ; HELLMAN, Martin E.: New Directions in Cryptography. In: *IEEE Transactions on Information Theory* IT-22 (1976), Nr. 6, 644–654. `citeseer.ist.psu.edu/diffie76new.html`

[DH95] DIEHL, Norbert ; HELD, Albert: *Mobile Computing - Systeme, Kommunikation, Anwendungen*. International Thomson Publishing GmbH, 1995

[DVP+02] DAMIANI, Ernesto ; VIMERCATI, De C. ; PARABOSCHI, Stefano ; SAMARATI, Pierangela ; VIOLANTE, Fabio: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. New York, NY, USA : ACM Press, 2002. – ISBN 1–58113–612–9, S. 207–216

[EC05] ERTAUL, Levent ; CHAVAN, Nitu: Security of Ad Hoc Networks and Threshold Cryptography. In: *International Conference on Wireless Networks, Communications, and Mobile Computing* 1 (2005), S. 69–74

[Eck03] ECKERT, Claudia: Mobil, aber sicher! In: MATTERN, Friedemann (Hrsg.): *Total vernetzt - Szenarien einer informatisierten Welt*. Springer-Verlag, May 2003

[Eck06] ECKERT, Claudia: *IT-Sicherheit. Konzepte - Verfahren - Protokolle*. München, Wien : Oldenbourg Verlag, 2006

[ED06] EREN, Evren ; DETKEN, Kai-Oliver: *Mobile Security - Risiken mobiler Kommunikation und Lösungen zur mobilen Sicherheit*. Carl Hanser Verlag, 2006

[Gam00] *Kapitel* Trust: Making and Breaking Cooperative Relations. In: GAMBETTA, Diego: *Can We Trust Trust?* Department of Sociology, University of Oxford, 2000, 213–237

[GH01] GERHOLD, Diethelm ; HEIL, Helmut: Das neue Bundesdatenschutzgesetz. In: *Datenschutz und Datensicherheit* 25 (2001), Nr. 7

[Hob98] HOBERT, Guido: *Datenschutz und Datensicherheit im Internet: Interdependenz und Korrelation von rechtlichen Grundlagen und technischen Möglichkeiten*. New York, NY, USA : P. Lang Publishing Co., 1998. – ISBN 3631339925

[Hoo05] HOOK, David: *Beginning Cryptography with Java*. John Wiley & Sons, 2005

[Jen02] JENDRICKE, U.: *Sichere Kommunikation zum Schutz der Privatsphäre durch Identitätsmanagement*, Diss., 2002

[Jø96] JØSANG, Audun: The right type of trust for distributed systems. In: *NSPW '96: Proceedings of the 1996 workshop on New security paradigms*. New York, NY, USA : ACM Press, 1996. – ISBN 0–89791–944–0, S. 119–131

[Kli05] KLIMA, Vlastimil: *Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications*. Cryptology ePrint Archive, Report 2005/102, 2005

[Kun05] KUNZE, Christian P.: Unterstützung mobiler Prozesse im Mobile Computing. In: DRESSLER, Falko (Hrsg.) ; KLEINÖDER, Jürgen (Hrsg.) ; Universität Erlangen-Nürnberg (Veranst.): *Technischer Bericht zum 1. GI/ITG KuVS Fachgespräch Energiebewusste Systeme und Methoden* Universität Erlangen-Nürnberg, 2005, S. 42–47

[KZL06] KUNZE, Christian P. ; ZAPLATA, Sonja ; LAMERSDORF, Winfried: Mobile Process Description and Execution. In: ELIASSEN, Frank (Hrsg.) ; MONTRESOR, Alberto (Hrsg.): *Proceedings of the 6th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, Springer-Verlag, 2006, S. 32 – 47

[KZL07] KUNZE, Christian P. ; ZAPLATA, Sonja ; LAMERSDORF, Winfried: Mobile Processes: Enhancing Cooperation in Distributed Mobile Environments. In: *Journal of Computers* 2 (2007), 2, Nr. 1, S. 1–11

[Lin05] LINK, Johannes: *Softwaretests mit JUnit*. dpunkt Verlag, 2005

[MÖ3] MÖLLER, Jan: *Stellungnahme zu juristischen Aspekten des P3P-Einsatzes in mobilen Endgeräten*. Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, 2003. `https://www.datenschutzzentrum.de/projekte/p3p/Gutachten_Mobilgeraete.pdf`. – Abruf: 24. Mai 2007

[MC01] MCKNIGHT, D. H. ; CHERVANY, Norman L.: Trust and Distrust Definitions: One Bite at a Time. In: *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference*. London, UK : Springer-Verlag, 2001. – ISBN 3–540–43069–5, S. 27–54

[MC04] MONTENEGRO, Gabriel ; CASTELLUCCIA, Claude: Crypto-based identifiers (CBIDs): Concepts and applications. In: *ACM Trans. Inf. Syst. Secur.* 7 (2004), Nr. 1, S. 97–127. `http://dx.doi.org/http://doi.acm.org/10.1145/984334.984338`. – DOI http://doi.acm.org/10.1145/984334.984338. – ISSN 1094–9224

[MDM05] MERWE, Johann van d. ; DAWOUD, Dawoud ; MCDONALD, Stephen: Fully self-organized peer-to-peer key management for mobile ad hoc networks. In: *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*. New York, NY, USA : ACM Press, 2005. – ISBN 1–59593–142–2, S. 21–30

[MDS95] MAYER, R. C. ; DAVIS, J. H. ; SCHOORMAN, F. D.: An Integrative Model of Organizational Trust. In: *The Academy of Management Review* 20 (1995), Nr. 3

[NA99] NETWORK ASSOCIATES, Inc.: *An Introduction to Cryptography*, 1999. `http://www.pgpi.org/doc/guide/6.5/en/intro/`. – PGP-Dokumentation, Abruf 10.05.2007

[OR01] O'SHEA, Greg ; ROE, Michael: Child-proof authentication for MIPv6 (CAM). In: *SIGCOMM Comput. Commun. Rev.* 31 (2001), Nr. 2, S. 4–8. `http://dx.doi.org/http://doi.acm.org/10.1145/505666.505668`. – DOI http://doi.acm.org/10.1145/505666.505668. – ISSN 0146–4833

[Pac04] PACE: An Architectural Style for Trust Management in Decentralized Applications. In: *WICSA '04: Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*. Washington, DC, USA : IEEE Computer Society, 2004. – ISBN 0–7695–2172–X, S. 221

[RBB03] ROTHERMEL, Kurt ; BAUER, Martin ; BECKER, Christian: Digitale Weltmodelle - Grundlage kontextbezogener Systeme. In: MATTERN, Friedemann (Hrsg.): *Total vernetzt - Szenarien einer informatisierten Welt*. Springer-Verlag, May 2003

[RBBC98] ROUSSEAU, Denise M. ; BITKIN, Sim B. ; BURT, Ronald S. ; CAMERER, Colin: Not So Different After All: A Cross-Discipline View of Trust. In: *The Academy of Management Review* 23 (1998), 7, Nr. 3

[Res00] RESCORLA, Eric: *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, 2000

[Rot05] ROTH, Jörg: *Mobile Computing - Grundlagen, Technik, Konzepte*. dpunkt Verlag GmbH, 2005

[RZFK00] RESNICK, Paul ; ZECKHAUSER, Richard ; FRIEDMAN, Eric ; KUWABARA, Ko: Reputation Systems: Facilitating Trust in Internet Interactions. In: *Communications of the ACM* 43 (2000), Nr. 12

[Sat96] SATYANARAYANAN, M.: Fundamental Challenges in Mobile Computing. In: *PODC '96: Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, ACM Press, 1996, S. 1–7

[Sat03] SATYANARAYANAN, M.: Privacy: The Achilles Heel of Pervasive Computing? In: *IEEE Pervasive Computing* 2 (2003), Nr. 1, S. 2–3

[Sch05] SCHWENK, Jörg: *Sicherheit und Kryptographie im Internet. Von sicherer E-Mail bis zu IP-Verschlüsselung*. Ruhr-Universität Bochum : Vieweg-Verlag, 2005

[SDET06] SURYANARAYANA, Girish ; DIALLO, Mamadou H. ; ERENKRANTZ, Justin R. ; TAYLOR, Richard N.: Architectural support for trust models in decentralized applications. In: *ICSE '06: Proceeding of the 28th international conference on Software engineering*. New York, NY, USA : ACM Press, 2006. – ISBN 1–59593–375–1, S. 52–61

[SLB06] SHERWOOD, Rob ; LEE, Seungjoon ; BHATTACHARJEE, Bobby: Cooperative peer groups in NICE. In: *Comput. Networks* 50 (2006), Nr. 4, S. 523–544. http://dx.doi.org/http://dx.doi.org/10.1016/j.comnet.2005.07.012. – DOI http://dx.doi.org/10.1016/j.comnet.2005.07.012. – ISSN 1389–1286

[SS01] SABATER, Jordi ; SIERRA, Carles: REGRET: Reputation in regarious societies. In: *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*. New York, NY, USA : ACM Press, 2001. – ISBN 1–58113–326–X, S. 194–195

[Sta02] STAJANO, Frank: *Security for Ubiquitous Computing*. John Wiley & Sons, 2002

[Sta03] STALLINGS, William: *Cryptography and Network Security: Principles and Practices*. Pearson Education, 2003

[SU06] SAVOLA, Reijo ; UUSITALO, Ilkka: Towards Node-Level Security Management in Self-Organizing Mobile Ad Hoc Networks. In: *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*. Washington, DC, USA : IEEE Computer Society, 2006. – ISBN 0–7695–2522–9, S. 36

[Tan00] TANENBAUM, Andrew S.: *Computernetzwerke*. Pearson Studium, 2000

[Tur06] TURJALEI, Mirwais: *Integration von Context-Awareness in eine Middleware für mobile Systeme*, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, Diplomarbeit, 6 2006

[Tyn05] TYNAN, Dan: *Computer Privacy Annoyances*. O'Reilly Media, 2005

[Vac07] VACCA, John R.: *Practical Internet Security*. New York : Springer-Verlag, 2007

[Wei91] WEISER, Mark: The Computer for the 21st century. In: *Scientific American* 265 (1991), Nr. 3, S. 94–104

[WYY05] WANG, Xiaoyun ; YIN, Yiqun L. ; YU, Hongbo: Finding collisions in the full SHA-1. In: *CRYPTO*, 2005, 17–36

[Zap05] ZAPLATA, Sonja: *Prozessintegration in Middleware für mobile Systeme*, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, Diplomarbeit, 10 2005

[ZN06] ZHENG, Pei ; NI, Lionel M.: *Smart Phone and Next-Generation Mobile Computing*. Morgan Kaufmann Publishers, 2006

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt und mich anderer als der im beigefügten Verzeichnis angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht.
Außerdem erkläre ich, dass ich mit der Einstellung dieser Diplomarbeit in den Bestand der Bibliotheken der Universität Hamburg einverstanden bin.

Hamburg, September 2007

Alice Winnicki