Universität Ulm, Fakultät für Ingenieurwissenschaften und Informatik
Institut für Medieninformatik
Leiter: Prof. Dr. Michael Weber

Privacy- und Sicherheitsaspekte in ubiquitären Umgebungen

Dissertation

zur Erlangung des Doktorgrades
Dr. rer. nat.
der Fakultät für Ingenieurwissenschaften und Informatik
der Universität Ulm

vorgelegt von
Stefan Schlott
aus Wiesensteig

2008

Amtierender Dekan (Zeitpunkt der Einreichung):	Prof. Dr. Helmuth Partsch
Amtierender Dekan (Tag der Promotion):	Prof. Dr. Michael Weber
1. Gutachter:	Prof. Dr. Michael Weber
2. Gutachter:	Prof. Dr.-Ing. Franz J. Hauck
Tag der Promotion:	21.11.2008

Zusammenfassung

Ubiquitous Computing ist die Vision einer nahen Zukunft, in welcher die Welt des Alltags von Computern, die nicht mehr als solche wahrgenommen werden, durchdrungen sein wird. Dem Gewinn an Lebensqualität steht jedoch die potentielle Gefahr für die Privatsphäre gegenüber: Durch eine automatische, unbemerkte Kommunikation der Geräte verlieren die Benutzer Kontrolle und Gefühl für die Weitergabe von Informationen.

In dieser Arbeit wird zunächst mit Hilfe von Simulationen die Gefährdung der Privatsphäre in einem universellen Angriffsszenario eruiert. Im Anschluß daran stellt diese Arbeit Möglichkeiten vor, welche es mobilen Geräten erlaubt, komplett anonym zu agieren. Der Widerspruch zwischen Anonymität und einer authentischen Verschlüsselung wird dabei besonders gewürdigt. Das neue Problem der gegenseitigen Wiedererkennung anonymer Geräte wird zusammen mit einem effizienten Lösungsansatz aufgezeigt.

In einem weiteren Schritt stellt die Arbeit eine Möglichkeit vor, wie die Kommunikation von Anwendungsprogrammen mit anderen Geräten in Bezug auf die Privatsphäre und die weitergegebenen Informationen beobachtet werden kann. So ist es möglich, die Kommunikation gegen Privacy-Richtlinien des Benutzers zu überprüfen und bei Verstoß die Kommunikation zu unterbinden.

Abstract

Ubiquitous Computing is a vision of a near future, in which everyday life will be pervaded with computers which are no longer perceived as separate, technical devices. The gain of quality of life faces a potential danger to privacy: Automatic, unnoticed communication makes the users lose perception and control of propagation of information.

This thesis elicts the privacy endangerment in a generic attack scenario by evaluation of simulations. Further, this thesis introduces ways which allow mobile devices to act completely anonymous. The discrepancy between anonymity and an authenitc encryption is discussed in detail. The new problem of mutual reidentification of anonymous devices is introduced, along with an efficient solution.

Based on this anonymity, a method for monitoring the communication of the application regarding privacy is introduced. This allows checking (and in case of a violation the interruption) the communication against the user's privacy preferences.

Inhaltsverzeichnis

1 Einleitung und Ziele

> I wish that we lived in a golden age, where ethical behavior was assumed; where technically competent programmers respected the privacy of others; where we didn't need locks on our computers.
>
> *Clifford Stoll, 1989 [217]*

1.1 Die Vision des Ubiquitous Computing

Computer dringen immer weiter in unser Alltagsleben ein. In den meisten Berufszweigen sind sie nicht mehr wegzudenken, und in einem großen Teil der deutschen Haushalte befindet sich mindestens ein PC – viele davon mit Zugang zum Internet. Aktueller Trend ist die Vernetzung mit Hilfe von Funk-Technologien: DSL-Router werden mit integrierter WaveLAN-Basisstation ausgeliefert; Handys, Laptops und PDAs besitzen Bluetooth-Schnittstellen zur Kommunikation untereinander oder zum drahtlosen, bequemen Anschluß von weiterem Zubehör wie beispielsweise Headsets.

Die aktuelle Forschung und Entwicklung gibt einen Ausblick auf eine mögliche nahe Zukunft: Kleidung wird mit waschbarer Elektronik versehen, die den Träger unterwegs mit Radio und Musik versorgt. Kostengünstige Miniatursensoren mit Funkschnittstelle liefern Statusinformationen über die verschiedensten Dinge, vom Feuchtigkeitsgrad im Mauerwerk über den Alterungsprozeß der Autoreifen bis hin zur Raumtemperatur. Haushaltsgeräte wie Waschmaschine, Backofen und Kühlschrank werden Computer enthalten, die über ihren Inhalt Buch führen, ihn überwachen und den Besitzer über besondere Ereignisse informieren.

Zu diesen direkten Dienstleistungen von Geräten wird sich eine weitere Art von Gegenständen in unserem Alltag einfinden: Gegenstände, die neben ihrer Alltagsfunktion eine weitere Funktion als Ein- oder Ausgabegerät für die Welt der Computer besitzen werden – in Labors und Universitäten existieren bereits Projekte [17, 18, 120], in welchen mit solchen Techniken experimentiert wird. Der Kugelschreiber dient nicht mehr nur als Schreibgerät, sondern auch als Ersatz für den Mauszeiger und – mit Hilfe von Sensoren für Schreibrichtung und Andruck – zur Authentisierung mittels Unterschrift. Die Aktenmappe wird nicht mehr lediglich Papier enthalten, sondern auch die zugehörigen elektronischen Dateien. Ein PDA, der heute lediglich eine elektronische Form des Kalenders und des Adressbuchs darstellt, wird durch Kommunikation mit der Umgebung zum Sichtschirm für Geräte und Dienste der nahen Umgebung.

Diese nahe Zukunft wurde von Mark Weiser „Ubiquitous Computing“ getauft [234], ebenso verfolgt David Norman in seinem Buch „The Invisible Computer“ [167] diese Zukunftsideen.

Die Zukunft wird zeigen, welche dieser Anwendungen sich durchsetzen und welche als spleenige Ideen der Forschung wieder verschwinden werden. Doch bereits heute gehören solche Gegenstände in sehr begrenztem Umfang zu unserem Alltag:

Die EC-Karte der Bank trägt als Aufschrift Name und Kontonummer des Inhabers und dient mit Hilfe der darauf aufgetragenen Unterschrift als Hilfsmittel zur Authentisierung; gleichzeitig kann auf dem Chip dieser Geldkarte Bargeld in elektronischer Form transportiert und ausgetauscht werden. Ähnliches gilt für die Karten der Krankenkassen. Der Reisepass [145, 146, 128] und in Bälde auch der Personalausweis beinhalten einen RFID[1]-Chip, welcher über eine drahtlose Schnittstelle die Ausweisinformationen einem authentisierten Lesegerät übermittelt. Ebenfalls der RFID-Technik bedienen sich Etiketten auf Produkten, welche sowohl der Vereinfachung der Logistik in der Verkaufs- und Versandkette dienen, als auch als Diebstahlsicherung im Laden fungieren. Einige Kleidungshersteller planen sogar, diese Funkchips schon bei der Herstellung in die Kleidungsstücke einzunähen; andere Hersteller überlegen, ob mit Hilfe dieser Technik in Zukunft Garantieleistungen ohne Kassenbeleg möglich sind, da alle benötigten Informationen auf dem Funkchip abgelegt oder mit diesem assoziiert bei Händler und Hersteller gespeichert werden.

All diese Ansätze haben mehrere Gemeinsamkeiten:

- Die Benutzer werden die Präsenz dieser Technik immer weniger wahrnehmen. Für sie wird die Dienstleistung bzw. der gebotene Mehrwert im Fokus stehen.
- Mit der Zunahme der Anzahl solcher Geräte und deren Verschmelzung mit dem Alltag muß der benötigte Konfigurationsaufwand schwinden. Geräte, die erst eine aufwendige Einrichtung und Installation benötigen, werden von der Masse nicht akzeptiert werden.
- Die Geräte kommunizieren mit ihrer Umgebung. Mit Ausnahme stationärer Geräte ist eine Vernetzung per Kabel nicht möglich; aus Gründen des Komforts ist sie wahrscheinlich auch bei stationären Geräten nicht erwünscht. Daher wird die Kommunikation über eine unsichtbare Funkschnittstelle vonstatten gehen.
- Die Kommunikation mit der näheren Umgebung impliziert für die Entwicklung hin zu einer ubiquitären Umgebung, daß es von Vorteil ist, wenn keine zentrale Infrastruktur benötigt wird. Nur wenn es die Technik gestattet, sich von kleinen, technisch ausgestatteten Inseln beginnend, auszudehnen, ist eine Verbreitung des Ubiquitous Computing wahrscheinlich.

[1] RFID: Radio Frequency Identification. Funkschnittstelle, welche auch Chips ohne Energiequelle über eine Trägerwelle mit Energie speisen kann.

1.2 Die Privacy-Problematik

Die Verbindung von frei kommunizierenden Geräten mit dem Anspruch auf geringe Konfiguration und möglichst seltener Benutzerinteraktion stellt insbesondere an den Datenschutz und die Datensicherheit neue Herausforderungen; dies erwähnte Mark Weiser bereits in einer späteren Arbeit über Ubiquitous Computing [235].

Schon heute, bei der regelmäßigen Benutzung des Internets, hinterlassen viele unbedarfte Nutzer breite Datenspuren, die von den Firmen emsig gesammelt und ausgewertet werden. Selbst wenn die Daten pseudonymisiert gespeichert werden, erlaubt die Menge der Informationen noch immer eine sehr genaue Analyse, wenn nicht sogar eine eindeutige Zuordnung [90]. Daß mitunter schon eine geringe Anzahl an Merkmalen genügt, wurde bereits bei der Volkszählung 1983 exemplarisch vorgerechnet: Innerhalb einer Testdatei mit den Daten von 100.000 Personen genügten sechs Kriterien (von welchen jedes für sich nicht auffällig war), um die Auswahl auf drei Personen einzuschränken [38, 39].

Sollte sich die Vision des Ubiquitous Computings weiter verwirklichen, wird die Menge an Informationen nochmals exponentiell steigen. Daraus entstehende Datensammlungen generieren den oft zitierten „gläsernen Bürger“ mit Leichtigkeit. Trotzdem spielt das Thema Privacy bei den aktuellen EU-Projekten zum Ubiquitous Computing [84, 190] nur eine geringe Rolle.

Das Sammeln von Daten ist für den Betroffenen nicht wahrnehmbar, er spürt die Auswirkungen erst stark verzögert und allenfalls indirekt. Politik- und Soziologiewissenschaftler statuieren, daß ein Leben in einem Zustand ständiger (potentieller) Überwachung unseren demokratischen Prinzipien widerspricht: Wer sich selbst aus Angst vor Beobachtung in seinem Verhalten verbiegt, der lebt nicht mehr in Freiheit [89].

Daher warnen sowohl aktuelle wissenschaftliche Arbeiten vor den Privacy-Problemen ubiquitärer Techniken (z. B. [152, 169]), genauso wie eine aktuelle Studie zur Technikfolgenabschätzung im Ubiquitous Computing zu dem Schluß kommt: „Ohne Datenschutz kein Ubiquitäres Computing!“ [31].

Geräte, die man ständig bei sich trägt bzw. solche, die sensitive Informationen beinhalten, sollten nicht allzu freigiebig mit Informationen sein – nur allzu leicht lassen sich sonst detaillierte Bewegungs- und Persönlichkeitsprofile erzeugen. Auf der anderen Seite werden häufige Sicherheitsabfragen vom Benutzer als lästig empfunden; einige der Geräte werden mangels Ausgabemöglichkeit gar nicht in der Lage sein, eine solche darzustellen.

Um eine harmonische Zusammenarbeit von Mensch und Gerät zu erreichen, ist es vonnöten, ein entsprechendes Sicherheitskonzept für die Kommunikation zu implementieren. Dieses Konzept muß über die gewöhnlichen Forderungen nach einer abhörsicheren Übertragung hinausgehen. Es muß die Präferenzen und Wünsche des

Benutzers bezüglich der Übertragungssicherheit und der Sensitivität der übertragenen Daten kennen und durchsetzen können. Im Idealfall kann das Gerät anhand einer Klassifizierung des benutzten Dienstes eigenständig entscheiden, welche Herausgabe von Informationen im Sinne des Besitzers ist.

1.3 Realisierung eines Privacy-Frameworks

Die Problemstellung des Schutzes der Privatsphäre im Ubiquitous Computing ist allen ubiquitären Anwendungen und Geräten gemein. Der Benutzer muß die Sicherheit haben, daß seine Privatsphäre unangetastet bleibt bzw. nur die Teile offenbart, die er kommunizieren will. Zu den Grundlagen des Schutzes der Privatsphäre zählt die Verschlüsselung einer Übertragung; alle klassischen Ansätze benötigen hierfür jedoch eine eindeutige Identität, will man auch aktive Angreifer ausschließen – dies steht jedoch im Widerspruch zum grundlegenden Wunsch nach Anonymität.

Ein weiteres Problem ist die Kontrolle der von den Anwendungen benutzten Daten: Die Anonymität im Netzwerk und die Verschlüsselung der Daten kann auf gegebene Protokolle mit einer wohlbekannten Struktur aufsetzen. Anwendungsprotokolle hingegen sind individuell gestaltet und erlauben so zunächst keinen generischen Ansatz zum Schutz der personenbezogenen Daten.

Zuletzt sei nochmals auf die Problematik der Interaktion mit dem Benutzer hingewiesen: Die Idee der Durchdringung des Alltags widerspricht einer ständigen Kommunikation mit dem Benutzer. Sicherheitsrückfragen, wie sie von vielen Sicherheitslösungen bei Desktop-Computern bekannt sind, sollten so selten wie möglich geschehen.

Diese Arbeit stellt Bausteine für ein Privacy-Framework vor, welches diese Probleme adressiert. Dabei werden folgende Grundsätze verfolgt:

- Der Punkt für die Entscheidung über die Preisgabe von Daten (und damit auch die Stelle, an der möglicherweise Rückfragen an den Benutzer nötig werden), soll so weit wie möglich nach oben, in die Anwendungsebene geschoben werden. Erst hier sind sämtliche Informationen vorhanden, um eine solche Entscheidung treffen zu können.
- Prinzipiell sollen die Geräte in der Lage sein, soweit irgend möglich anonym zu agieren. Die Preisgabe der eigenen Identität oder anderer Informationen soll immer optional sein.
- Das Framework soll weder eine zugrundeliegende Infrastruktur noch vertrauenswürdige Instanzen zwingend erfordern. Ein System, das mit einer existierenden Infrastruktur in seiner Umgebung funktioniert, würde unserer Meinung nach keine Akzeptanz erfahren.

Die Arbeit orientiert sich hierbei am Aufbau der klassischen Netzwerk-Schichtenmodelle. Dies ermöglicht eine einfache Übertragung der vorgestellten Konzepte auch auf andere Anwendungsbereiche. Die vorgestellten Konzepte adressieren ausschließlich Probleme, die durch geeignete Algorithmen (und damit bei der Software-Erstellung) zu lösen sind; Privacy-Probleme, die auf der Hardware-Ebene zu suchen sind (wie charakteristische Signaturen, die durch Fertigungsschwankungen in den verwendeten Bauteilen entstehen können), werden zwar kurz angesprochen, aber nicht näher diskutiert.

Ziel der Arbeit ist es, ein Gerüst für privacy-bezogene Entscheidungen beim Ablauf einer ubiquitären Anwendung zu bieten. Eine solche Entscheidungskomponente, wie sie wohl im Forschungskontext der künstlichen Intelligenz und des automatischen Schließens entstehen könnte, bleibt außerhalb des Fokusses dieser Arbeit.

1.4 Aufbau der Arbeit

Die Gliederung dieser Arbeit orientiert sich an den Schichten klassischer Netzwerkmodelle.

In Kapitel 2 bieten wir[2] einen Überblick über die Entwicklung des Begriffs der Privatsphäre. Wir stellen verschiedene Systematisierungen und Möglichkeiten des Schutzes vor. Des weiteren liefern wir eine Argumentation, wieso trotz sozialer und gesetzlicher Regelungen ein technisches Schutzkonzept, wie es in dieser Arbeit vorgestellt wird, sinnvoll und nötig ist.

Kapitel 3 stellt zunächst die klassischen Netzwerkschichtenmodelle vor; wir geben Definitionen für verschiedene Begriffe von Anonymität und erläutern, welche personenbezogenen oder identifizierenden Merkmale auf jeder Schicht anfallen.

Im folgenden Kapitel 4 stellen wir eine Möglichkeit vor, wie mobile Knoten ohne Benutzung einer Infrastruktur ein weitgehendes Maß an Anonymität erlangen können. Dieser Ansatz liefert uns die (in Kapitel 6) benötigte Anonymität auf den Protokollschichten unterhalb der Anwendungsschicht; er wird mit Hilfe von komplexen Simulationen untersucht und belegt.

Das Kapitel 5 diskutiert den grundlegenden Zwiespalt zwischen einem sicheren Schlüsselaustausch und der Forderung nach Anonymität. Wir zeigen eine Möglichkeit auf, wie eine Authentisierung, welche die Preisgabe der eigenen Identität bedeutet, erst verzögert und bei Bedarf vonstatten gehen kann. Des weiteren diskutieren wir die in der gegenwärtigen Forschung noch unbetrachtete Problemstellung der Wiedererkennung anonymer Knoten: Wir erläutern dieses neue Problem und

[2] Für die Darstellung der Forschungsergebnisse wurde die „wir“-Form gewählt, da sie angenehmer als Formulierungen im Passiv zu lesen ist; diese Arbeit wurde dennoch ausschließlich vom genannten Autor verfaßt.

stellen ein effizientes Protokoll zur Lösung dieses Problems sowohl für die direkte als auch für Multicast-Kommunikation vor.

Im 6. Kapitel widmen wir uns schließlich der Anwendungsebene: Wir stellen am Beispiel einer einfachen Skriptsprache ein Verfahren vor, mit dem der Informationsfluß beim Ablauf einer Anwendung verfolgt werden kann. Die Parameter der Diensteaufrufe (welche einem RPC[3] entsprechen) werden hier semantisch gebunden, so daß Falschaussagen über Privacy-Garantien nicht möglich sind. Diese Diensteaufrufe sind auch die Schnittstelle, an welcher ein Privacy-Monitor den Informationsfluß kontrollieren und gegebenenfalls die Anwendung abbrechen kann.

Die Arbeit schließt mit einem Fazit in Kapitel 7.

[3] RPC: Remote Procedure Call. Aufruf einer Funktion auf einem anderen System via Netzwerk.

2 Privatsphäre

> You have zero privacy anyway—get over it.
>
> *Scott McNealy, 1999 [210]*

Die Begriffe Privatsphäre, Privatheit und „Privacy“ unterlagen im Laufe der Zeit einem stetigen Wandel. Ihre Bedeutung, Auslegung und tatsächliche Umsetzung änderte sich je nach geschichtlichen, sozialen und technischen Umständen. Dieses Kapitel soll eine Übersicht über die Entwicklung und mögliche Kategorisierungen geben. Ein besonderes Augenmerk gilt natürlich dem Bereich des Ubiquitous Computing: Die Bedeutung der Privatsphäre und die Möglichkeiten zu ihrem Schutz sollen betrachtet werden.

2.1 Gliederung und Entwicklung

Bei der Betrachtung der geschichtlichen Entwicklung westlicher Kulturen stellt man fest, daß sich die Privatsphäre zumindest als moralischer Wert in nahezu jedem Zeitabschnitt wiederfindet. Um das heutige Verständnis und die Verflechtung mit weiteren Werten zu beleuchten, ist ein Überblick über die Entwicklung der Begriffe und deren Festschreibung in Gesetzen von Vorteil.

2.1.1 Geschichtliche Entwicklung

Der Rechtsexperte Lawrence Lessig schreibt in einem seiner Aufsätze [154]: „Practical privacy is shaped by four strongly interacting forces: markets, social norms, legislation, and technology.“ Sowohl der Begriff der Privatheit als auch der tatsächliche Umgang mit der Privatsphäre wird von sozialen Normen, der Gesetzgebung, dem Markt und technologischen Entwicklungen beeinflußt. Dabei stellen Markt und Technologie Einflußfaktoren dar, welche Gesetzgebung und soziale Normen beeinflussen bzw. Anpassungen an diesen Regeln hervorrufen. Einen geschichtlichen Überblick über die Stellung und Wertschätzung der Privatsphäre kann man also erhalten, wenn man entsprechende Passagen in Gesetzestexten, Urteilen untersucht.

Die frühen, schriftlich niedergelegten Gesetzeswerke waren entweder Sammlungen von Gewohnheitsrechten (wie z. B. der Sachsenspiegel [236]) oder Zusammenstellungen von Präzedenzfällen. Hier fanden allgemein anerkannte gesellschaftliche Richtlinien Einzug. Somit stellen solche Gesetzeswerke ein Abbild der anerkannten ge-

sellschaftlichen Normen und Werte dar. Sucht man nach Gesetzen, welche die Privatsphäre betreffen, so findet man bereits in einem Englischen Gesetzeswerk aus dem Spätmittelalter (1361) den „Justices of the Peace Act“: Dieses Gesetz droht mit Haft für Lauscher und Voyeure [158, Seite 15].

Während der Debatte um ein Steuergesetz beschrieb der Engländer William Pitt (1708 – 1778) im Jahr 1763 das unantastbare Recht auf die Privatsphäre des eigenen Haushalts mit geradezu poetischen Worten [168]:

> „The poorest man may in his cottage bid defiance to all the force of the Crown. It may be frail; its roof may shake; the wind may blow through it; the storms may enter, the rain may enter, but the King of England cannot enter; all his forces dare not cross the threshold of the ruined tenement!“

Der technische Fortschritt barg auch in der Geschichte häufig neue Probleme und Fragestellungen bezüglich der Gesellschaftsform und deren Werte. So kam es 1890 in Amerika zu einer Klage, da in der Regenbogenpresse Bilder von Personen ohne deren Einverständnis abgedruckt wurden. Die neue Technik (Fotografie, Zeitungsdruck) erforderte eine neue Evaluation der hergebrachten Werte bezüglich der Privatsphäre der betroffenen Personen. Der Richter urteilte damals, daß jedermann das Recht auf seine Privatsphäre habe; und dieses Recht definierte er als den Anspruch, in Ruhe gelassen zu werden [231]:

> Privacy is „the right to be let alone“.

Bei der Deklaration der Menschenrechte 1948 wurde das Anrecht auf den Schutz der Privatsphäre recht umfassend festgelegt [5, Artikel 12]:

> „No-one should be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks on his honour or reputation. Everyone has the right to the protection of the law against such interferences or attacks.“

Der fortschreitenden technischen Entwicklung, insbesondere bei der elektronischen Datenverarbeitung, folgten entsprechende Gesetze und Urteile, um diese in geregelten Bahnen zu halten. Entsprechend entstanden weitere Beschreibungen und Definitionen des Privacy-Begriffs. Aus dem Jahr 1967 stammt eine Definition, welche Privacy eng an den Kommunikationsbegriff bindet [238]:

> Privacy is „the claim of individuals (...) to determine for themselves when, how and to what extent information is communicated to others“.

Privatheit sei also der Anspruch darauf, daß jedes Individuum für sich entscheiden kann, wann, zu welchem Umfang und auf welche Weise eine Information anderen kommuniziert wird.

Auf der juristischen Seite nimmt Deutschland auf dem Gebiet der Privatsphäre eine Vorreiterrolle ein: So verabschiedete Hessen 1970 das weltweit erste Datenschutzge-

setz. 1983 folgte das sogenannte „Volkszählungsurteil“, das vom Bundesverfassungsgericht in Folge auf Klagen gegen die geplante Volkszählung gefällt wurde. Dieses Urteil prägt den Begriff der „informationellen Selbstbestimmung“: Jeder Bürger besitzt das Recht, selbst darüber zu verfügen, wer welche Daten über ihn besitzt. Damit folgt das Gericht der oben zitierten Definition von Westin [238], geht allerdings noch einen Schritt weiter: Die personenbezogenen Daten gehören immer der Person, auf die sie sich beziehen, ganz egal, von wem sie erhoben wurden. Dies ist ein starker Unterschied zu anderen Staaten (wie beispielsweise den USA), wo Daten derjenige besitzt (und damit auch verwerten oder verkaufen darf), der sie gespeichert hält. Dem Urteil nach hat das Recht auf informationelle Selbstbestimmung Verfassungsrang, da es sich direkt aus den Artikeln 1 und 2 des Grundgesetzes ableitet.

Weitere Regelungen zum Umgang mit personenbezogenen Daten brachte das Bundesdatenschutzgesetz von 1990. Ziel des Gesetzes ist es, „den Einzelnen davor zu schützen, daß er durch den Umgang mit seinen personenbezogenen Daten in seinem Persönlichkeitsbereich beeinträchtigt wird“. Grundlage für den Umgang mit Daten, welche die Privatsphäre von Personen betreffen, sind die Prinzipien der Datensparsamkeit, des Erlaubnisvorbehalts und der Erforderlichkeit. Das Erforderlichkeitsprinzip besagt, daß nur Daten gespeichert werden dürfen, die für den Grund der Speicherung zwingend erforderlich sind. Die Datensparsamkeit schreibt vor, daß die Datensätze so schlank wie möglich zu halten sind. Der Erlaubnisvorbehalt besagt schließlich, daß Daten nur mit Einwilligung des Betroffenen gespeichert werden dürfen; außerdem hat dieser jederzeit das Recht, die gespeicherten Daten einzusehen oder deren Löschung zu veranlassen.

Moderne Definitionen von Privacy findet man natürlich auch außerhalb der Gesetzgebung; im „Cypherpunk's Manifesto“ [121], einem Aufsatz aus der Hacker- und Crypto-Szene, findet man eine Definition von Privacy, die den juristischen Definitionen sehr ähnelt:

> „Privacy is the power to selectively reveal oneself to the world“

Die Ähnlichkeit belegt die Annahme vom Anfang – Gesellschaft und (zumindest die deutsche) Gesetzesgebung scheinen auch heute noch nicht allzu stark zu divergieren.

2.1.2 Gliederung der Privatsphäre

Bei eingehender Diskussion des Begriffs „Privatsphäre“ stellt man fest, daß sich dieser in weitere, nicht zwingend voneinander abhängige Teilbereiche gliedern läßt. Eine erste solche Gliederung findet man bei Clarke [60]; das Electronic Privacy Information Center (EPIC, [83, Overview]) und weitere Autoren [151, 187] gelangen zu ähnlichen Gliederungen:

Körperliche Privatheit (bodily privacy): Sie bezeichnet den Schutz des eigenen Körpers vor invasiven Tests wie Drogen-, Gentests oder Leibesvisitationen.

Territorielle Privatheit (territorial privacy): Der Schutz der Privatsphäre innerhalb eines persönlichen Gebiets, sei es die eigene Wohnung oder der Arbeitsplatz, wird als territorielle Privatheit bezeichnet. Dies beinhaltet den Schutz vor Eingriffen wie Durchsuchungen oder Videoüberwachung. Im öffentlichen Raum versteht man unter territorieller Privatheit den Schutz vor Videoüberwachung, Ausweis- oder anderen Identitätsüberprüfungen, sowie der freien Bewegung ohne Verfolgung oder dem Erstellen von Bewegungsprofilen.

Privatheit der Kommunikation (privacy of communications): Diese Anforderung garantiert die Vertraulichkeit von Telefongesprächen, E-Mails, das Briefgeheimnis, etc.

Informationelle Privatheit (information privacy): Sie wird gemeinhin als Datenschutz bezeichnet. Sie umfaßt Regeln bezüglich der Handhabung, Speicherung und Weitergabe personenbezogener Daten. Letztere reichen von Logeinträgen über Kreditkartentransaktionen bis hin zu Personalakten und medizinischen Aufzeichnungen.

Die Philosophin Beate Rössler faßt die erwähnten Aspekte anders zusammen; der Schutz der Privatsphäre bedeutet in ihren Arbeiten Zugriffsschutz bzw. Zugriffskontrolle. Sie differenziert die Privatsphäre in drei Arten von Privatheit [188, 189]:

Informationelle Privatheit: Die informationelle Privatheit bezieht sich auf Daten über die eigene Person, also generell Fakten, die andere über eine Person wissen.

Dezisionale Privatheit: Sie betrifft das freie Treffen von privaten Entscheidungen und Handlungen(z. B. die Berufswahl, die eigene Religion, Wahl der Kleidung, etc.): Dezisionale Privatheit ist gewährleistet, wenn man in seinen eigenen Handlungen frei und ohne Rechtfertigungszwang ist.

Lokale Privatheit: Die lokale Privatheit entspricht der obigen Definition territorieller Privatheit: Sie umfaßt den Schutz von persönlichen Gebieten vor (physikalischen) Übertritten Fremder oder vor Überwachung oder Abhörmaßnahmen.

2.1.3 Kontrollmechanismen

Möchte man bestimmte Regeln etablieren, benötigt man Mittel, um diese zu kontrollieren. Regelungen zum Schutz der Privatsphäre lassen sich auf verschiedene Weisen realisieren:

Gesetzliche Mittel

Die Privatsphäre kann von staatlicher Seite durch Gesetze und Vorschriften geschützt werden. Die Einhaltung solcher Gesetze kann *präventiv*, beispielsweise durch routinemäßige oder stichprobenartige Kontrollen, geschehen. Ein Beispiel für Regelungen, die auf diese Art überprüft werden, sind die Lebensmittelvorschriften: Die Einhaltung der Hygienevorschriften wird stichprobenartig durch unangekündigte Besuche des LKD[1] überprüft.

Alternativ kann die Kontrolle *reaktiv* geschehen: Vermutet jemand ein Vergehen, so erstattet er Anzeige. Daraufhin erfolgt eine entsprechende Untersuchung des Sachverhalts.

Soziale Kontrolle

Neben den „harten Kontrollmechanismen“ durch den Gesetzesgeber kann der Schutz der Privatsphäre auch durch soziale Regelungen erfolgen. Dies geschieht implizit durch die allgemein anerkannten ethischen Standards einer Gesellschaft – es gibt Dinge, die „tut man einfach nicht“, weil es sich nicht gehört. Für solche etablierten Verhaltensstandards, implementiert durch die elterliche Erziehung und das soziale Umfeld, benötigt man nicht zwingend eine gesetzliche Reglementierung.

Ebenfalls in die Kategorie der sozialen Kontrolle fallen die Mechanismen der Selbstregulierung und der Selbstverpflichtung. Firmen, Konsortien oder ähnliche Gruppierungen geben sich hierbei ein Regelwerk und erlegen sich selbst die Pflicht auf, sich daran zu halten. Grund hierfür kann eine vertrauensbildende Maßnahme gegenüber Kunden sein – oder aber der Wunsch, auf diese Weise einer gesetzlichen Reglementierung zu entgehen. Ein Beispiel für eine solche Selbstregulierung ist die Alterseinstufung von Kinofilmen durch die FSK[2].

Technische Mittel

Zu guter Letzt kann eine Kontrolle der Privatsphäre durch technische Maßnahmen erfolgen. Das Gerät wird mit Richtlinien und Zugriffsregeln ausgestattet, welche die Präferenzen des Benutzers widerspiegeln. Entsprechende Software (oder Hardware) sorgt bei technischen Mitteln dafür, daß bestimmte Regeln eingehalten werden. Im Gegensatz zu den vorigen beiden Mitteln sind die technischen Mittel Individuumszentriert: Die Technischen Mittel kann der Betroffene selbst anwenden und durchsetzen, während die vorhergehenden Mittel einer entsprechenden Umgebung in Form von geeigneter Gesetzeslage bzw. gesellschaftlichen Normen bedürfen.

[1] LKD: Lebensmittelkontrolldienst

[2] FSK: Freiwillige Selbstkontrolle. Kontrollgremium der Vertreter der Filmindustrie in Deutschland, welche die Altersfreigabe neuer Kinofilme bestimmt.

Ein Beispiel für eine technische Kontrolle aus einem anderen Bereich sind Navigationsgeräte, die bei Überschreiten der verzeichneten Höchstgeschwindigkeit den Benutzer warnen. Ein weiteres Beispiel sind Webbrowser, welche P3P[3]-Informationen von Webseiten [65] auswerten, gegen die Benutzereinstellungen abgleichen und gegebenenfalls den Zugriff auf die Seite verwehren.

2.1.4 Aktuelle Tendenzen und Konsequenzen

> Both sides of the calendar debate were wrong; the new century began on 11 September 2001.
>
> *Bruce Schneier, 2001 [197]*

Wie in Kapitel 2.1.1 auf Seite 7 gezeigt, wird die Privatsphäre als wünschenswertes und schützenswertes Gut anerkannt. In einer amerikanischen Studie aus dem Jahr 2000 [180] wünschten sich 86% der Befragten, vor der Verwendung ihrer personenbezogenen Daten gefragt zu werden. Weiterhin waren 84% darüber besorgt, daß Firmen und Personen Daten ohne ihr Wissen über sie und ihre Familien sammeln. Dieses Umfrageergebnis stellte damit eine starke Kritik an der amerikanischen Datenschutzpraxis dar – wie oben beschrieben, gehören in den USA Daten demjenigen, der sie gespeichert hat (und nicht dem, auf den sie sich beziehen).

Weitere Erhebungen zeigen, daß es einem Großteil der Bevölkerung zunehmend schwerer fällt, die Privatsphäre wahrzunehmen; durch Techniken wie Kundenkarten, Browser-Cookies und Web-Bugs kommt es zu einem schleichenden, nicht spürbarem Verlust von Teilen der eigenen Privatsphäre. So wird bei Umfragen [67, 224] die Privatsphäre zwar als wichtiges und schützenswertes Gut angegeben, jedoch differiert das praktische Verhalten von diesen Wünschen: Laut [227] hatten 2003 mehr als 52% der deutschen eine Kundenkarte, trotzdem gaben wenige Jahre zuvor in [106] mehr als 87% an, daß ihnen ihre Privatsphäre wichtig sei. Beate Rössler bezeichnet dieses Phänomen als *kognitive Asymmetrie* [189].

Mit den Anschlägen vom 11. September 2001 änderte sich die Weltpolitik dramatisch. Viele Regierungen verschärften Sicherheits- und Überwachungsgesetze, was von der Bevölkerung zunächst als sinnvoll hingenommen wurde. Inzwischen mehren sich jedoch wieder die Stimmen, welche die Summe an neuer Überwachung kritisieren ([189] argumentiert sogar, daß eine liberaldemokratische Gesellschaft von der Autonomie ihrer Bürger abhängt, und diese nicht ohne den Schutz von Privatheit lebbar ist); aber auch an der Tatsache, daß sich große Firmen [47, 48] mit der Thematik auseinandersetzen, ist als Zeichen dafür zu werten, daß der Schutz der Privatsphäre nach wie vor ein ernstzunehmendes Thema darstellt.

Daß das Respektieren der Privatsphäre einen positiven wirtschaftlichen Faktor darstellen und zu effizienteren Prozessen führen kann, zeigt Laudon [150]. Wenn bei-

[3] P3P: Platform for Privacy Preferences. Ein maschinenlesbares Format, mit dem Webseiten ihre Privacy-Policy beschreiben können. Beschreibung in Kapitel 6.2.3 auf Seite 93.

spielsweise die Verletzung der Privatsphäre durch unerwünschte Anrufe wegfallen würde, würde dies dazu führen, daß Telefonate zuverlässiger beantwortet würden. Analog würde ein Abebben der Flut an Spam-Mails das Medium E-Mail wieder zu einem effizienten Kommunikationsmedium machen. Auch der Vertrauensbonus von Kunden gegenüber Diensteanbietern sei nicht zu unterschätzen.

2.2 Herausforderungen des Ubiquitous Computings

Einige Eigenschaften der Vision des Ubiquitous Computings haben direkte Auswirkungen auf die Privatsphäre.

Durchdringung der Umgebung: In einer Welt, in der Ubiquitous Computing weite Verbreitung gefunden hat, befindet man sich ständig in der Gegenwart von miniaturisierten Computern, die über drahtlose Schnittstellen mit Geräten, die man mit sich trägt, kommunizieren (oder dies zumindest potentiell können). Die Gewöhnung an diesen Zustand läßt mögliche Gefahren für die Privatsphäre vergessen. Umgekehrt bieten sich für Datensammler vielfältigste Möglichkeiten, Personen in einer solchen Umgebung zu überwachen.

Unsichtbarkeit der Geräte: Die Idee des Ubiquitous Computing sieht nicht nur vor, daß die Geräte allgegenwärtig sind, sondern daß sie für den Benutzer faktisch unsichtbar sind. Dies verstärkt den Effekt der kognitiven Asymmetrie: Es besteht die Gefahr des Verlusts von Teilen der Privatsphäre, da man weder die Geräte in der Umgebung noch die Kommunikation mit ihnen wahrnehmen kann.

Automatisierung: Durch die Automatisierung von Abläufen zwischen den Geräten ist zwar ein hoher Komfortgrad erreichbar, jedoch tragen ganze Abläufe, die nicht bewußt vom Benutzer gesteuert werden, weiter zur kognitiven Asymmetrie bei.

Sensorik: Leistungsfähigere Sensoren in den Kleingeräten werden es erlauben, immer detailliertere Informationen über ihre Umgebung – und damit auch über die Menschen dort zu sammeln. Beispielsweise kann über ein Mikrofon die Zahl der Personen anhand unterschiedlicher Klangcharakteristik der Stimme bestimmt werden; abgesehen von einer Spracherkennung verrät der Tonfall viel über die Stimmung der Person.

Datenspeicherung: Die Vielzahl der Kleingeräte und der Preisverfall von Speicherbausteinen und -medien wird dafür sorgen, daß das Vorhalten von umfassenden Daten über längere Zeiträume hinweg möglich ist. Die Zusammenführung von Datenspeichern mehrerer Sensoren und Kleingeräte wird eine sehr große Datenfülle ergeben, welche die Möglichkeit bietet, sehr detaillierte Benutzerprofile zu bilden.

Das Problem der Privatsphäre wurde bereits von Mark Weiser in seinen ersten Veröffentlichungen über Ubiquitous Computing angesprochen [234, 235]. Eine detailliertere Aufstellung mit Richtlinien, die für ein Privacy-freundliches Ubiquitous Computing einzuhalten sind, stellt Langheinrich [147] auf:

- Hinweise für den Benutzer (notice): Der Benutzer muß auf die Gefahr (Sammlung/Auswertung von Daten) hingewiesen werden.
- Wahlmöglichkeit und Einverständnis (choice and consent): Die Verarbeitung darf nur mit Einverständnis des Benutzers geschehen.
- Anonymität und Pseudonymität: Die Daten müssen weitestgehend anonymisiert werden.
- Lokale Begrenzung, lokaler Bezug (proximity and locality): Daten dürfen nur innerhalb ihres lokalen Bezugs verwertet werden.
- Angepaßte Sicherheit (adequate security): Die Kommunikation muß gemäß ihrer Brisanz abgesichert werden.
- Regreßhaftung der Diensteanbieter (access and recourse): Geräte und Dienste, die sich nicht an Zusicherungen bezüglich der Privatsphäre halten, müssen ausgeschlossen (und gegebenenfalls rechtlich belangt) werden können.

2.3 Fazit

Zusammenfassend läßt sich feststellen, daß die Menschen ihre Privatsphäre nach wie vor als hohes Gut ansehen. Soweit von ihnen feststellbar, reagieren sie empfindlich auf Verletzung dieses Grundrechts – insbesondere den Deutschen wird hier eine Vorreiterrolle zugeschrieben.

Die Wahrnehmung solcher Verletzungen ist jedoch das grundlegende Problem: Neue Technologien nutzen immer komfortablere, aber damit auch immer weniger wahrnehmbare Wege der Kommunikation. Somit schwindet die Sensibilität der Menschen bezüglich ihrer Privatsphäre an diesen Stellen; Marketingstrategen nutzen dies aus, wie man an den paradoxen Umfrageergebnissen bezüglich Kundenkarten (Beispiel auf Seite 12) deutlich sehen kann.

Als erste Konsequenz hieraus verfolgen einige Arbeiten den Ansatz, die Privatsphäre wieder in den Fokus der Benutzer zu rücken: Im Web geschieht dies mit der P3P-Initiative [66, 65]; die Arbeit von Langheinrich [149], Kapitel 4 überträgt diesen Ansatz auf das Ubiquitous Computing, indem sogenannte Privacy Beacons entsprechende Metainformationen an die Umgebung senden.

Eine solche Herangehensweise birgt jedoch zwei Nachteile: Zum einen verliert man hierdurch einiges an Komfort, welchen das Ubiquitous Computing durch seine unaufdringliche Natur mit sich bringen sollte. Viel schwerwiegender ist jedoch, daß bei

so einem Ansatz der Betrug durch einen Diensteanbieter für diesen geradezu verlockend sein muß: Personenbezogene Daten können eine wahre Goldgrube sein, und die Datenspeicherung und -weitergabe ist für den Benutzer unspürbar und – falls sie jemals festgestellt werden sollte – nahezu unmöglich nachvollziehbar.

Ein konsequenter Ansatz sollte daher harte Verbindlichkeiten bezüglich personenbezogener Daten verfolgen; bei einer Trennung von Dienst und beschreibenden Metadaten ist dies nicht möglich: Gibt es keinen Mechanismus, welcher die Dienste an die korrespondierenden Metadaten mit einer Privacy-Beschreibung bindet, so ist es möglich, in den Metadaten beliebige falsche Angaben zu machen, genauso wie es möglich ist, Dienste oder Diensteteile komplett zu verschweigen. Wenn wir uns nochmals die geschilderten Kontrollmöglichkeiten vor Augen halten, stellen wir fest, daß sowohl gesetzliche als auch soziale Kontrolle nur sehr schwer durchzuführen ist, aus dem oben genannten Grund. Dazu kommt noch, daß ein Benutzer möglicherweise eigene, weitergehende Präferenzen hat – insbesondere falls man mit der Weitergabe von personenbezogenen Daten auch die theoretische juristische Handhabe (wie sie in Deutschland vom Datenschutzgesetz gewährleistet wird) verliert, wiegt die Übertragung doppelt schwer.

Daher müssen die ubiquitären Geräte, die personenbezogene Daten übertragen können, den Benutzer in dessen Handlungen aktiv unterstützen. Nur wenn ein solches Gerät die Präferenzen des Benutzers kennt, kann es aktiv die unerwünschte Weitergabe von Daten verhindern.

2.3.1 Verfolgter Ansatz

Um einen konsequenten Schutz der personenbezogenen Daten (wozu auch beispielsweise Bewegungsprofile gehören, welche implizit entstehen) zu gewährleisten, wird in dieser Arbeit der Aspekt der Privacy als erstes und mit Vorrang betrachtet. Die Übertragungssicherheit kann nur an zweiter Stelle folgen, da verschiedene kryptographische Verfahren eine eindeutige Identität voraussetzen oder implizieren. Die jeweiligen Anwendungen satteln zuletzt auf diesem Gerüst auf; durch Ablaufkontrolle kann der Informationsfluß kontrolliert und gegen die Privacyanforderungen des Benutzers abgeglichen werden.

2.3.2 Definition von „Privatsphäre“

In dieser Arbeit betrachten wir den Begriff Privacy aus der technischen Sicht:

Definition 2.1 (Privatsphäre, personenbezogene Daten) *Die* Privatsphäre *sei die Menge aller personenbezogenen Daten.* Personenbezogene Daten *sind Informationen, die eine Person auszeichnen, beschreiben oder welche die Person aufgrund von persönlichen Entscheidungen erlangt hat. Diese Definition beschreibt zunächst ohne weitere Wertung eine Menge von Daten; der eigentliche Schutz ge-*

schieht durch die Entscheidung des Besitzers, welche dieser Daten er wem gegenüber preis gibt.

Die Begriffe „Schutz der Privatsphäre“, „Privatheit“ und „Privacy“ werden synonym genutzt. Die oben erläuterten Gliederungen sollen helfen, Privacy-Probleme zu identifizieren; der Schutz soll aber gliederungsübergreifend und nicht nur auf einen Teilaspekt beschränkt sein.

3 Identifizierende Merkmale

> What's in a name? That which we call a rose by any other name would smell as sweet.
>
> *William Shakespeare (1564–1616), „Romeo and Juliet“*

In der Vision des Ubiquitous Computings geht man davon aus, daß jeder Mensch einige Kleingeräte mit sich herumträgt – möglicherweise sogar ständig und ohne dies bewußt wahrzunehmen. Durch in diesen Geräten gespeicherte personenbezogene Daten, sowie durch die alleinige Tatsache, daß das Gerät am Körper mit sich geführt wird, erstreckt sich die Privatsphäre dieser Menschen mit auf diese Geräte. Der Schutz dieser Informationen, die von diesen Klein- und Kleinstgeräten – im folgenden kurz als *Knoten* bezeichnet – ausgehen können, ist das Ziel dieser Arbeit.

Es ist der zugrundeliegende Ansatz dieser Arbeit, den mobilen Knoten weitestgehende Anonymität zu gewähren, welche diese dann auf Wunsch kontrolliert aufgeben können. Um das Ziel der Anonymität zu erreichen, müssen zunächst die Möglichkeiten zur Identifizierung betrachtet werden, um anschließend geeignete Gegenmaßnahmen treffen zu können. In diesem Kapitel betrachten wir verschiedene Definitionen von Anonymität und Identifizierung; außerdem identifizieren wir die Stellen, an denen eine Deanonymisierung möglich ist. An diesen Stellen müssen Maßnahmen zur Anonymisierung und Kontrolle getroffen werden, welche wir in den folgenden Kapiteln beschreiben.

Die verschiedenen Begriffe aus diesem Kontext werden in Publikationen häufig in leicht unterschiedlicher Bedeutung verwendet. Pfitzmann und Köhntopp geben in ihren Arbeiten [174, 175] eine Übersicht über viele der verschiedenen Begriffe und versuchen, diese mit einer möglichst genauen Definition zu versehen.

3.1 Identifizierung und Authentisierung

Im folgenden verwenden wir diese Begriffe nach dieser Definition:

Identifizierung: Mit Identifizierung bezeichnen wir das eindeutige Erkennen eines Netzteilnehmers. Die Identifizierung kann rein passiv (also auch durch einen Angreifer, der das Netzwerk nur abhört) erfolgen. Falls der Netzteilnehmer ursprünglich versuchte, anonym zu agieren, spricht man auch von *Deanonymisierung*.

Authentisierung: Mit Hilfe der Authentisierung beweist ein Knoten seine Identität (oder die Zugehörigkeit zu einer Rolle). Um Angriffe durch das Wiedereinspielen von mitgeschnittenen Netzwerkpaketen zu vermeiden, muß eine Authentisierung interaktiv erfolgen.

3.2 Anonymität und Pseudonymität

Unter *Anonymität* versteht man den Zustand, innerhalb einer Menge von Subjekten nicht identifizierbar zu sein (eine Identifizierung ist also nicht möglich). Diese Menge von Subjekten nennt man *Anonymitätsmenge (anonymity set)* [175]. Die erste naheliegende Folge ist, daß es keine Anonymität geben kann, wenn die Anonymitätsmenge eine Größe von eins hat.

Neben dieser allgemeinen Definition findet man häufig die folgenden Begriffe:

Faktische Anonymität: Diese Begriffsdefinition stammt aus dem Bundesstatistikgesetz [6]. Sie besagt, daß eine Person faktische Anonymität (innerhalb einer Datenmenge) genießt, falls ein unverhältnismäßig großer Aufwand an Zeit, Kosten und Arbeitskraft erforderlich ist, um den Personenbezug wieder herzustellen.

Insbesondere der Bezug auf die benötigte Zeit bedeutet, daß sich die faktische Anonymität von gespeicherten Daten im Laufe der Zeit ändern kann: Durch Verbesserung von Datamining-Techniken, zusätzlich gewonnen und fusionierten Daten und nicht zuletzt wegen des Preisverfalls von Rechenzeit ist es möglich, daß ab einem bestimmten Punkt die faktische Anonymität verloren geht. Eine ausführliche Betrachtung dieser Problematik findet man in [239].

k-Anonymität: Die Definition von k-Anonymität besagt, daß ein Individuum von weiteren $k-1$ Individuen bezüglich einer Menge von Kriterien (dem sogenannten Quasi-Identifier) nicht unterscheidbar ist [222, 221].
Innerhalb einer Tabelle T mit Feldern $F_1, \ldots, F_n$ und einem Quasi-Identifier $QI \subseteq F_1, \ldots, F_n$ liefert jede Anfrage entweder 0 oder mindestens k Ergebnisse.

Pseudonymität bezeichnet die Möglichkeit, unter einem Pseudonym als identifizierendes Merkmal zu agieren [175]. Ein *Pseudonym* ist hierbei ein eindeutiger Bezeichner, der aber nur für eine bestimmte Zeit oder einen bestimmten Zweck verwendet wird. Im Gegensatz zur Anonymität können alle Aktionen, die unter demselben Pseudonym geschehen, zusammengeführt werden. Je nach Anwendung können Pseudonyme in verschiedener Weise eingesetzt werden:

Personenpseudonym: Ein Personenpseudonym ist ein Bezeichner, der vom Nutzer über lange Zeit unabhängig vom Einsatzzweck verwendet wird. Beispiele hierfür sind E-Mail-Adressen oder Künstlernamen.

Rollenpseudonym: Wird für eine Rolle, die ein Nutzer einnimmt, immer dasselbe Pseudonym benutzt, so nennt man dies ein Rollenpseudonym. Rollen können beispielsweise die verschiedenen Hobbies eines Internet-Nutzers sein: Er meldet sich in verschiedenen Foren an, wobei er bei Foren, die sich um dasselbe Thema drehen, immer den selben Nickname verwendet.

Beziehungspseudonym: Für jeden Kommunikationspartner wird hier ein separates Pseudonym benutzt.

Transaktionpseudonym: Für jede Transaktion wird ein separates Pseudonym verwendet. Eine solche Transaktionen kann beispielsweise eine einzelne Online-Bestellung sein. Ein Cookie, der von einem Webshop verwendet und am Ende des Shopbesuchs gelöscht wird, wäre ein weiteres Beispiel für ein Transaktionspseudonym.

Im Kontext der Privatsphäre sind Pseudonyme von besonderem Interesse, wenn sie *unverknüpfbar (unlinkable)* sind. Die mathematische Definition erfolgt analog zu Shannons Definition für perfekte Chiffren [208]: Von *Unverknüpfbarkeit (Unlinkability)* spricht man, wenn aus Sicht eines Angreifers die Wahrscheinlichkeit für einen Zusammenhang zweier Pseudonyme p_1, p_2 genauso groß wie die Wahrscheinlichkeit für das Auftreten eines einzelnen der beiden ist: $P(p_2|p_1 \text{ zuvor beobachtet}) = P(p_2)$.

3.3 Schichtenmodelle

Wie an den oben gegebenen Beispielen für Pseudonyme zu erkennen ist, kann eine Identifizierung an den verschiedensten Stellen erfolgen. Für ein strukturiertes Bearbeiten der in dieser Arbeit gestellten Aufgabe bietet es sich daher an, diese Stellen zu kategorisieren. Hierfür bieten sich die Schichtenmodelle der Netzwerkstacks an; die beiden wichtigsten Vertreter sollen in diesem Abschnitt kurz vorgestellt werden, bevor im Rest des Kapitels die Möglichkeiten zur Identifizierung auf jeder Schicht besprochen werden.

Schon früh in der Geschichte der Computernetzwerke zeigte sich, daß eine Systematisierung der Netzwerkkommunikation hilfreich ist. Zum einen ließ sich so die komplexe Problematik in kleinere, besser zu bewältigende Probleme zerlegen; zum anderen ermöglichten wohldefinierte Schnittstellen den Austausch einzelner Teilprotokolle gegen andere Komponenten [223].

Die beiden wichtigsten Schichtenmodelle sind wohl das ISO-OSI Schichtenmodell und das Internet Schichtenmodell [63]: Das OSI-Modell stellt eine sehr allgemeine und feingranulare Gliederung dar; das Internet-Schichtenmodell ist eine vereinfachte Darstellung, welche die Gliederung der TCP/IP-Protokollsuite widerspiegelt.

Das OSI Schichtenmodell gliedert die Netzwerkkommunikation in die folgenden sieben Schichten [70]:

- Physical Layer: Beschreibt die physikalische Repräsentation von binären Daten. Hierzu gehören Eigenschaften wie Bitrate, Spannungspotentiale für 0 und 1, etc.
- Data Link Layer: Diese Schicht implementiert Fehlererkennung für die Kommunikation zwischen zwei Rechnern via dem Physical Layer. Framing und Medienzugriff fallen ebenfalls in den Zuständigkeitsbereich dieser Schicht.
- Network Layer: Die Netzwerkschicht sorgt für die Übertragung von Datenpaketen zwischen Rechnern, die sich nicht im selben lokalen Netz
- Transport Layer: Implementiert die Übertragung von Nachrichten zwischen zwei Kommunikationsendpunkten (Ports). Je nach Ausführung kann diese Schicht verbindungsorientiert oder verbindungslos agieren.
- Session Layer: Dient dem Aufbau der Kommunikation zwischen zwei Anwendungen; im Falle eines Abbruchs einer vorhergehenden Verbindung ist es Aufgabe des Session layers, die vorherige Sitzung wieder aufzunehmen.
- Presentation Layer: Die Darstellungsschicht sorgt für eine systemunabhängige Repräsentation der übertragenen Daten.
- Application Layer: Die Protokolle der Anwendungen werden auf dieser Schicht behandelt.

Das Internet-Schichtenmodell vereinfacht die Sicht der Dinge etwas [178], es konzentriert sich in erster Linie auf die Software-Schnittstellen:

- Hardwareschicht (Local Network Layer): Diese Schicht entspricht grob dem Data link layer des OSI-Modells. Sie kümmert sich um Versand und Empfang von Datenpaketen über die lokale Netzinfrastruktur.
- Routingschicht (Internetwork Layer): Der Internetwork Layer ermöglicht die Adressierung und den Versand von Daten über Netzwerkgrenzen hinaus. Das IP-Protokoll (sowie das Signalisierungsprotokoll ICMP) sind hier angesiedelt.
- Transportschicht (Transport Layer): Stellt die Verbindung zwischen den Kommunikationsendpunkten der Anwendungen her. Im normalen IP-Protokoll-Stack sind hier die Protokolle UDP und TCP angesiedelt. Je nach Protokoll (z. B. bei TCP) wird auf dieser Schicht die Zustellgarantie implementiert, die Reihenfolge der Pakete beim Empfänger wiederhergestellt oder vom paketbasierten Ansatz zu Datenströmen abstrahiert.
- Anwendungsschicht (Application Layer): Auf dieser Schicht sind wiederum die Protokolle der Anwendungen angesiedelt. Protokolle wie SSL/TLS [75] nehmen in sofern eine Sonderrolle ein, da sie genaugenommen zwischen dieser und der vorigen Schicht liegen (im ISO-OSI-Modell würden sie dem Session Layer zugeordnet werden).

Wegen der weiten Verbreitung der Internet-Protokollsuite orientieren wir uns im folgenden an diesem Modell. Die folgenden Abschnitte zeigen (sowohl allgemein als auch am Beispiel von TCP/IP), welche identifizierenden Merkmale auftreten. Weiterhin nutzen wir diese Kategorisierung, um den Betrachtungsrahmen dieser Arbeit einzugrenzen.

3.4 Hardwareschicht

Wie bereits erwähnt, erfolgt hier der Medienzugriff; um Datenpakete an lokale Rechner zustellen zu können, benötigt jeder Rechner eine (zumindest lokal) eindeutige Adresse – die MAC-Adresse[1].

Doch auch die verwendete Hardware an sich kann identifizierende Charakteristika besitzen [102]: Durch Toleranzen in der Hardware unterscheiden sich die erzeugten Signale; in [69] wird beispielsweise die Präambel für den Medienzugriff (welche immer dieselbe Bitsequenz repräsentiert) genutzt, um Netzwerkkarten zu unterscheiden. Dieses Problem kann durch hochwertige Bauteile mit geringer Fertigungstoleranz vermieden werden; es wird im Rest dieser Arbeit nicht näher betrachtet.

3.5 Routingschicht

Die Routingschicht dient der Vernetzung über die lokalen Netzgrenzen hinaus; dementsprechend benötigt das Protokoll dieser Schicht eine weltweit eindeutige Adresse.

Dies ist in der Internet-Protokollsuite die IP-Adresse; allerdings gibt es bei IP noch weitere Daten, die zu einer Identifizierung führen können: IP erlaubt das Fragmentieren und Reassemblieren von Paketen; um die Fragmente eines Pakets wieder zusammensetzen zu können, erhält jedes Paket eine Sequenznummer. Mit Hilfe dieser Sequenznummer können Rechner unterschieden werden, selbst wenn die sendende IP-Adresse verändert wurde [21]. Weitere charakteristische Verhaltensweisen der Implementierung des IP-Stacks erlauben Schlüsse auf das verwendete Betriebssystem; solche Merkmale lassen sich jedoch vermeiden bzw. entfernen [232].

3.6 Transportschicht

Die Transportschicht führt für jeden Netzteilnehmer mehrere Kommunikationsendpunkte ein; bei TCP/IP sind dies die Ports. Unter Umständen lassen diese gewisse

[1] MAC: Medium Access Control. Eindeutige Nummer, mit welcher ein Gerät im lokalen Netz adressiert werden kann.

Rückschlüsse auf die Verbindung zu (bei TCP werden beispielsweise Portnummern größer 32768 für ausgehende Verbindungen verwendet).

Wenn Protokolle (wie TCP) die Reihenfolge der Pakete sicherstellen, verwenden sie hierzu Sequenznummern; manche Systeme verwenden als neuen Sequenzstart eine vorhersehbare Sequenznummer, was eine Identifizierung dieser Rechner erlauben würde. Da eine solche Vorhersagbarkeit auch Angriffe auf die Kommunikation (TCP Hijacking [71]) erlaubt, sorgen inzwischen fast alle Betriebssysteme dafür, daß die Sequenznummern nicht mehr erraten werden können – dies entspricht der oben erläuterten Forderung nach Unverknüpfbarkeit.

3.7 Anwendungsschicht

Die eigentlichen Anwendungen setzen auf den darunterliegenden Schichten auf und nutzen diese als abstraktes Kommunikationsmedium. Aus Sicht dieser Systematik besteht der entscheidende Unterschied der Anwendungsschicht darin, daß die Anwendungsdaten opak sind: Waren die Daten der vorangehenden Schichten nach einem bekannten Schema strukturiert, gibt es für die Daten der Anwendungsschicht keine allgemeingültige Struktur.

Gerade auf der Anwendungsschicht werden die meisten personenbezogenen und privacy-sensiblen Daten übermittelt. Ist das verwendete Anwendungsprotokoll bekannt, so ist durch das Sammeln dieser Daten eine immer feinere Clusterung bis hin zur eindeutigen Identifizierung [39] möglich. Techniken des Datamining erlauben es, durch Inferenz verschiedene Datensätze zusammenzuführen (bzw. mit großer Wahrscheinlichkeit richtigen Benutzern zuzuordnen).

Ein gezielter Angriff auf bestimmte Daten erfordert die Kenntnis des entsprechenden Anwendungsprotokolls; da wir einen generischen Ansatz zum Schutz der personenbezogenen Daten erlangen wollen, muß dieser Wissensvorsprung des Angreifers ausgeglichen werden: Wir benötigen auch auf der Anwendungsschicht entweder (analog zu den Daten der tieferliegenden Schichten) ein generisches, automatisch analysierbares Protokollformat; oder aber die Anwendung selbst muß in einem isolierten, kontrollierbaren System ablaufen, welches nur über wohldefinierte Schnittstellen mit anderen Systemen kommunizieren darf. Die Kontrolle der Daten würde dann an diesen Schnittstellen erfolgen.

3.8 Fazit

Schichtenmodelle bieten eine Abstraktionsmöglichkeit für verschiedene Kommunikationsmedien und erlauben es Treiber- oder Anwendungsprogrammierern, durch

Implementieren standardisierter Schnittstellen eine Schicht mit einer konkreten Instanz auszufüllen. Aus der Sicht der Privacyproblematik liefert diese Schnittstellendefinitionen auch die Informationen, die benötigt werden, um dieses Problem zu diskutieren; die strukturierten Headerformate erlauben sogar eine gezielte Steuerung.

Einzig die Anwendungsschicht bleibt normalerweise außen vor, da hier beliebige Anwendungsprotokolle übertragen werden können. Durch die von uns vorgeschlagene Strukturierung können wir aber auch hier eine Möglichkeit schaffen, die ausgehenden Daten zu kontrollieren.

Daher wird im folgenden Kapitel vorgestellt, wie mit den identifizierenden Informationen auf den Schichten mit standardisierten Headern verfahren wird. Das übernächste Kapitel schließlich stellt ein angepaßtes Anwendungsprotokoll vor, welches wiederum die Kontrolle privacy-relevanter Informationen ermöglicht. Somit wird die gesamte Kommunikation abgedeckt.

4 Tracking und Anonymisierung

> Es war sogar möglich, daß jeder einzelne ständig überwacht wurde. (...) Man mußte in der Annahme leben – und jeder stellte sich instinktiv darauf ein – daß jedes Geräusch, das man machte, überhört und, außer in der Dunkelheit, jede Bewegung beobachtet wurde.
>
> *George Orwell (1903–1950), „1984“*

4.1 Einleitung

Wie in Kapitel 3 geschildert, gibt es innerhalb der Netzwerkprotokolle mehrere Stellen, an welchen ein Gerät eindeutig identifiziert werden kann. Diese eindeutigen Nummern (welche letztlich Benutzerpseudonyme darstellen) werden benötigt, um den korrekten Ablauf verschiedener Protokolle zu gewährleisten. Allerdings widersprechen sie dem Wunsch nach Anonymität, weshalb sie zu geeigneten Gelegenheiten gewechselt werden sollen.

Definition 4.1 (Session/Sitzung) *Im folgenden bezeichnen wir die Zeitspanne, in der dasselbe Pseudonym genutzt wird, als* Session *oder* Sitzung.

Üblicherweise wird der Begriff der Sitzung für einen länger andauernden, zustandsbehafteten Vorgang benutzt (der möglicherweise sogar unterbrochen und zu einem späteren Zeitpunkt wieder fortgesetzt werden kann). Wir zielen jedoch darauf ab, diese Zustandsbehaftung möglichst kurz zu halten und so eine Durchmischung der Kommunikationen zu erreichen. Da aber auch kurze Kommunikationsabschnitte nicht immer ohne Zustandsinformation auskommen können, belegen wir diese in unserer Arbeit mit dem Sitzungsbegriff (beispielsweise bilden Verbindungsauf- und -abbau sowie die Datenübertragung mit verschiedenen Garantien bei TCP eine solche Session, da ohne die dafür nötigen Zustandsinformationen diese Leistungen nicht erfüllbar wären).

Die Zeitpunkte des Wechsels sind einerseits bestimmt durch die einzelnen Netzwerkprotokolle, andererseits bildet die Anwendungsschicht die oberste semantische Schicht, weshalb diese ebenfalls die Möglichkeit haben sollte, einen Pseudonymwechsel zu signalisieren.

Die Pseudonymwechsel sollten aus mehreren Gründen so häufig wie möglich erfolgen: Dem Benutzer ist die Mächtigkeit der Angreifer nicht bekannt; so können beispielsweise mehrere Kommunikationspartner ihre gesammelten Daten abgleichen

und so Benutzerprofile zusammenführen und dadurch zusätzliche Informationen gewinnen (wenn beispielsweise Diensteanbieter A kein Name bekanntgegeben wurde, bei B jedoch ein namentlicher Login erfolgte, und für beide dasselbe Pseudonym genutzt wurde, so können nach der Zusammenführung auch die Daten, die A kommuniziert wurden, eindeutig zugeordnet werden). Weiterhin können Datensammler nicht nur die tatsächlich übertragenen Daten nutzen, sondern den weiteren, impliziten Kontext wie Zeitpunkt der Übertragung, Ort, Zeitdauer zwischen zwei Anfragen, etc. nutzen.

Der häufige Wechsel kann zweierlei Dinge bewirken: Zum einen erschwert es Angreifern das Zusammenführen der Daten, da je Session weniger Information vorhanden ist, die einen Benutzer über die Gültigkeitsdauer des Pseudonyms hinweg identifizieren kann. Zum anderen stellen kurze Sessions mit daraus resultierender geringer Informationsmenge im Desasterfall (also der korrekten Zusammenführung mehrerer Sessions) eine Maßnahme zur Schadensbegrenzung dar, da immer noch die Chance besteht, daß der Angreifer nicht sämtliche, sondern nur einen Teil der Sessions des Nutzers zusammenführen konnte – auf diese Weise entsteht nicht automatisch eine globale, zusammenhängende Sicht der Dinge.

Das Zusammenführen von Sitzungen anhand von Informationen aus dem Anwendungsprotokoll wird im folgenden Kapitel diskutiert; dieses Kapitel betrachtet die Anonymisierung bzw. Pseudonymisierung unterhalb der Anwendungsschicht. Mit Hilfe von Simulationen untersuchen wir den Gewinn an Privatsphäre und die Resistenz gegen verschiedene Angreifermodelle. Die dabei verwendeten Angreifermodelle kommen mit möglichst wenig Randinformationen und Vorwissen aus, so daß sie möglichst generisch einsetzbar sind. Um eine untere Schranke für die erreichte Privatsphäre angeben zu können, nehmen wir in unserer Simulation einen sehr mächtigen Angreifer an, der die Positionsinformation sämtlicher sendender Knoten exakt und unverrauscht bestimmen kann.

4.2 Grundlagen und Vorarbeiten

Dieser Abschnitt ist – gemäß den Anforderungen – in drei Teile gegliedert. Zunächst erläutern wir die Grundlagen der Pseudonymerzeugung (und damit einhergehende Probleme). Der zweite Teil zeigt bestehende Ansätze zur Anonymisierung mobiler Knoten auf, der letzte Teil erläutert in der Literatur diskutierte Angriffsmöglichkeiten auf solche Verfahren.

4.2.1 Pseudonymerzeugung

Pseudonyme dienen der Adressierung einer Entität für eine gewisse Zeitspanne – in unserem Fall für eine Session (die Definition des Pseudonymbegriffs haben wir

bereits auf Seite 18 gegeben). An die Verfahren zur Erzeugung von Pseudonymen stellen wir insbesondere folgende Anforderungen:

- Kollisionsresistenz: Wenn zwei Knoten innerhalb desselben Kommunikationsbereichs dasselbe Pseudonym als Netzwerkkennung nutzen, stört das die Kommunikationsprotokolle in ihrem Ablauf. Daher muß sichergestellt werden, daß jedes Pseudonym eindeutig ist, bzw. geeignete Mechanismen zur Kollisionserkennung vorhanden sind.
- Einfache Erzeugung: Da der Wechsel des Pseudonyms möglichst häufig erfolgen soll, darf die Generierung eines neuen Pseudonyms nicht zu aufwendig sein.
- Unverknüpfbarkeit: Um die Privatsphäre der Nutzer zu schützen, darf es einem Angreifer nicht möglich sein, mehrere Pseudonyme des Nutzers erfolgreich derselben Entität zuzuordnen.

Prinzipiell unterscheiden wir zwei Arten von Pseudonymen: Zufällig generierte Pseudonyme sowie auflösbare Pseudonyme.

Zufällige Pseudonyme

Die einfachste Form zur Generierung von Pseudonymen besteht in der Wahl einer rein zufälligen Zahl. Ein Angriff auf die Unverknüpfbarkeit besteht damit aus einem Angriff auf die Entropiequelle[1] und ggf. den Pseudozufallszahlen-Generator.

Eine einfache Erzeugung sowie Unverknüpfbarkeit sind also zu gewährleisten. Die Kollisionsresistenz bedarf jedoch näherer Betrachtung: Angenommen, die Länge des Pseudonyms betrage b Bits und ein Knoten hätte immer n Nachbarn (mit denen also eine Kollision entstehen kann), so beträgt die Chance, daß keiner der Nachbarn dasselbe Pseudonym wählt

$$P(keine\ Kollision) = (1 - \frac{1}{2^b})^n$$

Entsprechend sinkt die Wahrscheinlichkeit bei t Pseudonymwechseln auf $P = (1 - \frac{1}{2^b})^{nt}$. Betrachtet man jedoch die Robustheit des gesamten Systems, also die Wahrscheinlichkeit, daß irgendeine Kollision auftritt, tritt der Effekt des *Geburtstagsparadoxons* (erstmals diskutiert um 1930 [142]) auf – die Wahrscheinlichkeit, daß es zu keiner Kollision kommt, fällt deutlich stärker:

$$P(keine\ Kollision) = (1 - \frac{n(n-1)}{2^{b+1}})^t$$

[1] Entropiequelle: In der Kryptographie übliche Bezeichung für eine Quelle echter Zufallsbits

Geht man von $n = 100$ Nachbarknoten, einem wählbaren Adressteil von $b = 24$ Bit Länge und $t = 240$ Pseudonymwechseln aus (was 20 Pseudonymwechsel pro Stunde auf einen Tag verteilt entspricht), so kommt es mit einer Chance von rund 25 % zu einer Adresskollision. Damit wird deutlich, daß eine Kollisionsbehandlung vonnöten ist. Weitere Betrachtungen der Kollisionswahrscheinlichkeit findet man in [96, 136].

Eine Kollisionserkennung kann entweder aktiv oder passiv erfolgen: Im passiven Fall beobachtet ein Gerät den Netzverkehr; wird eine Adresse für eine definierte Zeitspanne nicht verwendet, so geht das Gerät davon aus, daß diese frei ist.

Im aktiven Fall sendet ein Gerät eine Anfrage (Probe) mit der neu gewählten Adresse; ist die Adresse bereits in Verwendung, antwortet das entsprechende Gerät mit einer negativen Quittung. Als Zieladresse wird jeweils die lokale Broadcast-Adresse verwendet. Dieses Prinzip wird beispielsweise beim Gracious-ARP[2]-Protokoll [215, Kapitel 4.7, Seite 62] eingesetzt.

Auflösbare Pseudonyme

Bei auflösbaren Pseudonymen können mit Hilfe einer (geheimen) Zusatzinformation mehrere Pseudonyme derselben Entität zugeordnet werden. Wir unterscheiden hierbei Varianten, bei denen die Auflösung auf Initiative des Pseudonymnutzers geschieht sowie solchen Verfahren, bei welchen die Auflösung durch andere Personen (typischerweise eine vertrauenswürdige, unabhängige Partei) geschieht. Einsatzzweck für erstere Variante sind Szenarien, bei welchen ein Knoten im Nachhinein beweisen will, daß mehrere, in der Vergangenheit verwendete Pseudonyme von ihm stammen; die zweite Variante kann beispielsweise eingesetzt werden, um im Mißbrauchsfall auf eine Realperson zurückschließen zu können.

Zur ersten Kategorie gehört das Verfahren von Gruteser und Grunwald [96]: Die Autoren verwenden in ihrer Arbeit eine Hash Chain (siehe Anhang A.1.5 auf Seite 125), um Pseudonyme zu erzeugen: Von einem geheimen Startwert ausgehend generieren sie eine Reihe von Pseudonymen, welche dann in umgekehrter Reihenfolge verwendet werden. Dabei sei der Wertebereich der Pseudonyme kleiner dem der Hashfunktion, so daß nur ein Teil des Hashergebnisses für das Pseudonym verwendet wird; der Rest bleibt zunächst geheim, so daß es einem Beobachter nicht möglich ist, die Pseudonyme nachzurechnen. Erst durch Kenntnis eines kompletten Hashwerts (bzw. des Startwerts) ist es möglich, die Hashkette nachzuvollziehen.

Die einfachste Variante für die zweite Kategorie besteht in einer vertrauenswürdigen Partei, zu der jeder Netzteilnehmer eine sichere Verbindung besitzt: Die Partei weiß zu jeder Zeit, welcher Knoten unter welchem Pseudonym unterwegs ist; des weiteren kann sie die Frage beantworten, ob ein Pseudonym momentan gültig ist (sprich: ob das Pseudonym bei ihr angemeldet ist).

[2] ARP: Address Resolution Protocol. Protokoll des IP-Stacks, das eine lokale IP-Adresse in die zugehörige MAC-Adresse auflöse.

Dieses Verfahren wird bei GSM-Netzen verwendet [230]: Jeder Netzteilnehmer nutzt innerhalb einer GSM-Zelle eine TMSI[3]; dem VLR[4] wird die korrespondierende Identität in Form der IMSI[5] mitgeteilt. Beim Zellwechsel wird eine neue TMSI gewählt; so soll verhindert werden, daß nicht authorisierte Personen Bewegungsprofile erstellen können.

Dieser Ansatz gewährt Schutz vor externen Beobachtern, nicht aber vor Insiderangriffen innerhalb der GSM-Infrastruktur. Eine Möglichkeit, hiergegen Vorsorge zu treffen, stellen Kesdogan et al. [136] vor: Anstelle einer zentralen Instanz zur Auflösung der Pseudonyme schlagen die Autoren die Verwendung eines HTD[6] vor, welches komplett unter der Kontrolle des Nutzers steht. Als weitere Erweiterung wird die Information auf mehrere Parteien verteilt, so daß nur durch Kooperation aller Parteien die Information rekonstruiert werden kann.

In der Arbeit von Candebat et al. [50] werden die Pseudonyme mit Hilfe eines Vermittlers (Mediator) erzeugt; das Schlüsselmaterial wird zwischen Mediator und Benutzer mittels Schwellwert-Kryptographie (Threshold Cryptography) aufgeteilt.

Besteht das Ziel darin, eine geschlossene Nutzergruppe zu erhalten, so wünscht man sich „authentisierte Pseudonyme“, bei denen festgestellt werden kann, ob die Pseudonyme von einer zentralen Instanz ausgegeben wurden. Will man jedoch vermeiden, daß diese Instanz Kenntnis aller Pseudonyme besitzt (wie im obigen Beispiel bei GSM der Fall), so kann man die Signaturen mit Hilfe von Blind Signature Schemes [53, 54, 55] erzeugen. Ein Widerrufen von Pseudonymen, die auf diese Weise erzeugt wurden, ist jedoch nicht möglich.

Eine weitere Verfeinerung der Pseudonymerzeugung durch einen Identity Provider, bei der auch das Widerrufen sämtlicher Pseudonyme eines Nutzers möglich ist, stellt die Arbeit von Brands et al. [33] dar. Die Pseudonyme werden interaktiv mit dem Identity Provider erzeugt, so daß selbst dieser die resultierenden Pseudonyme nicht kennt; jedoch bettet dieser eine Art Wasserzeichen in die Pseudonyme ein, so daß er bei Vorlage von zwei Pseudonymen entscheiden kann, ob diese zum selben Benutzer gehören. Anhand dieser Markierung können auch sämtliche Pseudonyme eines Benutzers widerrufen werden.

Ähnlich geartete Arbeiten zum Erzeugen und Widerrufen von Pseudonymen wurden von Camenisch und Lysyanskaya [43, 44] veröffentlicht.

[3] TMSI: Temporary Mobile Subscriber Identity
[4] VLR: Visitor Location Register
[5] IMSI: International Mobile Subscriber Identity. Weltweit eindeutige Teilnehmerkennung für Mobilfunknetze.
[6] HTD: Home Trusted Device

4.2.2 Tracking

In einigen Fällen genügt ein Pseudonymwechsel allein nicht, um ein Zusammenführen von Pseudonymen zu verhindern: Ein Angreifer kann weitere Informationen nutzen, um eine Zuordnung zu erreichen. Dies kann durch Abgleich des Kommunikationsinhalts geschehen (beispielsweise zwei aufeinanderfolgende Kommunikationen, in denen dasselbe Geburtsdatum angegeben wird, gehören mit hoher Wahrscheinlichkeit zur selben Person); eine weitere, und im Kontext dieses Kapitels besonders relevante Information ist der Orts- und Bewegungskontext.

Eine Übersicht über die erste Problematik bietet die Arbeit von Clauß et al. [61]: Ausgehend von der Thematik des Identity Management betrachten geben die Autoren einen Überblick über Pseudonymerzeugung mit dem Ziel der Netzwerkanonymät; im Zuge einer Kategorisierung von möglichen Angriffsformen wird die Zusammenführung von Pseudonymen anhand von übertragenen Informationen diskutiert.

In [97] betrachten die Autoren die Möglichkeit eines Location-Based Serviceproviders (LBS-Provider), periodische Positionsdaten derselben Person zuzuordnen. Dabei versuchen die Autoren, die Daten anhand der Positionsinformationen zusammenzuführen. In der hier vorgestellten Idee werden Kalman-Filter [131] sowie das Multi-Hypothesen-Tracking nach Reid [182] verwendet, um die Datenspuren zu verfolgen – ursprünglich diente der Algorithmus von Reid dazu, Radarmessungen zu Flugzeugbezeichnungen zuzuordnen.

4.2.3 Anonymisierung

Je nach Szenario und gegebener Infrastruktur gibt es verschiedene Ansätze, eine Anonymisierung zu erreichen. Zunächst einmal lassen sich infrastrukturlose Szenarien von Umgebungen, die bestimmte Geräte oder Techniken bereitstellt, unterscheiden: In *infrastrukturlosen Szenarien* ist jeder Netzteilnehmer dazu gezwungen, nur mit eigenen Mitteln seine Anonymität zu schützen. Demgegenüber stehen *infrastrukturbasierte Systeme*, bei welchen die Knoten auf eine bestimmte (möglicherweise vertrauenswürdige) Umgebung aufbauen können. Diese Infrastruktur kann entweder zentralisiert oder verteilt existieren.

Eine weitere Möglichkeit der Klassifikation besteht nach der Art des Angriffs. Man spricht von *point anonymity*, wenn es um die Anonymisierung von einzelnen Anfragen bei einem LBS-Provider geht; die mögliche Verknüpfbarkeit von mehreren Anfragen wird hierbei außer Acht gelassen. Wird auch der Abgleich mehrerer Anfragen als potentieller Angriff betrachtet, so spricht man von *trace anonymity*.

Wir schlagen eine dritte Möglichkeit der Klassifikation vor: Die verwendete Methode, um die gesendeten Daten so zu verschleiern, so daß der gewünschte Grad an Anonymität erreicht wird.

Jamming: Mit Jamming wird das Blockieren der Kommunikation bezeichnet. Dies kann neben dem Einsatz eines Störsenders durch geeignetes Fehlverhalten in Protokollabläufen, beispielsweise durch gezieltes Herbeiführen von Kollisionen erreicht werden. Dadurch können andere, nicht privacy-sensitive Geräte an einer korrekten Antwort gehindert werden.

Inhibition: Durch gezieltes Weglassen von Daten kann die Anonymität gewahrt werden. Dies erfordert jedoch geeignete Protokolle und ist daher nur in einigen Szenarien einsetzbar.

Blurring: Erforderliche Information wird mit einer gewissen Unschärfe versehen. Durch eine partielle Randomisierung der Daten wird der Informationsgehalt für den Empfänger verringert; die Vergröberung kann je nach Dienst unterschiedlich sein – beispielsweise benötigt ein Dienst für die Wettervorhersage nicht die exakte Position des Benutzers. Punktanonymität ist hiermit sehr gut zu erreichen, beim Zusammenführen von mehreren Messwerten (trace anonymity) kann es jedoch zu Schwierigkeiten kommen [80].

Mixing: Durch geeignete Vorgehensweise versteckt sich ein Nutzer in der Menge der weiteren Benutzer. Dies kann beispielsweise durch Normierung des Verhaltens geschehen; durch Unterstützung einer entsprechenden Infrastruktur können Kommunikationspfade verschleiert werden. Mix-Netze für E-Mails [52] oder das Onion-Router-Netz TOR [78] (siehe weiter unten) sind klassische Anwendungen für diesen Ansatz.

Übersichtsarbeiten

Eine Übersicht über verschiedene Ansätze, Anonymität der Benutzer und Location Privacy zu erreichen gibt [12]. Die Arbeit betrachtet die Privacy-Mechanismen einiger etablierter Systeme (wie GSM) und illustriert in knapper Weise vier Ansätze, um Anonymität zu erhalten.

Umfangreicher und konkreter sind die Arbeiten von Görlach et al. [99, 100]: Sie stellen eine Klassifizierung von möglichen Angriffen auf die Privatsphäre vor (Abhören von Kommunikation, Informationsweitergabe, Beobachtung der Umgebung, Datenabgleich) und geben eine Übersicht über Publikationen rund um das Problemfeld der Privatheit der Ortsinformation.

Im Kontext des Discreet-Projekts (Discreet Service Provision in Smart Environments, [1]) wurde ein Deliverable mit dem State of the Art zum Thema Location Privacy erstellt [9]. Neben einer Zusammenfassung von Begriffsdefinitionen findet man hier eine Einführung zum Thema Privacy, einen Überblick über das Themenfeld Key Management, sowie eine Übersicht verschiedener Ansätze, um die Privatsphäre in mobilen Umgebungen zu erhalten. Der Fokus liegt hierbei ausschließlich auf infrastrukturbasierten Verfahren wie beispielsweise verschiedenen Mix-Netz-Ausprägungen.

Verfahren mit zentraler Infrastruktur

Zu den frühesten Arbeiten zur Anonymisierung von Übertragungen zählt die Arbeit von Chaum [52], welche eine Methode zum anonymen Versenden von E-Mails vorstellt. Anstatt eine Mail direkt dem Empfänger zuzustellen, wird sie über eine Kette von Zwischenstationen gesendet; diese Kette wird vom Sender vorbereitet, so daß die Zwischenstationen hier keine Einflußmöglichkeit haben. Um zu verhindern, daß an diesen Stellen Informationen über die Kommunikation gewonnen werden können, werden die Daten mit Hilfe eines Public-Key-Verfahrens geschachtelt verschlüsselt. Will beispielsweise A an B eine Nachricht M über die Stationen $1 \dots n$ versenden, sieht die Nachricht wie folgt aus:

$$E_1(2, E_2(3, E_3(\dots, E_B(M))))$$

Jede Zwischenstation entschlüsselt die Nachricht, und erhält dabei die Adresse der nächsten Station und einen Datenblock, der passend für diese verschlüsselt ist (siehe Beispiel in Bild 4.1). Verfahren, bei denen schrittweise solche geschachtelten Verschlüsselungsschichten entfernt werden, nennt man – der Analogie zum schichtartigen Aufbau einer Zwiebel – *Onion Routing*.

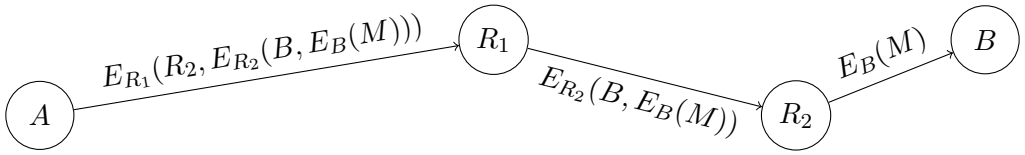

Abbildung 4.1: Mixnetz mit Onion Routing

Hätte man eine einzelne vertrauenswürdige Instanz, könnte diese Sender und Empfänger bereits vollständig entkoppeln; dies ist jedoch nur selten der Fall, außerdem bliebe sie gegen Angriffe von außerhalb, beispielsweise durch Beobachtung des Netzverkehrs, anfällig. Um diesen Angriffen zu entgehen, wird eine längere Kette solcher Rechner verwendet. Das Onion Routing ist genau dann erfolgreich, wenn mindestens ein Rechner in der Kette korrekt arbeitet: Jeder kompromittierte Rechner gibt dem Angreifer den Zusammenhang zwischen eingehendem und ausgehendem Paket preis. Ein einzelner funktionierender Knoten hingegen entkoppelt diese Kette.

Als weitere Vorsichtsmaßnahmen werden die einzelnen Schritte mit einem Padding zufälliger Länge versehen, um ein Wiedererkennen der Nachricht anhand ihrer Länge zu vermeiden. Zusätzlich senden die einzelnen Rechner die Nachricht nicht sofort weiter – ein potenter Angreifer, der das gesamte Netz beobachtet, könnte sonst den Nachrichtenpfad anhand der Kommunikationsaktivität verfolgen. Nachrichten werden eine bestimmte Zeit gesammelt und erst dann bearbeitet; durch die Längenveränderung sind eingehende und ausgehende Daten einander nicht mehr zuzuordnen,

durch die Blockverarbeitung gibt es auch keine zeitliche Korrelation mehr. Wegen dieser Eigenschaft des Vermischens nennt man solche Netze auch *Mix-Netze*.

Eine Übertragung dieses Ansatzes von E-Mails auf allgemeine TCP-Verbindungen stellt das TOR-Netzwerk [78] dar. Die Einstiegspunkte des Netzes agieren wie ein SOCKS-Proxy [153], so daß beliebige Netzwerkanwendungen, welche die Verwendung eines SOCKS-Proxys unterstützen, auf diese Weise anonymisiert werden können. Da bei TCP-Verbindungen (im Gegensatz zu E-Mails) Verzögerungen von mehreren Minuten nicht tragbar sind, wird hier auf ein Sammeln und Verzögern von Paketen verzichtet. Vielmehr wird darauf vertraut, daß die Paketfrequenz hoch genug ist, um eine ausreichende Vermischung zu erhalten; außerdem wird ein Angreifer, der das gesamte Netz beobachten kann, von den Autoren als hinreichend unrealistisches Szenario erachtet.

Die wohl populärste Anwendung von TOR ist die Verwendung in Kombination mit einem Webbrowser; dabei zeigt sich ein Problem eines solch generischen Ansatzes: Beim naiven Einsatz von TOR werden die Seitenanfragen selbst zwar anonymisiert – da jedoch die Serverkomponenten der URLs zunächst in IP-Adressen aufgelöst werden müssen, kann ein Beobachter anhand des DNS[7]-Verkehrs nachvollziehen, welche Sites betrachtet werden. DNS bildet so einen deanonymisierenden Seitenkanal, der separat behandelt werden muß.

Das regelmäßige Abfragen derselben Informationsquelle (beispielsweise des E-Mail-Accounts) kann bei mobilen Geräten zur Re-Identifizierung und damit dem Erstellen von Bewegungsprofilen führen; Cooper und Birman [62] stellen für solche Szenarien ein Zugriffsprotokoll vor, das ebenfalls auf dem Mix-Ansatz basiert.

Selbiges gilt insbesondere für Mobiltelefone, weshalb das Experiment, auch hier eine Anonymisierung via Mix-Netz einzusetzen, naheliegt; [86] demonstriert dies für das GSM-Netz.

Gruteser und Grunwald [95] schlagen vor, durch Einführen einer geeigneten Middleware-Architektur Zeit- und Ortsangaben sowie die Anfragefrequenz automatisiert zu verändern, um so die Location Privacy für die Dienstenutzer zu erhöhen.

Die Problematik, die Privatsphäre (insbesondere die der Ortsinformation) in kostenpflichtigen Location-based Services zu schützen wird von Kölsch et al. [144] diskutiert. Die Autoren verteilen die benötigten Funktionen auf mehrere Parteien: Den Application Provider, den Mobile Operator und einen Location Intermediary. Angelehnt an die existierenden Mobilfunknetze dient der Mobile Operator dem Betreiben der Kommunikationsinfrastruktur, dem Bestimmen der eigenen Position (die Position muß diesem ohnehin bekannt sein), sowie der Abrechnung. Der Application Provider liefert den gewünschten Dienst, der Location Intermediary dient der Vermittlung zwischen Application Provider und Mobile Operator. Alle Parteien verwenden für die jeweiligen Kommunikationspfade unterschiedliche Pseudonyme,

[7] DNS: Domain Name System. Protokoll zur Auflösung von Domainnamen zu IP-Adressen.

so daß ein Zusammenführen aller Informationen an keiner der drei Stellen möglich ist.

Das Gaia-Projekt versucht, eine Infrastruktur für Ubiquitous Computing zu erstellen, welche auch Security und Privacy berücksichtigt [49]. Zur Wahrung der Privatsphäre der Lokation und (auf Wunsch) der vollständigen Anonymität des Benutzers kommt hierzu das sogenannte *Mist Routing* zum Einsatz [7].

Das Mist Routing Protocol nutzt ebenfalls Mixe, um eine Anonymisierung zu erreichen; jedoch wird die Route nicht in einem mehrfach verschlüsselten Paket (wie beim Onion Routing) mitgegeben, sondern in Form von Circuits (analog zum Aufbau der Virtual Channels bei ATM [58, Kapitel 31]): Jedes Paket enthält eine Handle ID; jeder Router weiß für eine eingehende Handle ID, welcher Schlüssel zum Entschlüsseln benutzt wird, über welchen Link das Paket weiterversandt werden soll, wie die Handle ID dabei zu verändern ist und welcher Schlüssel für die Verschlüsselung benutzt werden soll (siehe Beispiel in Bild 4.2). Diese Circuits baut der Sender Schritt für Schritt auf. Für den Versand eines Pakets muß der Sender die Daten nun nur noch zwei mal verschlüsseln: Einmal für den Empfänger (Ende-zu-Ende-Verschlüsselung) und einmal für den nächsten Router (Hop-by-Hop-Verschlüsselung).

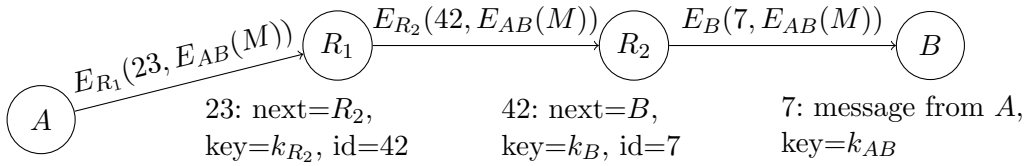

Abbildung 4.2: Mist Routing

Somit kann die Anonymität eines mobilen Knotens gewährleistet werden. Möchte nun ein Gerät einen mobilen Knoten erreichen, so kann dieser keine direkte Route zu ihm aufbauen, da ihm der Aufenthaltsort des Knotens nicht bekannt ist. Hierfür bedient sich das Mist Routing Protocol sogenannter *Lighthouses*, Stellvertreter-Rechnern mit fixer Adresse, welche die Daten dann an den mobilen Knoten weiterleiten können. Dieses Prinzip ist beispielsweise von Mobile IP (Home Agent) bekannt [173]. Der mobile Knoten baut eine Route zu seinem Lighthouse auf (und aktualisiert diese bei Veränderung der Position); über diese Route kann das Lighthouse eintreffende Daten weiterleiten. So bleibt die Lokationsinformation des mobilen Knotens sowohl für den Sender als auch für den als Lighthouse fungierenden Rechner verborgen.

Infrastrukturlose Verfahren

Die radikalste Art der Anonymisierung, ohne auf eine Infrastruktur zurückzugreifen, ist das Blockieren (Jamming) der Kommunikation. Um nur bestimmte Adreßberei-

che von RFID-Tags zu blockieren, haben Juels et al. [129, 127] einen „Blocker Tag“ entwickelt: Der Zugriff auf einzelne RFID-Tags erfolgt interaktiv entlang eines Binärbaums, wobei die Tags bei jedem Bit rückmelden, ob ihre Adresse mit 0 oder 1 fortgesetzt wird. Dies ergibt einen effizienten Zugriff auf mehrere Tags gleichzeitig.

Der Blocker Tag nutzt dies und meldet (im gewünschten Adreßbereich) immer, daß seine Adresse sowohl mit 0 als auch mit 1 weiterginge. Für das Lesegerät erweckt dies den Eindruck, es sei eine sehr große Zahl an RFID-Tags im Lesebereich. Die tatsächlich vorhandenen Tags verschwinden in der Menge der Falschmeldungen, außerdem wird der Lesezugriff sehr stark verzögert.

In [126] fassen die Autoren ihre Arbeit nochmals zusammen und präsentieren sie zusammen mit einer Übersicht über die Privacy-Problematik beim Einsatz von RFID.

Der „RFID Guardian“ von Rieback et al. [184] ist eine Art „Man in the Middle“ für RFID-Tags: Das Gerät agiert als RFID-Empfänger, besitzt aber auch ein RFID-Lesegerät. Auf diese Weise registriert es Zugriffe auf RFID-Chips und hat so die Möglichkeit, an deren Stelle zu antworten (z. B. um eine Zugriffskontrolle durchzuführen) oder die Antwort des eigentlich adressierten Chips durch Senden eines kurzen Störsignals komplett zu blockieren. Auf diese Weise realisieren die Autoren ein tragbares Gerät, das die Privatsphäre entsprechend vom Benutzer programmierten Präferenzen schützen kann.

Beresford und Stajano [23] schlagen vor, zum Schutz der Location Privacy in bestimmten Bereichen (vorzugsweise bewegungstechnisch neuralgischen Punkten wie Kreuzungen) die Kommunikation abzuschalten. Durch die Bewegung der Knoten vermischen sich die Bewegungsspuren, beim Verlassen dieser sogenannten *mix zone* wählen die Knoten ein neues Pseudonym. Die Vermischung verhindert das Zuordnen vom alten zum neu gewählten Pseudonym.

In einer Folgepublikation [22] untersuchen die Autoren Eintritts- und Austrittswahrscheinlichkeiten in die Mix Zone. Aus der Tatsache heraus, daß beispielsweise Fußgänger in einem Gang nur in den seltensten Fällen eine Kehrtwende machen, ergeben sich mitunter deutlich höhere Wahrscheinlichkeiten für bestimmte Eintritts-Austrittspaare, so daß ein Beobachter mit höherer Wahrscheinlichkeit als der Durchschnittswahrscheinlichkeit den Pfad eines Knoten verfolgen kann.

In einer weiterführenden Arbeit [116] zur der weiter oben beschriebenen Veröffentlichung von Gruteser und Hoh [97] versuchen die Autoren, durch gezieltes Verändern der Positionsdaten (*Path Pertubation*) die erwähnten Algorithmen zu stören. Dies ist im gewählten Szenario möglich, da die LBS-Provider die Position der mobilen Nutzer nicht selbst bestimmen, sondern diese ihre Geoposition selbst bestimmen und übertragen müssen.

In unserer Arbeit [193] untersuchten wir (ausgehend von einem anderen Szenario) zeitgleich mit Gruteser und Grunwald [96] den Einfluß von Kommunikationspausen und Pseudonymwechsel auf das Tracking von mobilen Geräten; im Gegensatz zu

Beresford und Stajano verwendeten wir hierbei keine dedizierte Mix Zone, sondern führten Kommunikationspausen nach einer bestimmten Zeitdauer einer Kommunikation ein. Dabei kamen wir zu ähnlichen Ergebnissen: Simulationen zeigten, daß mit zunehmender Teilnehmerzahl und längerer Pause zwischen einzelnen Kommunikationen der Trackingerfolg einbricht. Beide Veröffentlichungen benutzen jedoch nur relativ einfache Trackingverfahren, weshalb in diesem Kapitel hieran mit detaillierteren Simulationen angeknüpft werden soll.

4.3 Transient Mix-Zones durch Kommunikationspausen

Im vorigen Abschnitt sind wir auf verschiedene Vorarbeiten zur Anonymisierung von Netzwerkkommunikation eingegangen. Im folgenden diskutieren wir Vor- und Nachteile dieser Arbeiten und stellen unseren Ansatz der transienten Mixzonen vor. Um die Wirksamkeit dieses Ansatzes zu belegen, führen wir Simulationen durch; die hierfür verwendeten Algorithmen werden ebenfalls im folgenden beschrieben.

Ubiquitous Computing beschreibt eine Welt, in der wir ständig von Kleincomputern umgeben sind und in der die Kleingeräte, welche die Menschen mit sich herumtragen, in möglicherweise ständigem Kontakt mit ihrer Umgebung sind. Die meiste Zeit bewegen wir uns nicht in Fahrzeugen, sondern sind zu Fuß unterwegs. Daher liegt der besondere Fokus in den folgenden Untersuchungen auf Fußgängern.

In den Vorarbeiten gibt es eine Reihe von Ansätzen und Projekten (wie beispielsweise dem Gaia-Projekt), welche den Schutz der Privatsphäre ausreichend berücksichtigen und implementieren. Jedoch benötigen die meisten von ihnen hierfür eine geeignete Infrastruktur in der Umgebung der mobilen Geräte. Ein großflächiges Deployment einer solchen Infrastruktur ist jedoch höchst unwahrscheinlich und allenfalls für Insellösungen geeignet. Selbst infrastrukturlose Ansätze wie die Mixzonen benötigen die Information, in welchem Bereich die Geräte nicht kommunizieren sollen: Dies muß entweder eine vorab auf alle Geräte (welche ihre Position selbst bestimmen können) verteilt werden, oder aber durch Beacon-Signale vor Ort publik gemacht werden. Daher soll hier ein dezentraler Ansatz betrachtet werden, der ohne Vorbedingungen funktionieren kann.

Ein weiterer Kritikpunkt an den oben illustrierten Angriffen auf die Privatsphäre ist die Tendenz, auf einzelne Verfahren und bestimmte Situationen immer speziellere Angriffe vorzustellen – und umgekehrt immer speziellere Abwehrmaßnahmen vorzustellen (beispielsweise [116]).

Ähnliche Kritik gilt für die Angriffe auf die Anonymität der Mixzonen [22]: Hier liegt ein *Henne-Ei-Problem* vor – um die benötigten Statistiken zu erstellen, muß man für eine Zeit lang die Passanten verfolgen; genau diese Verfolgung versuchen die Anonymisierungsverfahren jedoch zu verhindern. Des weiteren gilt auch hier die Kritik der Spezialisierung: Solche Statistiken gelten nur für einen bestimmten Ort,

die Bewegungsmuster ändern sich in Abhängigkeit von der Tages- und Jahreszeit (was im übrigen in der Arbeit nicht einmal angesprochen wurde).

4.3.1 Szenario und Anforderungen

Das hier ausgeführte Szenario verzichtet auf zentrale Komponenten; sämtliche Kommunikation erfolgt direkt zwischen den beiden Kommunikationspartnern, ohne ein dazwischenliegendes Routing zu benutzen. Durch diese direkte Kommunikation wird der (im Ubiquitous Computing wichtige) Lokationskontext hergestellt: Typischerweise sind nur Geräte in der näheren Umgebung von Interesse.

Wie bereits erwähnt liegt der Fokus hier auf der Betrachtung von Fußgängern. Einige der verwandten Arbeiten beziehen sich ebenfalls explizit auf Fußgänger; da das Gewinnen von Bewegungsdaten einer großen Fußgängermenge sehr schwierig ist, greift man fast immer auf synthetisch generierte Bewegungsdaten zurück. Problem der meisten dieser Veröffentlichungen ist jedoch, daß sie für Simulationen nur die populären Bewegungsmodelle wie Random Walk einsetzen. Dieser Kritik wollen wir gerecht werden, weshalb bei den Simulationen auch spezielle Fußgängermodelle zum Einsatz kommen.

Das Anonymisierungsverfahren soll einen möglichst weitreichenden Schutz gegen einen Angreifer bieten, der die lokale Kommunikation überwachen kann. Der Schutz soll generische Trackingangriffe möglichst unmöglich machen. Hochspezialisierte Angriffe werden in diesem Szenario als hinreichend unwahrscheinlich betrachtet: Sie erfordern einen hohen Aufwand durch den Angreifer (wie das Erheben von Statistiken in [22]) und liefern im Gegenzug nur eine geringe Datenmenge – nämlich über den beobachteten Bereich, für den der Angriff spezialisiert wurde.

4.3.2 Klassifikation der Angreifer

In unserer Arbeit [193] stellten wir folgende Klassifikation von lokal agierenden Angreifern – geordnet nach ihrer Mächtigkeit – vor:

Lauschender Netzteilnehmer (casual listener): Ein Netzteilnehmer ohne weitere besondere technische Möglichkeiten; ein solcher Angreifer kann alle Nachrichten innerhalb der Empfangsreichweite abhören (durch Nutzung des Promiscuous Mode des eigenen Netzwerkinterfaces), jedoch keine Richtungspeilungen vornehmen.

Aktiver Verfolger (tracker): Als Tracker bezeichnen wir einen Angreifer, der in der Lage ist, die Bewegungen eines einzelnen Knoten (beispielsweise mittels Triangulation mit zwei Richtantennen) zu verfolgen. Durch eine zweite radiale Antenne können sie Aktivität der restlichen Knoten beobachten, nicht aber deren Position bestimmen.

Beobachter aus der Vogelperspektive (all-seeing observer): Dieser Angreifer ist eher theoretischer Natur – und zugleich das mächtigste Angreifermodell; der Angreifer besitzt die Möglichkeit, die Position aller aktiven Knoten zu bestimmen.

Für einen realen Angriff wären alle Positionsbestimmungen mit einem gewissen Fehler behaftet; für unsere Betrachtungen gehen wir jedoch von der (härteren) Annahme aus, daß die Positionsbestimmung exakt erfolgt.

4.3.3 Transient Mix Zones

Unser hier vorgeschlagener Ansatz zur Anonymisierung sei wie folgt definiert:

- Ein Knoten führt eine Session unter dem Pseudonym T_i. In dieser Zeit ist er sowohl verfolgbar (über Triangulation ist seine Position bestimmbar) als auch für Angreifer, die mächtiger als ein Casual Listener sind, identifizierbar: Selbst wenn der Knoten für verschiedene Kommunikationspartner unterschiedliche Pseudonyme benutzen würde, könnten diese durch die Positionsbestimmung dem selben Ausgangspunkt (und damit dem selben Knoten) zugeordnet werden.
- Nach Beenden der Session stellt der Knoten für eine Zeitspanne von mindestens t_b sämtliche Kommunikation ein (*backoff time*). Durch die Bewegung des mobilen Knotens entsteht durch die Vermischung mit den anderen Knoten eine Art vorrübergehende Mixzone. Der Name *Transient Mix Zone* wurde in Anlehnung an die Mixzonen von Beresford und Stajano gewählt; im Unterschied zu diesen handelt es sich aber eben nicht um fest definierte Bereiche ohne Kommunikation.
- Ist die Zeitspanne t_b vorbei und will der Knoten eine neue Session beginnen, so wählt er für diese ein neues Pseudonym T_{i+1}. Wegen der Unverknüpfbarkeit der Pseudonyme bleibt dem Angreifer als einzige Verfolgungsmöglichkeit die Prädiktion der Knotenbewegung.

Der Grad der Anonymisierung steigt mit der Zahl der Knoten in der Umgebung sowie der Zeitspanne t_b [193]. Dies wird in den Simulationen in diesem Kapitel näher untersucht.

4.3.4 Bewegungsmodelle

Bewegungsmodelle dienen dazu, *synthetische Bewegungsprofile* zu erzeugen; man bedient sich dieser Modelle zumeist, weil das Erfassen realer Bewegungsdaten mit vertretbarem technischen Aufwand nicht realisierbar ist. Mitunter werden bestimmte einfache Modelle angenommen, da diese bestimmte stochastische Eigenschaften besitzen, welche eine Berechnung von Wahrscheinlichkeiten zulassen.

Die wichtigsten und für die Simulationen hier relevanten seien im folgenden kurz vorgestellt; eine Übersicht über weitere Bewegungsmodelle findet man in [36, 101].

Random Walk

Das Random-Walk-Modell (manchmal auch als Brown'sches Bewegungsmodell bezeichnet) gehört zu den klassischen, einfachen Bewegungsmodellen. Jeder Knoten bewegt sich in eine zufällige Richtung $[0, 2\pi[$ mit einer zufällig gewählten Geschwindigkeit $[v_{min}, v_{max}]$ für eine konstante Zeit t (oder, in manchen Varianten, bis eine Strecke der Länge l zurückgelegt wurde). Im Anschluß daran werden Richtung und Geschwindigkeit erneut zufällig bestimmt – das Modell ist gedächtnislos, vorherige Schritte haben keinen Einfluß auf die Entscheidung für die neuen Parameter. Dies führt zu unrealistisch wirkenden Sprüngen in der Geschwindigkeit und zu unnatürlich spitzen Winkeln bei den Richtungswechseln.

Für das Verhalten am Rand gibt es ebenfalls verschiedene Varianten: Meist werden Knoten, die gegen den Rand stoßen, reflektiert; beim *Unbounded Random Walk* landen Knoten beim Übertreten des Randes auf der gegenüberliegenden Seite der Simulationsumgebung. Die Dichtefunktion eines zweidimensionalen Unbounded Random Walks läßt sich rechnerisch bestimmen [211].

Random Waypoint

Die Bewegung im Random-Waypoint-Modell teilt sich in eine Bewegungs- und eine Ruhephase. Ein Knoten wählt (gleichverteilt) zufällig einen Zielpunkt innerhalb des Simulationsbereichs, sowie eine Geschwindigkeit $[v_{min}, v_{max}]$, mit der er sich auf den Zielpunkt zubewegt. Nach Erreichen des Ziels pausiert der Knoten für eine konstante Zeit t, bevor er einen neuen Zielpunkt wählt.

Eine Variante des Modells simuliert Bereiche besonderer Attraktivität (sog. *Hot Spots*), indem diese bei der Wahl der Zielpunkte mit einer höheren Wahrscheinlichkeit gewichtet werden.

Eine geschlossene Darstellung der Dichtefunktion im Zweidimensionalen ist nicht bekannt, jedoch gibt es Untersuchungen und Abschätzungen für die Aufenthaltswahrscheinlichkeiten der Knoten [27, 26].

Bewegungsmodelle für Fußgänger

Die obigen, klassischen Bewegungsmodelle gehen freilich nicht auf Besonderheiten einzelner Szenarien ein – dazu sind sie (bewußt) zu abstrakt. Spezialisierte Bewegungsmodelle machen insbesondere dann Sinn, wenn der Abstraktionsgrad der

allgemeinen Modelle zu hoch ist. Beispielsweise gibt es für Bewegungen von Fahrzeugen gesonderte Bewegungsmodelle, welche berücksichtigen, daß sich die Knoten nur auf Straßen und entsprechend der Verkehrsvorschriften (Höchstgeschwindigkeit, Fahrtrichtung) bewegen können [241, 57].

Die realistische Simulation von Fußgänger-Bewegungen ist insbesondere bei der Gebäude- und Städteplanung von großer Wichtigkeit: Zum einen versucht man auch hier, Staus und stockende Bewegungen bei größeren Fußgängermengen zu vermeiden. Zum anderen werden hier die Auswirkungen von Flucht- und Paniksituationen untersucht.

Viele der Fußgängermodelle sind *hybride Bewegungsmodelle*: Sie unterscheiden zwischen einer *makroskopischen Bewegung* (beispielsweise der Routenplanung und daraus resultierenden Wegpunkten, die der Reihe nach angesteuert werden) und einer *mikroskopischen Bewegung* (wie dem Ausweichen entgegenkommender Passanten) [110].

Insbesondere im mikroskopischen Bereich entfalten die Fußgängermodelle ihre besondere Dynamik. Analogien, die in verschiedenen Ansätzen verfolgt werden, sind Bewegungen entsprechend der Gaskinetik [179] oder Flüssigkeitsdynamik [135]. Eine knappe Übersicht über die Entwicklungsschritte moderner Bewegungsmodelle für Fußgänger (dem Benefit-Cost-Modell, dem Magnetic-Force-Modell [170], sowie dem weiter unten detailliert beschriebenen Social-Force-Modell) findet man in [225].

In [183] findet man eine Untersuchung von Fußgängerbewegungen, um realistische Bewegungsmuster für Charaktere in Computerspielen zu erreichen. Dieser Ansatz modelliert verschiedene Verhaltensmuster wie Verfolgen, oder Gruppenbildung explizit (im Gegensatz zu anderen Ansätzen, bei denen solche Verhaltensweisen von selbst entstehen).

Eine detaillierte Betrachtung von mikroskopischem Bewegungsverhalten bei Fußgängern findet man in [37]. Hier werden Bewegungsmuster in Gebäuden, z. B. beim Durchqueren eines Gangs oder dem Abbiegen um eine Ecke bei Probanden beobachtet und untersucht.

Eine Mischung von Routenplanung und Fußgängermodell mit sozialer Komponente stellt die Arbeit von Pelechano et al. [172] dar. Sie stammt aus dem Bereich der Panik- und Fluchtforschung.

Social-Force-Bewegungsmodell

Das Social-Force-Modell nach Helbing et. al. [112, 111] gehört zu den renomiertesten und in der Literatur am häufigsten erwähnten modernen Fußgängerbewegungsmodellen. Es zeichnet sich insgesondere dadurch aus, daß es Effekte der Selbstorganisation wie das spontane Bilden von Marschkorridoren (*lane formation*) simuliert, diese aber nicht explizit formuliert; solche Effekte sind auch in Fußgängergruppen

beobachtbar, obwohl kein Mensch bewußt seine Wegstrecke danach wählt. Selbiges gilt für „Tropfeneffekte“ an Korridorverengungen wie Türöffnungen.

Die Bewegung eines Knotens setzt sich aus mehreren Komponenten zusammen (exemplarisch dargestellt in Abbildung 4.3):

Bewegungsziel: Jeder Knoten i bewegt sich auf einer für ihn möglichst bequemen (also möglichst direkten) Linie auf einen Zielpunkt zu. Dies versucht er, mit seiner bevorzugten Geschwindigkeit zu tun; wich die Geschwindigkeit zum vorhergehenden Zeitpunkt von diesem bevorzugten Wert ab, so nähert sich der Knoten dieser Geschwindigkeit innerhalb der sog. Relaxation Time an. Der resultierende Vektor aus Richtung und tatsächlicher Geschwindigkeit ist im Beispiel mit $\vec{f}_i^0$ notiert.

Beeinflussung durch andere Knoten: Ein Knoten i ändert seine Richtung so, daß er allen anderen Knoten (vorzugsweise) nicht zu nahe kommt; dies repräsentiert die Aversion, mit anderen Personen zusammenzurempeln. Dies wird durch eine abstoßende Kraft repräsentiert; diese ist umso größer, je näher der jeweils andere Knoten ist. Im Beispiel befindet sich Knoten j in der Nähe von Knoten i; die resultierende Abstoßungskraft auf i ist mit $\vec{f}_{i,j}(d_{i,j})$, also als Funktion in Abhängigkeit des Abstands $d_{i,j}$ dargestellt.

Beeinflussung durch Hindernisse: Es widerstrebt Fußgängern auch, mit Hindernissen zu kollidieren (der Rand einer Simulationsumgebung wird üblicherweise ebenfalls als Hindernis modelliert) oder an ihnen entlangzustreifen. Dies wird ebenfalls durch eine abstoßende Kraft modelliert. Im Bild sieht man ein Hindernis o, das sich im Abstand $d_{i,o}$ zum Knoten i befindet; die resultierende Abstoßung beträgt $\vec{f}_{i,o}(d_{i,o})$.

Die resultierende Kraft (im Beispiel: $\vec{f}_i$) ist die Resultierende der Kraftvektoren; diese wird abschließend auf eine Maximalgeschwindigkeit begrenzt sowie die Richtung um eine zufällige Fluktuation verändert.

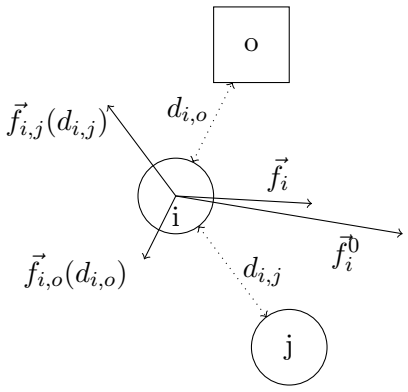

Abbildung 4.3: Beeinflussungskomponenten im Social-Force-Modell

Beobachtungen haben ergeben, daß die bevorzugte Geschwindigkeit von Fußgängern als Gaußverteilung mit einem Mittelwert von $1,34\,\frac{m}{s}$ und einer Standardabweichung von $0,26\,\frac{m}{s}$ angenommen werden darf [113, 233]. Die maximale Geschwindigkeit liegt dabei 30 % über der bevorzugten Geschwindigkeit.

4.3.5 Trackingverfahren

In unseren Simulationen kommen zwei Trackingverfahren zum Einsatz: Der Simple Tracker sowie das Multi-Hypothesen-Tracking. Diese seien im folgenden beschrieben. Eine Übersicht über weitere Trackingalgorithmen und -filter findet man in [219, 101, 140].

Simple Tracker

Der Simple Tracker (vorgestellt von uns in [193]) ist ein sehr einfaches Trackingverfahren. Die einzigen Parameter, die der Tracker benötigt, sind die maximale Knotengeschwindigkeit v_{max} sowie die minimale Backoff-Zeit t_b.

Nachdem ein beobachteter Knoten die Kommunikation einstellt, nimmt dieser Algorithmus den letzten bekannten Aufenthaltspunkt als Mittelpunkt M für eine kreisförmige Suche. Der Algorithmus nimmt an, daß der nächste Knoten, der nach einer Zeitspanne $t \geq t_b$ innerhalb eines Kreises um M mit Radius $t \cdot v_{max}$ neu erscheint, wiederum derselbe Knoten ist.

Um die Beobachtung eines einzelnen Knotens durchführen zu können, muß der Angreifer mindestens die Mächtigkeit des oben beschriebenen aktiven Verfolgers besitzen. Vorteil dieses Modells sind die geringen Anforderungen, die geringe Menge an Annahmen (das Vorwissen ist leicht zu erhalten bzw. einfach zu schätzen) sowie die Unabhängigkeit vom Bewegungsprofil der beobachteten Knoten.

Kalman-Filter

Der Kalman-Filter ist ein Algorithmus zur linearen Prädiktion von verrauschten Messreihen [131]. Der Filter arbeitet in einer *Prädiktor-Korrektor-Schleife* (siehe Abbildung 4.4), d. h. daß der Filter jede Prädiktion mit der zugehörigen Beobachtung vergleicht und seine Parameter entsprechend anpaßt. Der Algorithmus versucht dabei, den quadratischen Fehler zu minimieren.

Im Laufe der Zeit wurden verschiedene Varianten des Kalman-Filters (z. B. auch für nichtlineare Approximation) entwickelt; die folgende Beschreibung bezieht sich auf die Variante aus der Veröffentlichung von Reid [182], welche im folgenden Absatz im Multi-Hypothesen-Tracking zum Einsatz kommt. Diese Variante nimmt an, daß

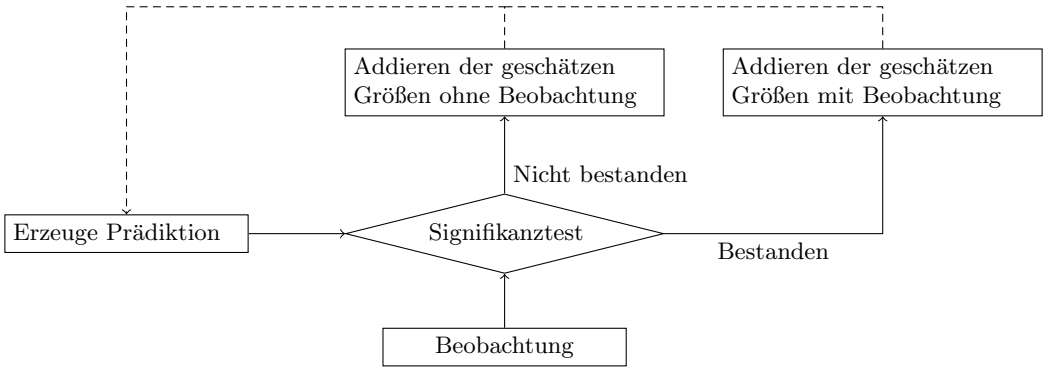

Abbildung 4.4: Ablauf des Kalmanfilter-Algorithmus

sich die Veränderung des Zustands $x(k)$ der beobachteten Objekte mit folgender Regel beschreiben (oder zumindest annähern) läßt:

$$x(k+1) = \Phi x(k) \tag{4.1}$$

Die Zustandsmatrix beschreibt die Veränderungen des Zustandsvektors während einer Iteration. Werden beispielsweise Position und lineare Geschwindigkeit als Objekteigenschaften gewählt (also $x = (x, y, v_x, v_y)^T$), so würde die entsprechende Zustandsübergangs-Matrix (für ein Zeitinvervall t) wie folgt aussehen:

$$\Phi = \begin{pmatrix} 1 & 0 & t & 0 \\ 0 & 1 & 0 & t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Prädiktion bestimmt den geschätzten Zustand mit Formel 4.1. Die Unsicherheit der Schätzung wird mit Hilfe der Zustands-Kovarianz-Matrix $\bar{P}(k)$ ausgedrückt:

$$\bar{P}(k) = \Phi \cdot \bar{P}(k-1) \cdot \Phi^T + \Gamma \cdot Q \cdot \Gamma^T \tag{4.2}$$

Die Beeinträchtigungsmatrix Γ selektiert gewissermaßen die Werte, die aus beobachteten Werten berechnet werden, Q stellt die Modellunsicherheit dar (da in unserem Beispiel Φ abhängig vom Zeitintervall t ist, ist hier auch Q abhängig von der Zeit). Angenommen, man würde im obigen Beispiel nur die Position der Objekte messen, so sähe die Beeinträchtigungsmatrix wie folgt aus:

$$\Gamma = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Für den in Abbildung 4.4 dargestellten Signifikanztest verwendet Reid den η-σ-*Test*:

$$(z(k) - H \cdot x(k))^T \cdot B^{-1} \cdot (z(k) - H \cdot x(k)) \leq \eta \tag{4.3}$$

mit

$$B = H \cdot \bar{P}(k) \cdot H^T + R \tag{4.4}$$

R ist hierbei die Kovarianzmatrix der Messungen. Die Meßwert-Matrix H ist das Gegenstück zu Γ: Sie selektiert die Werte, die tatsächlich beobachtet und gemessen werden. In unserem Beispiel muß sie demnach folgendermaßen aussehen:

$$H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Im Standardprozeß des Kalmanfilters, wie in Bild 4.4 dargestellt, erfolgen unterschiedliche Aktualisierungen in Abhängigkeit des Signifikanztests: Wurde die Messung als zu verrauscht klassifiziert, wird mit den Modelldaten aktualisiert; ansonsten wird die Messung mit dem Zustand des Filters verrechnet.

Dies geht davon aus, daß eine Messung einem beobachteten Objekt bereits zugewiesen wurde; beim Multi-Hypothesen-Tracking ist dem jedoch nicht der Fall. Hier wird der Signifikanztest dazu verwendet, um die Plausibilität einer Hypothese zu überprüfen: Wird der Signifikanztest erfüllt, so wird der Filter aktualisiert und eine weitere Hypothese erstellt. Schlägt der Test hingegen fehl, wird die Hypothesenannahme verworfen.

Bei der Aktualisierung der Filter werden Meßwerte und Prädiktion miteinander verknüpft. Die hierbei verwendete Kalman-Verstärkungsmatrix K bestimmt Reid wie folgt:

$$K = \bar{P}(k) \cdot H^T \cdot R^{-1} \tag{4.5}$$

Der aktuelle Zustand (der dann wiederum im nächsten Prädiktionsschritt verwendet wird) wird wie folgt verändert:

$$x(k) = x(k) + K \cdot (z(k) - H \cdot x(k)) \tag{4.6}$$

Die Zustands-Kovarianz-Matrix wird ebenfalls aktualisiert:

$$\bar{P}(k) = \bar{P}(k) - \bar{P}(k) \cdot H^T \cdot (H \cdot \bar{P}(k) \cdot H^T + R)^{-1} \cdot H \cdot \bar{P}(k) \tag{4.7}$$

Multi-Hypothesen-Tracking

Das Multi-Hypothesen-Tracking (MHT) wurde von Reid [182] mit dem Ziel entwickelt, mehrere Positionsdaten von Fahrzeugen (insbesondere Flugzeugen) zu einer Bewegungsspur zusammenzuführen. Zur Verfolgung der einzelnen Objekte benutzt das Verfahren einen Kalmanfilter (wie im vorigen Abschnitt beschrieben).

Beim normalen Kalmanfilter-Ablauf (Abbildung 4.4 auf Seite 43) wird ein Meßwert immer fest mit einem Filter assoziiert – es wird lediglich unterschieden, ob der Meßwert zur Aktualisierung des Filterzustands verwendet wird, oder ob die Messung als zu verrauscht klassifiziert und damit verworfen wird. Da eine solche feste Zuordnung von einem Meßpunkt zu einer Bewegungsspur sich rückblickend als falsch (oder zumindest unwahrscheinlich) erweisen kann, bildet das MHT einen Hypothesenbaum, der alle möglichen Zuordnungen von Messungen zu Objekten beinhaltet.

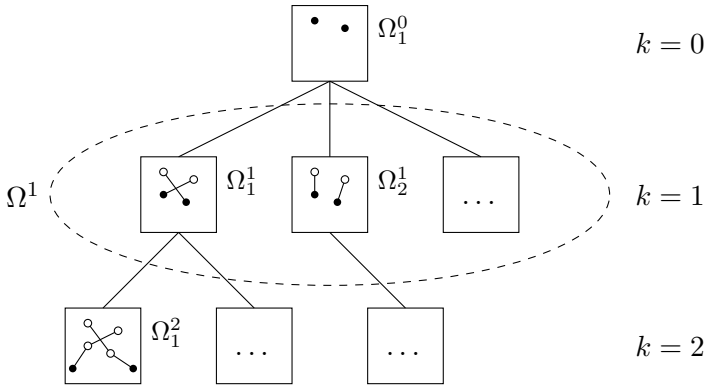

Abbildung 4.5: Beispiel-Hypothesenbaum für den MHT

Zu jedem Zeitschritt k wächst somit der Baum um eine weitere Ebene, indem die Blattknoten aus Ebene $k-1$ um die möglichen Zuordnungen der Messungen $Z(k) = \{z_m(k) \,|\, m = 1, 2, \ldots, M_k\}$ zum Zeitpunkt k ergänzt werden (siehe auch Beispiel in Bild 4.5). Die Zuordnung von Messung zu Objekt ist in jeder Hypothese eindeutig, d. h. jede Messung ist maximal einem Objekt zugeordnet. Die Menge aller Hypothesen, die zum Zeitpunkt k entstehen, wird mit Ω^k bezeichnet.

Der Ablauf des gesamten Algorithmus ist in Bild 4.6 dargestellt. Jede Hypothese beinhaltet die Zustände der Kalmanfilter der einzelnen Objekte. Nun wird für jede Hypothese eine neue Prädiktion erstellt, welche mit den Meßdaten abgeglichen wird. Für jedes Paar von Meßwerten und Objekt-Prädiktionen wird der η-σ-Test durchgeführt; besteht der Meßwert diesen Test, so wird das Paar als gültige Hypothese verzeichnet. Alternativ kann jeder Meßwert ein neues Objekt darstellen oder ein Meßfehler sein. Objekte, denen kein Meßwert zugeordnet wurde, können entweder temporär verdeckt, oder aber aus dem Meßbereich verschwunden sein.

Die Meßwerte beim Tracking von mobilen Knoten entsprechen dem Sensortyp 1 aus Reids Veröffentlichung; die Wahrscheinlichkeit einer Prädiktion beträgt hier:

$$P_i^k = \frac{1}{c} P_D^{N_{DT}} (1 - P_D)^{N_{TGT} - N_{DT}} \beta_{FT}^{N_{FT}} \beta_{NT}^{N_{NT}} \times \left[\prod_{m=1}^{N_{DT}} \mathcal{N}\left(z_m - H_m x_m, B_m\right) \right] P_g^{k-1} \tag{4.8}$$

Dabei ist:

P_i^k	Wahrscheinlichkeit der Prädiktion Ω_i^k auf Ebene k
P_g^{k-1}	Wahrscheinlichkeit des Vaterknotens, von dem Ω_g^{k-1} abstammt
N_{DT}	Anzahl der Messungen, die dem selben Objekt zugeordnet wurden
N_{FT}	Anzahl der fehlerhaften Messungen
N_{NT}	Anzahl der Messungen, die neuen Objekten zugewiesen wurden
N_{TGT}	Anzahl der Ziele
P_D	Erkennungswahrscheinlichkeit
P_D^{NT}	Erkennungswahrscheinlichkeit neuer Objekte
$\beta_{FT}^{N_{FT}}$	Dichte neu erkannter Objekte
$\beta_{NT}^{N_{NT}}$	Dichte der Fehlerkennungen
z_m, x_m	Geschätzter und korrespondierender Meßwert m zum Zeitpunkt k
H_m, B_m	Werte des Kalmanfilters (siehe Formel 4.4)
$\mathcal{N}(\eta, \sigma^2)$	Normalverteilung mit Erwartungswert η, Standardabweichung σ^2
c	Normierender Faktor ($\sum_i P_i^k = 1$)

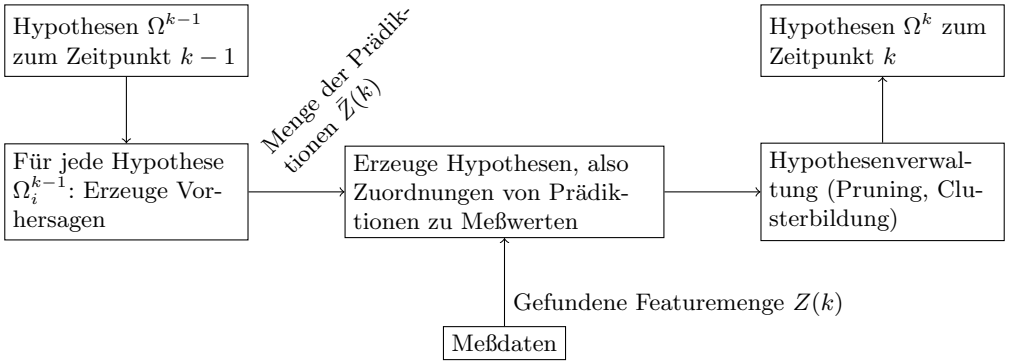

Abbildung 4.6: Ablauf des MHT-Algorithmus

Es ist offensichtlich, daß die Größe des Hypothesenbaums exponentiell anwächst. Um die Anzahl der Hypothesen zu reduzieren, schlägt Reid vor, die Prädiktionen

in Cluster einzuteilen, welche ein ausreichend ähnliches Bewegungsverhalten aufweisen. Dies reduziert die Anzahl an Kindknoten unter jedem Blatt.

Ein weiterer naheliegender Ansatz ist die Limitierung der Anzahl der neuen Hypothesen auf die n wahrscheinlichsten: Es werden für einen Blattknoten zunächst alle Hypothesen gebildet und nach ihrer Wahrscheinlichkeit sortiert. Die n wahrscheinlichsten bleiben erhalten, die restlichen werden sofort verworfen. Dies limitiert den Baum auf einen maximalen Grad n. Alternativ können alle Hypothesen unter einer gewissen Wahrscheinlichkeit entfernt werden – beispielsweise kann als Grenze ein Wert genutzt werden, unterhalb von dem angenommen werden kann, daß die numerischen Fehler bei der Weiterführung größeren Einfluß als neue Meßwerte haben dürften.

Um den Baum zu beschneiden, wird üblicherweise der Baum ab einer bestimmten Tiefe t beschnitten (*Multiscan-Verfahren*, *t-scan-Verfahren*): Erreicht der Baum eine Tiefe $t + 1$, so bestimmt man für jeden Knoten auf Tiefe 1 die Summe der Wahrscheinlichkeiten der Blätter, die an dem jeweiligen Knoten hängen. Der Knoten, der die höchste Summe erreicht, wird die neue Wurzel des Baums; die alte Wurzel sowie alle Geschwisterknoten (samt Teilbäume) werden verworfen. Somit schrumpft die Tiefe des Baums auf t. Die primitivste Form des Verfahrens, bei der faktisch auf den Hypothesenbaum vollständig verzichtet wird, nennt man *Zero-Scan-Verfahren.*

4.3.6 Simulation

Es gibt zwei wesentliche Faktoren, welche die Privatsphäre in den transienten Mix-Zonen beeinflussen: Die Backoff-Zeit sowie die Anzahl der Knoten. Sind in stark vereinfachten Modellen solche Zusammenhänge noch berechenbar, behilft man sich in komplexen Systemen mit Simulationen. In unserer Simulation sollen neben den traditionellen Bewegungsmodellen auch ein realistisches Bewegungsmodell zur Erzeugung synthetischer Fußgängerbewegungen zum Einsatz kommen, um so möglichst realistische Meßergebnisse zu erlangen. Unser simulierter Angreifer erhält mit dem Multihypothesen-Tracking ein sehr mächtiges Werkzeug zur Verfolgung der Spuren.

Unser Ziel ist es, die untere Grenzen zu finden, die für ein ausreichendes Maß an Privatsphäre benötigt werden; damit stellt sich also die Frage nach der Mindestanzahl an Knoten, die nötig sind, so daß der Mix-Ansatz funktioniert, ebenso wie die Mindestdauer der Backoff-Zeit. Die Annahmen über die Fähigkeiten des Angreifers werden bewußt pessimistisch getroffen: Die Ergebnisse unserer Simulation sollen eine Worst-Case-Abschätzung bieten; so wollen wir gewährleisten, daß bei einem realen Angriff die Ergebnisse eines Angreifers immer schlechter als unsere Simulationsergebnisse sind.

Für unsere ersten, einfachen Simulationen [193] wurde eine selbst geschriebene Software verwendet. In einer daran anschließenden Arbeit [101] wurde der Simulator

JiST/Swans [16] um entsprechende neue Module erweitert. JiST/Swans wurde dem häufig verwendeten Simulator NS-2 [3] aus zwei Gründen vorgezogen: Laufzeit und Ressourcenbedarf sind bei JiST bedeutend geringer. Des weiteren wurden für die Simulationen nur die Daten aus den relativ einfach nachzuvollziehenden Bewegungsmodellen genutzt, nicht aber die komplexe Netzwerk- und Übertragungsschicht; aus diesem Grund ist eine Vergleichbarkeit der Ergebnisse der beiden Simulationen gewährleistet.

JiST/Swans bot bereits Implementierungen der Bewegungsmodelle Random Walk und Random Waypoint; das Social-Force-Bewegungsmodell wurde für diese Arbeit neu implementiert. Außerdem wurde der Simulator um einen generischen Loggingmechanismus für die Zustände der simulierten Objekte erweitert – dieser lieferte die Daten für die verschiedenen Trackingalgorithmen, welche getrennt vom Simulator implementiert wurden. Durch diese Trennung war es einfach, verschiedene Trackingverfahren auf derselben Datenbasis zu testen.

Wahl der Parameter und der Erfolgsmetrik

Da der Zusammenhang zwischen Erfolg der Trackingverfahren, Anzahl der Knoten und der Backoff-Zeit t_b untersucht werden soll, sind diese Werte variabel. Das Anonymisierungsschema soll verhindern, daß aufeinanderfolgende Sessions eines Knotens demselben realen Knoten zugeordnet werden können. Unser Angreifer versuchte in jedem Simulationslauf, sämtliche Knoten zu tracken. Als Metrik für den Erfolg benutzen wir den Durchschnittswert der maximalen Zeitspanne von jedem Knoten, über welche es dem Angreifer möglich war, eine Folge von Sessions korrekt zuzuweisen. Vorteil dieser Metrik ist, daß sie unabhängig von t_b ist, was eine einfache Vergleichbarkeit ermöglicht.

Unsere simulierten Angreifer hatten die Mächtigkeit eines *all-seeing observers*, d. h. sie erhielten die Positionsinformation aller Knoten. Eine Meßungenauigkeit wurde nicht simuliert, die Messungen unserer Angreifer waren immer absolut präzise.

Der Simple Tracker besitzt als einzigen Parameter den Suchradius; da die Bewegungsgeschwindigkeit der Fußgänger in allen Bewegungsmodellen zufällig zwischen $1,0\ m/s$ und $1,34\ m/s$ gewählt wurde, wurde der Suchradius auf die maximale Bewegungsgeschwindigkeit $v_{max} = 1,34\ m/s$ gesetzt.

Wesentlich aufwendiger gestaltete sich die Findung geeigneter Parametersätze für das Multihypothesen-Tracking. In den Veröffentlichungen von Reid finden sich keine Hinweise zur Wahl der Parameter, weshalb die Einstellungen experimentell bestimmt werden mußten.

Die Parameter, welche für das Erkennen neuer bzw. verschwindender Knoten benutzt werden, wurden auf 0 gesetzt: Der Angreifer erhielt also die Information, daß keine Knoten den Simulationsbereich verließen und auch keine neuen Knoten die-

sen betraten. Die Parameter des Kalmanfilters wurden entsprechend den Werten der Beispiele im Abschnitt über den Kalmanfilter (siehe Kapitel 4.3.5 auf Seite 42) gesetzt; bei den Experimenten stellte sich heraus, daß die Werte, mit welchen die Kalmanmatrix vorbelegt wurde, nur geringen Einfluß auf den Trackingerfolg hatte.

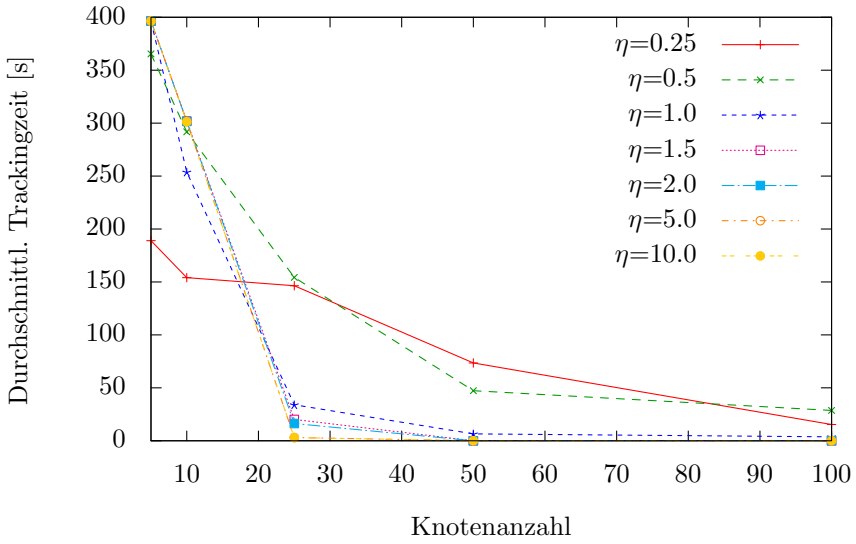

Abbildung 4.7: Vergleich Trackingerfolg und η bei einer Intervallänge von $t_b = 1\,s$ zwischen den Sessions

Die Wahl von η, dem Kriterium für den Signifikanztest, beeinflußt stark die Breite des Baums. Bei Simulationen mit sehr geringen Knotenzahlen erzielten hohe Werte besonders gute, bei größerer Knotendichte hingegen sehr schlechte Ergebnisse, wie in Schaubild 4.7 zu erkennen ist. Als Kompromiß wählten wir für die Intervallänge t_b zwischen den Sessions und einer Knotenzahl n

$$\eta = \begin{cases} \frac{4 \cdot t_b}{n} & falls\ \frac{4 \cdot t_b}{n} > 0,6 \\ 0,6 & sonst \end{cases}$$

Um die Größe des Hypothesenbaums zu begrenzen, wurde nach einer Baumtiefe von 100 das oben beschriebene Multiscan-Verfahren verwendet. Zusätzlich wurde die Breite der Blattebene dadurch begrenzt, daß alle Hypothesen, deren Wahrscheinlichkeit unter $0,1$ % des Durchschnitts der Hypothesenwahrscheinlichkeiten lag, verworfen wurden.

Auch bei den folgenden beiden Parametern gab es Werte, mit denen bei einer sehr kurzen Backoff-Zeit besonders gute Tracking-Resultate erzielten, dafür bei größeren Intervallängen stark einbrachen. Wir wählten wiederum Werte, die bei größeren Zeitintervallen noch ein gutes Ergebnis erbrachten.

Der Parameter „Measurement“ des Simulators gibt den Wert an, mit dem die Hauptdiagonale der Messungs-Kovarianzmatrix R gefüllt wird. Abbildung 4.8 auf Seite 51

zeigt einen Vergleich der Auswirkung verschiedener Parameterwerte. Für die vergleichenden Simulationen wurde der Wert $30,0$ gewählt. Der Simulationsparameter „Disturbance“ füllt die Hauptdiagonale der Kovarianzmatrix Q. Auch hier stellten wir Vergleichsmessungen an (siehe Abbildung 4.9 auf der nächsten Seite). Wir wählten als Kompromiss für einen möglichst guten Erfolg bei verschiedenen Intervallängen den Wert $0,01$.

Die Simulationszeit für jeden Simulationslauf betrug 1000 Sekunden; jeder Simulationslauf wurde mit 10 unterschiedlichen Datensätzen aus dem jeweiligen Bewegungsmodell durchgeführt, um so statistische Ausreißer zu erkennen.

Die Simulationen werden auf einem Feld von 20 x 20 m durchgeführt. Die meisten Veröffentlichungen nutzen einen Simulationsbereich von 100 x 100 m; um jedoch auf einem solch großen Feld eine höhere Dichte an mobilen Knoten zu erreichen, bedarf es einer Knotenzahl, welche den Bedarf an Rechenzeit und Speicher bei der Durchführung des Trackings weit jenseits der realisierbaren Grenzen gesteigert hätte (zum Vergleich: Auf den verwendeten Rechnern benötigte ein einzelner Simulationslauf mit 400 Knoten über zwei Tage Rechenzeit und mehr als 4 GB Arbeitsspeicher).

Um die Übertragbarkeit der Simulationsergebnisse zu untersuchen, führten wir das Tracking von Socialforce-Bewegungsdaten mit Hilfe des MHT auf verschiedenen Feldgrößen durch; dabei blieb die Knotendichte (also Knoten pro Quadratmeter Simulationsfläche) konstant. Die Ergebnisse dieser Simulation in Abbildung 4.10 auf Seite 52 dargestellt: Das Verhalten am Rand der Simulationsumgebung unterscheidet sich von den sonstigen Bewegungen (die Knoten prallen von ihm ab) und stören so die Arbeit des MHT. Mit steigender Feldgröße sinken Einflüsse der Simulationsränder, der durchschnittliche Trackingerfolg steigt an, bis er schließlich konstant bleibt.

An der Grafik erkennt man, daß ab einer Backoff-Zeit von 4 Sekunden der Trackingerfolg ab einer Feldgröße von 20 x 20 m nahezu konstant bleibt. Lediglich bei bei kürzeren Intervallen steigt der Trackingerfolg bei steigender Feldgröße weiter an und stabilisiert sich erst ab 40 x 40 m (Backoff-Zeit von 2 Sekunden) respektive 50 x 50 m (kürzer). Damit können wir annehmen, daß unsere Simulationsergebnisse auf größere Gebiete bei einer Backoff-Zeit ab 4 Sekunden übertragbar sind; bei kürzeren Intervallen muß damit gerechnet werden, daß der Trackingerfolg auf größeren Flächen größer als in unseren Simulationen ist, wie auch aus folgender Tabelle deutlich wird:

Backoff-Zeit	Erfolg 20 m	Durchschn. Erfolg 50 m - 100 m	Rel. Differenz
1 s	141 s	254 s	+80%
2 s	142 s	220 s	+55%
4 s	138 s	142 s	+3%
8 s	51 s	47 s	-9%

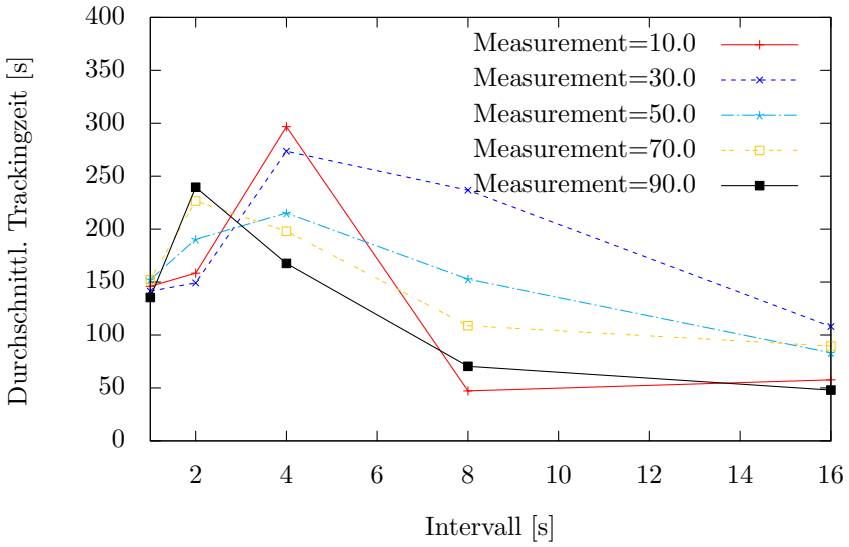

Abbildung 4.8: Vergleich verschiedener Measurement-Parameter (Socialforce-Bewegungsmodell, 10 Knoten)

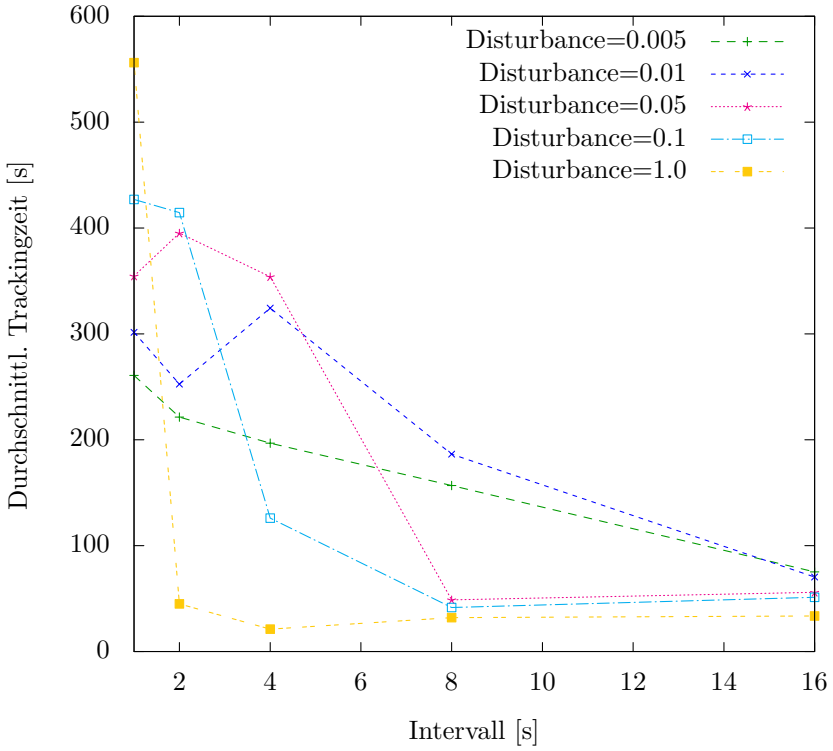

Abbildung 4.9: Vergleich verschiedener Disturbance-Parameter (Socialforce-Bewegungsmodell, 10 Knoten)

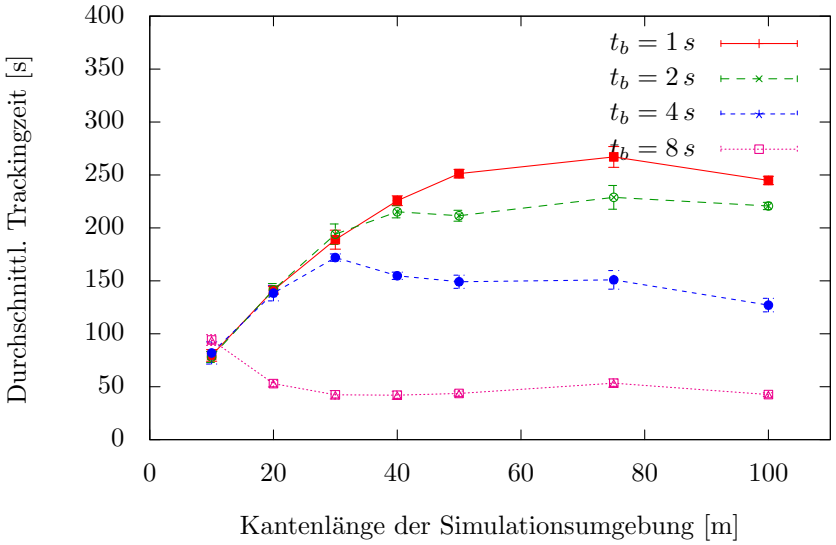

Abbildung 4.10: Vergleich Feldgröße zu Trackingerfolg (MHT und Socialforce-Bewegungsmodell) bei konstanter Knotendichte (6,25 Knoten auf 100 qm)

Auswertung der Simulationen

Ziel der Simulationen war es, die Auswirkungen von Knotenzahl und Länge der Backoff-Zeit auf den Trackingerfolg zu beobachten. Von besonderem Interesse ist natürlich das Ergebnis des MHT beim Verfolgen von realistischen Bewegungspfaden aus dem Socialforce-Modell; eine Übersicht über die Ergebnisse stellt Abbildung 4.11 auf der nächsten Seite dar. Bei der Interpretation des Schaubilds ist zu beachten, daß für sehr kurze Backoff-Zeiten die Übertragbarkeit auf größere Simulationsgebiete nur bedingt gegeben ist – hier werden mitunter deutlich bessere Tracking-Ergebnisse erzielt.

In der Grafik kann man deutlich erkennen, wie der Tracking-Erfolg mit steigender Knotenzahl stark sinkt. Ist das Tracking durch eine hohe Zahl an Knoten bereits stark beeinträchtigt, ist der Einfluß der Backoff-Zeit relativ gering; dies ist aus Privacy-Sicht jedoch unproblematisch, da die Beeinträchtigung durch die hohe Zahl an Knoten bereits für den gewünschten Schutz der Privatsphäre sorgt. Bei geringen Knotenzahlen sieht man jedoch bereits in diesem Übersichtsschaubild sehr deutlich, wie der Trackingerfolg mit steigender Backoff-Zeit fällt.

Noch klarer erkennbar wird dies in zweidimensionalen Schnitten durch das 3D-Schaubild; in den Schaubildern 4.12 bis 4.14 betrachten wir zusätzlich die unterschiedlichen Tracking-Erfolge des Simple Trackers und des MHT. Hier ist deutlich erkennbar, daß die Ergebnisse des MHT durchweg besser als die des Simple Trackers sind. Bei einer kurzen Backoff-Zeit von $b_t = 4\ s$ wird erst ab größeren Knotendichten ein vernünftiger Privacy-Wert erreicht, ab $b_t = 8\ s$ hingegen erlangt man bereits

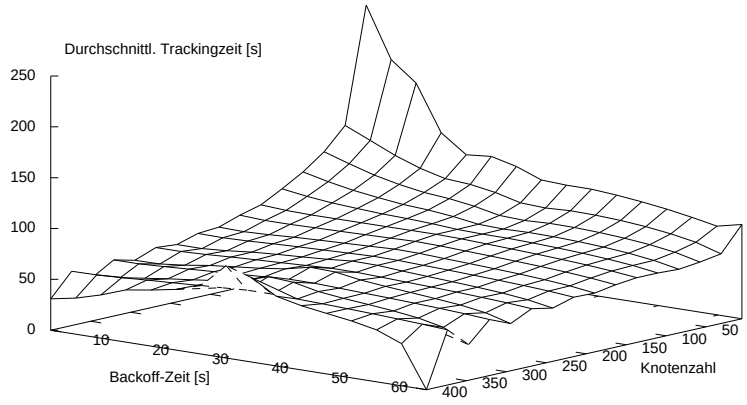

Abbildung 4.11: Socialforce-Bewegungsmodel und MHT

bei geringen Knotenzahlen einen adäquaten Schutz der Privatsphäre.

In Schaubild 4.15 auf Seite 55 vergleichen wir schließlich exemplarisch den Trackingerfolg von MHT bei verschiedenen Bewegungsdaten, die mit unterschiedlichen Bewegungsmodellen generiert wurden (die Ergebnisse mit anderer Knotenanzahl ähneln der hier gezeigten Kurve; wir haben dieses Schaubild ausgewählt, da hier die Knotenanzahl noch gering genug ist, um den Effekt der steigenden Intervallänge nicht zu überlagern).

Bei den Bewegungsdaten des Socialforce-Modells erzielte das MHT-Verfahren das beste Ergebnis – die Bewegungen entsprechen sehr gut der Annahme des Verfahrens, daß sich die Objekte relativ geradlinig bewegen und nur geringe Richtungskorrekturen vornehmen. Beim Random-Waypoint-Bewegungsmodell sind die Wendepunkte das Problem: Die lineare Bewegung zwischen zwei Wegpunkten läßt sich gut verfolgen; beim Erreichen des Wegpunkts kommt es jedoch oft zu einem spitzen Winkel, welcher zum Abriß des Trackingpfads führt. Daher liegen die Trackingergebnisse des MHT meist leicht unter denen des Socialforce-Modells.

Beim Random Walk stimmt die Hypothese des MHT, daß die Objekte vom Grundsatz her eine lineare Bewegung durchführen, nicht mehr; dementsprechend schlecht sind die Trackingergebnisse.

Weitere Nachteile eines realen Angreifers gegenüber Simulation

Wie bereits erwähnt, hat unser simulierter Angreifer eine Reihe von Vorteilen, die in einem realen Szenario nicht (oder allenfalls teilweise) gegeben sind:

- Reale Messungen sind verrauscht und mit Meßtoleranzen behaftet; der Angreifer unserer Simulationen besaß Zugang zu den exakten Positionsdaten.

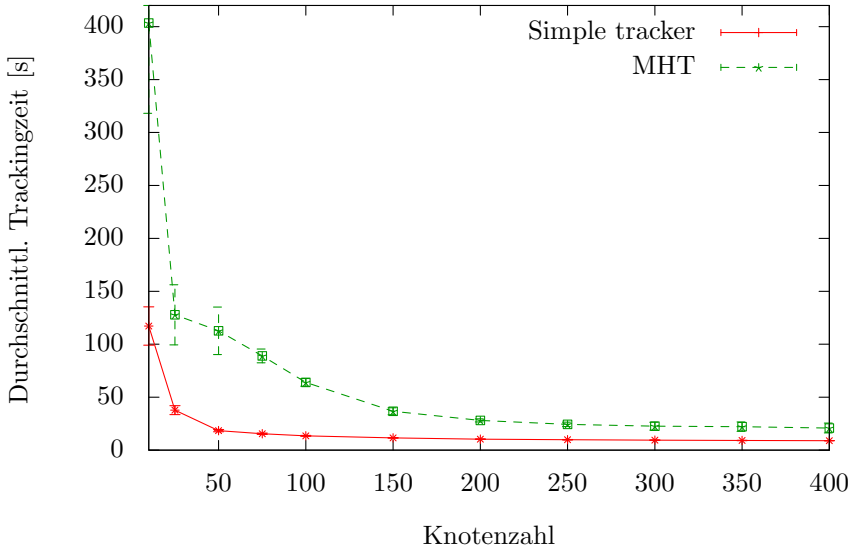

Abbildung 4.12: Vergleich Simple tracker und MHT bei $t_b = 4\,s$

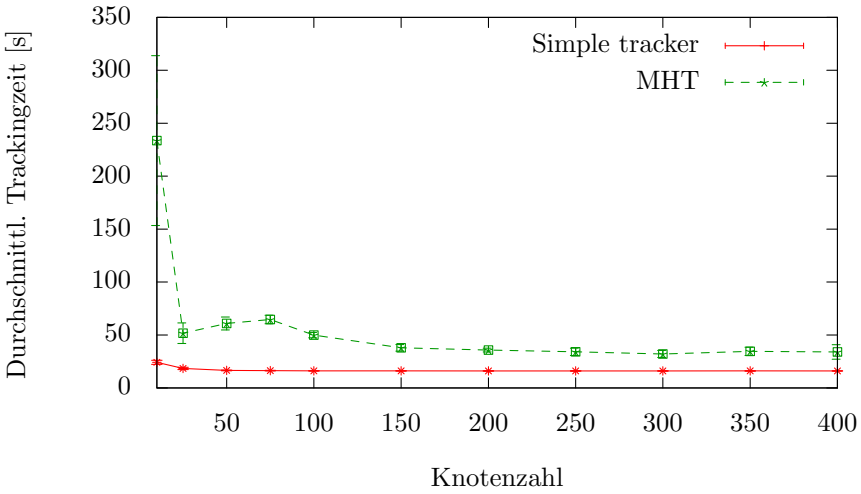

Abbildung 4.13: Vergleich Simple tracker und MHT bei $t_b = 8\,s$

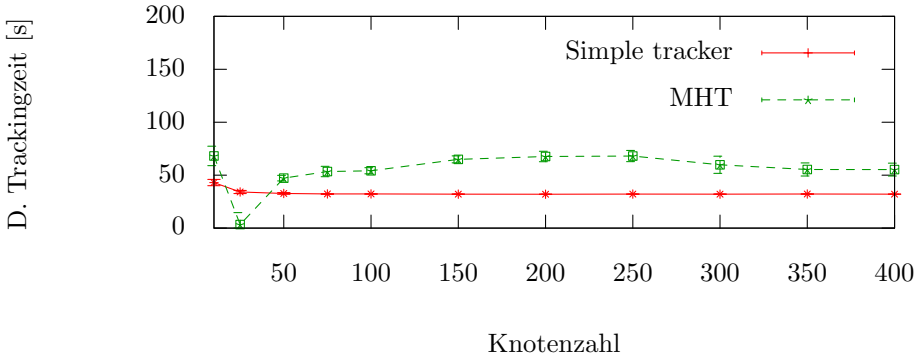

Abbildung 4.14: Vergleich Simple tracker und MHT bei $t_b = 16\,s$

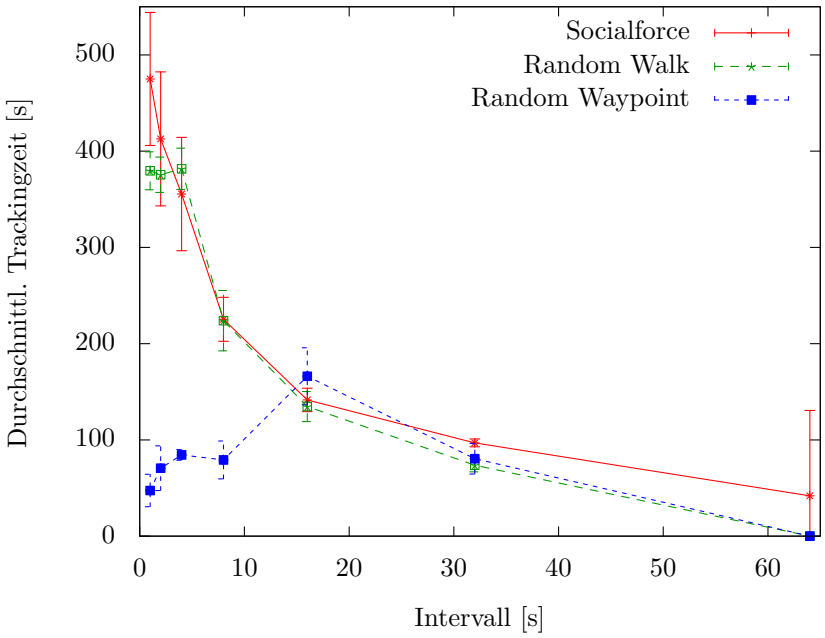

Abbildung 4.15: Vergleich MHT mit verschiedenen Bewegungsmodellen bei 75 Knoten

- Objekte betreten und verlassen den beobachteten Bereich; in unseren Simulationen war die Zahl der Knoten konstant, was bei der Parameterwahl für den MHT ausgenutzt wurde.

- Die technischen Rahmenbedingungen erlauben es nicht, die Position sämtlicher Objekte zu bestimmen; dem simulierten Angriff hingegen standen sämtliche Informationen zur Verfügung.

- Die Trackingergebnisse sollten möglichst zeitnah, wenn nicht sogar in Echtzeit („Online-Angriff“) zur Verfügung stehen. Unsere Simulationen wurden mit sehr großzügigen Begrenzungen der Baumtiefe des MHT gefahren, was sich in sehr hohem Speicherbedarf und langer Rechenzeit niederschlug („Offline-Angriff“).

- Eine Optimierung der Parameter, wie bei uns geschehen, ist nur für einen Offline-Angriff möglich; für zeitnahe Ergebnisse ist der Angreifer auf die geschätzten Parameterwerte festgelegt.

Daher darf man das Ergebnis dieser Simulationen als Worst-Case-Abschätzung und untere Schranke für den Schutz der Privatsphäre betrachten.

4.3.7 Fazit

Der Ansatz, seine Identität hinter wechselnden Pseudonymen zu verbergen, ist nicht neu; ebenfalls bekannt ist, daß eine solche Vorgehensweise keine perfekte Anonymität garantierten kann: Bei einer zu geringen Anzahl an Teilnehmern bei einem solchen Verfahren ist eine Deanonymisierung möglich. Ein Problem, das ebenfalls nicht behoben werden kann, ist das Zusammenführen mit weiteren, leicht zu gewinnenden Informationen: Erhält ein Beobachter beispielsweise über Stunden ein Funksignal aus dem Büro eines Firmenmitarbeiters, so ist mit großer Wahrscheinlichkeit anzunehmen, daß es sich hierbei um diesen Mitarbeiter handelt – ganz unabhängig davon, ob und wie dieser seine Netzwerk-Pseudonyme wechselt.

Viele Arbeiten im Bereich der Anonymisierung in drahtlosen Netzen gehen davon aus, daß die Ortsinformation von den Netzteilnehmern „proaktiv“ geliefert wird, also unter Kontrolle der Netzteilnehmer steht und damit auch von diesen verändert oder verschleiert werden kann. Wir haben in diesem Kapitel den selten diskutierten Fall eines „direkten Angreifers“ betrachtet: Unser Angreifer bestimmt durch technische Mittel den Ursprung einer Funkübertragung; der Netzteilnehmer hat keine Möglichkeit, dies zu verhindern – außer durch Einstellung der Kommunikation.

Die Auswirkungen von Kommunikationspausen mit anschließendem Pseudonymwechsel haben wir in diesem Kapitel mit Hilfe von Simulationen untersucht. Dabei verwendeten wir das Socialforce-Bewegungsmodell, was die Besonderheiten von Fußgängerbewegungen berücksichtigt – dies wurde bis dato in keiner anderen uns bekannten Publikation getan.

Die Auswertungen zeigen, daß ab einer Backoff-Zeit von $t_b = 8\ s$ schon bei einer vergleichsweise geringen Knotendichte der Trackingerfolg so weit absinkt, daß es einem Angreifer zwar möglich ist, vereinzelte Sessions korrekt zusammenzuführen, er jedoch nicht mehr in der Lage ist, einen Knoten durchschnittlich länger als eine Minute zu verfolgen. Da die Annahmen durchweg zu Gunsten des Angreifers gewählt wurden, darf dies als Worst-Case-Abschätzung gelten – im realen Einsatz dürfte der Trackingerfolg eines Angreifers noch geringer sein. Da ein realer Angreifer keine Möglichkeit hat, seine Tracking-Hypothesen zu verifizieren, ist er fast immer auf weitere Informationen angewiesen.

Knoten können also rein durch ihr Kommunikationsverhalten und ohne Verwendung einer Infrastruktur zumindest in vielen Situationen ein Mindestmaß an Privatsphäre generieren. Im folgenden Kapitel erläutern wir, wie sich „befreundete“ Knoten trotz der Verwendung von Pseudonymen wiedererkennen können. Auf die Problematik weiterer Informationen, welche ein Angreifer dem Anwendungsprotokoll entnehmen kann, um so einzelne Sessions zusammenzuführen, wird in Kapitel 6 eingegangen.

5 Erkennung und Schlüsselaustausch

> Bloß weil du nicht paranoid bist, heißt das noch lange nicht, daß sie nicht hinter dir her sind!
>
> *Robert Anton Wilson (1932–2007)*

5.1 Einleitung

Im vorigen Kapitel haben wir erläutert, wie ein mobiler Knoten weitestgehend anonym agieren kann. Um bei einer Kommunikation die Kontrolle über die eigene Privatsphäre nicht aus der Hand zu geben, ist es nötig, sowohl Kontrolle über die übermittelten Daten als auch über die Wahl des Empfängers zu haben. Die Wahl des Empfängers bedeutet nicht zwangsweise dessen Identifizierung, unter Umständen genügt auch das Wissen, es erneut mit demselben Kommunikationspartner zu tun zu haben. Darüberhinaus bedeutet dies aber auch den Ausschluß möglicher weiterer Mithörer, was durch eine entsprechende Verschlüsselung der Kommunikation gewährleistet wird.

Voraussetzung für eine verschlüsselte Verbindung ist jedoch der Besitz von gemeinsamem Schlüsselmaterial. In diesem Kapitel bieten wir zunächst eine Übersicht über verschiedene Möglichkeiten des Schlüsselaustauschs. Unser Beitrag zu diesem Problem ist eine Klassifikation verschiedener Fälle in Abhängigkeit der Anonymitätswünsche der Teilnehmer; ein Teil dieser Fälle lassen sich mit den klassischen Methoden lösen. Für zwei Fälle unserer Klassifikation bieten die gängigen Techniken jedoch keine Standardlösung.

Für einen der beiden Fälle schlagen wir eine Variante eines bekannten Verfahrens vor. Somit verbleibt eine letzte Situation, in der sich zwei Knoten, die sich zwar zuvor schon einmal begegnet sind, nun aber unter anderen Pseudonymen agieren, wiedererkennen sollen. Für diese neue Fragestellung stellen wir ein Protokoll vor, welches das Problem sowohl für paarweise Kommunikation als auch für Gruppenkommunikation effizient löst.

5.2 Vorarbeiten

Für die eigentliche Verschlüsselung gibt es zwei prinzipielle Verfahren: Symmetrische und asymmetrische Chiffren (Definition siehe Anhang A.1 auf Seite 123). Die

bekannten asymmetrische Chiffren haben den Nachteil, daß sie um vielfaches rechenintensiver als symmetrische Chiffren sind. Aus diesem Grund werden sie so weit wie möglich vermieden – man bedient sich eines Hybridansatzes, bei dem über ein asymmetrisches Verfahren ein gemeinsamer Sitzungsschlüssel ausgehandelt wird, der anschließend für eine symmetrische Verschlüsselung benutzt wird. Eine Übersicht über verschiedene moderne Chiffren findet man in [201], Details über die korrekte Modellierung von hybriden Protokollen und die Vor- und Nachteile verschiedener Modi für Blockchiffren werden in [88] ausführlich diskutiert. Neben den erwähnten Büchern bieten Armenia und Morabito [9] eine Übersicht über die klassischen Key-Management-Ansätze (wie beispielsweise der Nutzung einer Public-Key-Infrastruktur) und die spezielle Problematik im Ubiquitous Computing.

Das bekannteste Verfahren zum Erzeugen eines Sitzungsschlüssels ist das Diffie-Hellman-Verfahren [76]. In der Ursprungsform nutzt das Verfahren Potenzen modulo einer Primzahl n: Ein öffentlich bekannter Wert g wird von beiden Parteien mit einer zufällig gewählten Zahl (in $GF(n)$) potenziert; die Ergebnisse werden ausgetauscht und wiederum mit der gewählten Zahl potenziert. Dadurch erhalten beide Parteien dasselbe Ergebnis. Das Mithören der Kommunikation bringt keinen Informationsgewinn (außer durch Lösen des diskreten Logarithmen-Problems), jedoch ist wegen der fehlenden Authentisierung ein Man-in-the-Middle-Angriff möglich.

Abhilfe schafft die Benutzung einer digitalen Signatur wie beispielsweise im Station-to-Station-Protokoll [77]. Hier authentisieren sich die beiden Parteien, indem sie die zur Schlüsselerzeugung übertragenen Werte signiert über den erstellten verschlüsselten Kanal übertragen.

Das Handshake-Protokoll der TLS[1]-Suite [75] benutzt für eine anonyme Verbindung einen einfachen Schlüsselaustausch nach Diffie-Hellman; falls sich die Serverseite mit Hilfe eines Zertifikats identifiziert, erzeugt der Client einen zufälligen Sitzungsschlüssel, verschlüsselt diesen mit dem (vom Zertifikat beglaubigten) öffentlichen Schlüssel des Servers und sendet diese Daten an den Server. Optional kann der Server die Identifizierung des Clients verlangen; der Client sendet hierzu sein Zertifikat zusammen mit dem signierten Hashwert der bisher von ihm gesendeten Daten.

Manche Protokollvarianten verzichten bewußt auf Signaturen und nehmen dafür die Gefahr eines aktiven Angreifers in Kauf: Motivation hierfür kann sein, daß ein entsprechender Angriff sehr aufwendig durchzuführen ist, man den Aufwand einer PKI[2] nicht betreiben möchte, oder aber es keine technische Alternative gibt. Das Standardprotokoll IKE (Internet Key Exchange, [105]) bietet im „identity protection mode“ solche sogenannte *opportunistische Verschlüsselung*.

Eine Alternative zu asymmetrischen Verfahren bietet das Needham-Schroeder-Protokoll [162, 163] (die überarbeitete Fassung entstand nach dem Angriff auf das

[1] TLS: Transport Layer Security. Standardprotokoll zur Aushandlung einer verschlüsselten Übertragung und der Authentisierung der Kommunikationspartner. Nachfolger von SSL.
[2] PKI: Public Key Infrastructure

Protokoll von Denning und Sacco [74]): Hier dient ein Treuhänder, dem alle Parteien vertrauen, als Schlüsselvermittlung. Nachteil ist, daß das Verfahren nur online, also mit Verbindung zum Treuhänder funktioniert. Analoges gilt auch für das bekannte Kerberos-Protokoll [165].

Signaturen erfordern entweder einen vorherigen Austausch von öffentlichen Schlüsseln, oder aber eine gemeinsame Zertifizierungsinstanz, welche die Schlüssel der Kommunikationspartner beglaubigt. Entsprechende Voraussetzungen sind in ubiquitären Systemen und ähnlichen Szenarien meist nicht gegeben, weshalb man hier nach alternativen Ansätzen sucht. Abgesehen von der rein technischen Sicht, gibt es weitere Gründe, weshalb traditionelle Verfahren mitunter weniger geeignet sind: Das Ubiquitous Computing verbindet Computer eng mit dem alltäglichen Leben; ist beispielsweise bei Geschäftsanwendungen die Unabstreitbarkeit (non-repudiation) von Aussagen eine positive Eigenschaft, kann sie im sozialen Bereich für einen Benutzer sehr negative Folgen haben – mit einer einzigen unüberlegten Aussage kann er sich für den Zeitraum der Speicherung „brandmarken“ [32].

Stajano vermeidet in seinen „resurrecting duckling“-Aufsätzen [214, 212] ein drahtloses Aushandeln des Schlüssels komplett; vielmehr geschieht der Schlüsselaustausch während physikalischen Kontakts, was einen Man-in-the-Middle-Angriff ausschließt. Dieses „prägen“ (so genannt in Anlehnung an die Prägung von Küken und Muttertier bei Enten) soll an unauffälligen Stellen geschehen, also an Orten, an denen man den Gegenstand ohnehin aufbewahren würde. Auf diese Weise bleibt der technische Aspekt dem Benutzer verborgen.

Die Einsatzgebiete von RFID ähneln häufig den Szenarien des Ubiquitous Computing; Molnar und Wagner [159] stellen ein Protokoll vor, bei dem sich der RFID-Leser gegenüber einem anonymen RFID-Tag authentisiert. Grundlage für die Authentisierung ist ein gemeinsames Geheimnis, das in einem Challenge-Response-Verfahren verwendet wird. Die anschließende Erstellung eines Sitzungsschlüssels wird in der Arbeit zwar nicht besprochen, dies läßt sich jedoch sehr einfach mit Hilfe des gemeinsamen Schlüssels und den im Protokoll erzeugten Zufallszahlen bewerkstelligen.

Ein ähnlicher Ansatz wird auch in [133] verfolgt; hier wird besonderer Wert auf einfache Rechenoperationen gelegt. Nach jeder erfolgreichen Identifizierung wird der Schlüssel zwischen Leser und Tag gewechselt.

Hoepman [114] diskutiert das Problem des Schlüsselaustauschs für spontane, kurze Sitzungen („Ephemeral Key Exchange“) unabhängig von Besonderheiten der verwendeten Technik. Er zeigt, daß ein sicherer Schlüsselaustausch möglich ist, wenn zusätzlich zum eigentlichen Kommunikationskanal ein weiterer, schmalbandiger Kanal existiert, der entweder authentisch oder privat sein muß. Innerhalb enger Zeitgrenzen wird mit Hilfe des schmalbandigen Kanals ein Schlüssel mit niedriger Entropie erzeugt, der anschließend zum Erzeugen eines starken Schlüssels über den eigentlichen Kommunikationskanal genutzt wird. Beim Generieren des starken

Schlüssels wird das Prinzip der „Perfect Forward Secrecy“ eingehalten, d. h. ein späteres Brechen des schwachen Schlüssels kompromittiert nicht den erzeugten zweiten, stärkeren Schlüssel. Eine Erweiterung des Protokolls für den Einsatz in anonymen Broadcast-Netzen wurde in [115] veröffentlicht.

Je nach Szenario bestehen verschiedene Möglichkeiten, einen solchen schmalbandingen Kanal herzustellen; in [114] dient hierfür die Eingabe einer Zahlenkombination. Andere Anwendungsgebiete erlauben hier möglicherweise weitere Automatismen: In [228] wird zur Vernetzung von implantierten Körpersensoren ein gemeinsam verfügbares biometrisches Merkmal (wie beispielsweise die Zeitintervalle zwischen zwei Herzschlägen) genutzt, um einen Sitzungsschlüssel zu generieren.

Balfanz et al. [15] bezeichnen die Nutzung eines zweiten Kanals (entweder drahtlos wie in [114] oder durch Berührung [214]) als *pre-authentication*. Sie zeigen, daß ein sicherer Schlüsselaustausch möglich ist, falls ein lokationslimitierter Kanal für die Pre-Authentisierung gegeben ist. Ähnlich wie in [114, 115] wird auch hier argumentiert, daß ein authentischer Kanal ausreichend ist. Durch die Lokationsbegrenzung wird zum einen die Auswahl des Kommunikationspartners getroffen (die Frage, mit wem der Benutzer redet, beantwortet er auf diese Weise selbst), außerdem sind Angriffe auf die verbleibende Schwachstelle des Ansatzes – aktive Angriffe auf den lokationslimiterten Kanal – nur schwer zu implementieren.

Das „Secret Handshake“-Protokoll von Balfanz et al. [14] erlaubt zwei Partnern zu erkennen, ob sie derselben Gruppe angehören; das Protokoll ist so gestaltet, daß Nichtgruppenmitglieder keine Information über die Gruppenzugehörigkeit ihres Gegenübers erlangen. Bei einer erfolgreichen Erkennung entsteht automatisch ein Sitzungsschlüssel. Allerdings benutzen die Parteien immer dieselben Pseudonyme, so daß zwar Außenstehenden die Gruppenzugehörigkeit verborgen bleibt, die Personen anhand der Pseudonyme jedoch eindeutig identifizierbar sind.

Ein entfernt verwandtes Problem betrachten Bussard et al. [42]: Sie widmen sich der Problematik, wie ein Benutzer die Echtheit eines Dokuments bestätigen kann, ohne seine Anoymität aufzugeben. Hierzu werden Dokumente nicht mit der eigenen Identität signiert, sondern mit Zertifikaten, die man in der Vergangenheit gesammelt hat; diese Zertifikate belegen, daß man beispielsweise zu einer bestimmten Zeit an einem bestimmten Ort war, oder sie stellen Bestätigungen für erfolgreiche Transaktionen dar. Um diese Zertifikate zu erlangen, wird ein Wissensbeweis durchgeführt: Der Benutzer beweist dem Aussteller des Zertifikats, daß er in Besitz eines Schlüsselpaars ist, das von einer vertrauenswürdigen Instanz signiert wurde – ohne den Schlüssel selbst dabei preiszugeben. Hierzu verwenden die Autoren eine modifizierte Variante von Camenischs Gruppensignatur [46, 11].

Die Konsequenzen des Erkennens bzw. Wiedererkennens eines Knotens beschreiben Seigneur et. al. [204, 205]: Neben der Authentisierung ergeben sich weiterreichende Folgen für das gegenseitige Vertrauen; die Autoren nehmen zwar an, daß die Geräte in einer ubiquitären Umgebung einander eine gewisse Menge an Grundvertrauen

entgegenbringen, jedoch bildet das Wiedererkennen die Grundlage dafür, daß sich die jeweilige Vertrauensbeziehung verändern und weiterentwickeln kann. Die Autoren diskutieren ebenfalls die Trade-Off-Beziehung zwischen Trust und Privacy [206].

5.3 Bewertung

Im Kontext Sicherheit und Ubiquitous Computing wird das Problem des Schlüsselaustauschs bereits seit längerem diskutiert. Die klassischen Verfahren, die eine interaktive vertrauenswürdige Instanz (wie beispielsweise beim Needham-Schroeder-Protokoll) benötigen, scheiden in nahezu allen Betrachtungen aus.

Alle weiteren Ansätze sind abhängig vom Anwendungsszenario, es gilt, eine Abwägung zwischen Sicherheit, Infrastruktur und verschiedenen Arten der Benutzerinteraktion abzuwägen.

Steht ausreichend Rechenzeit für Public-Key-Operationen und die Möglichkeit, Zertifikate vertrauenswürdiger Instanzen zu verteilen, zur Verfügung, so stellen die klassischen Ansätze wie TLS eine brauchbare und praxiserprobte Lösung dar. Allerdings betrachten diese nicht das Problem der Privatsphäre, mindestens eine der beiden Parteien präsentiert bei diesen Protokollen ihre Identität.

Die Ansätze mit Benutzerinteraktion lösen das Schlüsselaustauschproblem ohne eine zentrale Instanz. Das Vertrauen, mit dem richtigen Gesprächspartner zu kommunizieren, wird durch die Benutzeraktion hergestellt: Durch Berührung [214] oder eine gerichtete Verbindung mit dem Zielgerät [15] wird ein Man-in-the-Middle-Angriff weitestgehend ausgeschlossen (wobei es auch hier schon Beispielsangriffe gab, beispielsweise bei Verwendung von RFID-Technik durch Aufkleben eines extrem dünnen Antennenpaars auf den Leser [137]). Der Schlüsselaustausch von Hoepman erfordert nicht nur die Interaktion des Benutzers, sondern außerdem entsprechende Ein-/Ausgabemöglichkeiten am Gerät des Benutzers.

Die Arbeit von Molnar und Wagner dient nur der Wiedererkennung und im von den Autoren geschilderten Szenario der Authentisierung des Lesers: Zwar wird hier auf asymmetrische Kryptographie verzichtet, jedoch wird für die Authentisierung ein vorher ausgetauschtes gemeinsames Geheimnis benötigt. Das Protokoll ist nur für die Authentisierung eines einzelnen Lesers (oder Gruppe von Lesern mit denselben Schlüsseln) ausgelegt.

5.4 Verschiedene Verbindungsarten

In den vorigen Abschnitten haben wir einen Überblick über vorhandene Arbeiten gegeben. Wie wir gesehen haben, lösen einige der Arbeiten nur Teile der hier

gestellten Forderungen nach Sicherheit, Anonymität und möglichst geringer Benutzerinteraktion.

Es ist in der Kryptographie jedoch eine goldene Regel, möglichst auf bewährte Verfahren, die ihre Sicherheit in der Praxis erwiesen haben, aufzubauen. Daher stellen wir in diesem Abschnitt eine Fallunterscheidung vor, um so an möglichst vielen Stellen von bekannten Verfahren profitieren zu können. In den folgenden Abschnitten stellen wir für die verbleibenden Situationen angepaßte bzw. neue Lösungen vor.

Die Forderung nach einer gesicherten, anonymen Kommunikation ist in Bezug auf aktive Angriffe eine Gratwanderung: Bei vollständiger Anonymität gibt es keinerlei Hinweis auf die Identität des Kommunikationspartners; ein aktiver Angreifer, der sich zwischen zwei Kommunikationspartner setzt, ist somit nicht erkennbar – eine vollkommen anonyme Kommunikation ist schließlich eine Kommunikation mit „irgendjemand“. Üblicherweise gibt es irgendein Kriterium, anhand dessen man seinen Kommunikationspartner auswählen möchte; die Schwierigkeit besteht nun darin, dieses Kriterium geeignet zu modellieren.

Wie in den Vorarbeiten deutlich wird, läßt sich ein Man-in-the-Middle-Angriff nur durch ein geeignetes identifizierendes Merkmal ausschließen. Dies kann ebenfalls auf kryptographischem Wege mit Hilfe von Signaturen und Zertifikaten geschehen, oder aber mit Hilfe eines geeigneten zweiten Kanals ([115] zeigt, daß entweder ein authentischer oder ein sicherer Kanal benötigt wird). Als Beispiele hierfür seien gerichtete drahtlose Verbindungen (wie z. B. mittels IrDA), Übertragung bei physikalischem Kontakt oder aber Einmal-Codes (z. B. auf einen Zettel gedruckt, ähnlich einer Aufruf-Nummer in einem Wartesaal) genannt. Allen Alternativen zu Zertifikaten ist gemeinsam, daß sie die Interaktion des Benutzers benötigen.

Um die Benutzerinteraktion möglichst selten in Anspruch zu nehmen, gliedern wir in verschiedene Verbindungsarten und Verbindungssituationen:

1. Übertragung unverschlüsselter Informationen, wie z. B. das Announcement eines Dienstes
2. Verschlüsselte Kommunikation, bei der sich beide Parteien identifizieren
3. Verschlüsselte Kommunikation, bei der sich einer der Partner identifiziert
4. Unmittelbare Fortsetzung einer zuvorgehenden verschlüsselten Kommunikation
5. Kommunikation zwischen zwei anonymen Parteien:
   a) Erstmalige Kommunikation
   b) Erneute Kommunikation
   c) Kommunikation unter Inkaufnahme eines Man-in-the-Middle

Neben dem Trivialfall 1 lassen sich die Fälle 2 und 3 direkt mit bekannten Protokollen lösen: TLS ist genau für diese Situationen in HTTPS[3] seit langem im Einsatz und authentisiert so Webseiten gegenüber dem Betrachter (und optional auch den Betrachter gegenüber dem Webserver). Sind die entsprechenden Zertifikate vorhanden, bedarf es hier keiner weiteren Benutzerinteraktion, um eine sichere Verbindung zu erstellen.

Die gesicherte erstmalige Kommunikation zwischen anonymen Stellen aus Fall 5a wurde in [114, 115] diskutiert: Hier wird eine Benutzerinteraktion zur Authentisierung des Kanals benötigt. Der Fall 5c ist ebenfalls mit dem bereits vorgestellten Diffie-Hellman-Schlüsselaustausch lösbar.

Je nach Situation ist auch eine spätere Authentisierung sinnvoll: Der Kommunikationsverlauf kann an einen Punkt gelangen, ab dem der Ausschluß eines Man-in-the-Middle und die gegenseitige Identifizierung vonnöten sind (beispielsweise nach dem Betrachten eines Ladenangebots, wenn der Kunde tatsächlich kaufen möchte und Bezahlinformationen übertragen werden müssen). Für diesen Fall (der natürlich auch für 2 und 3 verwendet werden kann) stellen wir in 5.5 eine Variante des Station-to-Station-Protokolls vor.

Das Wiederaufnehmen einer vorhergehenden Kommunikation (Fall 4) kann entweder über einen einfachen Mechanismus mit Hilfe eines Session-Identifiers gelöst werden; oder aber man bedient sich desselben Verfahrens, das auch für den verbleibenden Fall 5b verwendet wird: Auch beim Wiedererkennen anonymer Knoten wird eine früher ausgehandelte Verschlüsselung wieder aufgenommen. Da dieses Problem in der Literatur nach unserem Wissen noch nicht diskutiert wurde, haben wir hierfür ein neues Protokoll entworfen. Dieses stellen wir in Kapitel 5.6 auf der nächsten Seite vor.

5.5 Verzögertes Station-to-Station-Protokoll

Eine Möglichkeit, die Identifikation bei Bedarf durchzuführen, bietet eine Variante des Station-to-Station-Protokolls [77]: Wir teilen das Protokoll in zwei Schritte, den Schlüsselaustausch- und den Authentifizierungsschritt.

Der Schlüsselaustausch geschieht mit Hilfe des Diffie-Hellman-Protokolls; zu einer bekannten Primzahl n und einer Basis g wählen die Kommunikationspartner A und B jeweils eine Zufallszahl r_a bzw. r_b. Es werden nun folgende Nachrichten ausgetauscht:

$$A \to B: \; X = g^{r_a} \pmod{n} \tag{1.1}$$

$$B \to A: \; Y = g^{r_b} \pmod{n} \tag{1.2}$$

[3] HTTPS: Hypertext Transport Protocol Secured: HTTP, abgesichert mit SSL/TLS [75]

Durch Berechnung von $X^{r_b} \bmod n$ bzw. $Y^{r_a} \bmod n$ erhalten A und B den gemeinsamen Schlüssel $k = g^{r_a r_b} \bmod n$. In der Originalvariante des Station-to-Station-Protokolls folgt nun direkt der Authentifizierungsschritt; hierbei werden die ersten Signaturen der ersten beiden Nachrichten ausgetauscht:

$$A \rightarrow B : E_k(Cert_A, Sig_A(X, Y)) \tag{1.3}$$

$$B \rightarrow A : E_k(Cert_B, Sig_B(X, Y)) \tag{1.4}$$

Die Zertifikate $Cert$ beinhalten neben Informationen über die Identität den jeweiligen öffentlichen Schlüssel sowie eine Signatur einer Zertifizierungsinstanz, welche die Identität des Schlüsselinhabers beglaubigt.

Üblicherweise vermeidet man aus Performancegründen dieses Protokoll: Es erfordert bei jeder Partei fünf Public-Key-artige Operationen: Das Berechnen des Schlüsselteils X bzw. Y, die Bestimmung des Diffie-Hellman-Schlüssels, das Signieren von X, Y sowie das Überprüfen von Zertifikat und Signatur des Gegenübers im Authentisierungsschritt.

Ein Vorteil besteht in der Möglichkeit, die beiden Protokollschritte aufzutrennen. Der Authentisierungsschritt wird dann verzögert, bei Bedarf durchgeführt: Der Zeitpunkt stellt den Trade-Off zwischen Privacy- und Sicherheitsanforderungen dar.

Die Authentisierung kann zunächst auch nur einseitig erfolgen: So könnte beispielsweise ein anonymer Kunde A von einem Diensteanbieter B dessen Authentisierung von vornherein fordern; in diesem Fall würde B aufgefordert, nach der Generierung des Schlüssels die Nachricht 1.4 zu senden. Angenommen, B würde in einem Katalog verschiedene Artikel zum Verkauf anbieten; für A ist die Identität von B von Interesse, da er von vornherein keinem gefälschten Lockangebot aufsitzen möchte. Für B besteht zu diesem Zeitpunkt noch keine Veranlassung, die Identität von A zu bestimmen. Wenn sich A in diesem Beispiel schließlich zum Kauf eines Artikels entscheiden würde, könnte B erst zu diesem Zeitpunkt aus Sicherheitsgründen die Authentisierung von A fordern.

5.6 Wiedererkennung anonymer Knoten

In diesem Kapitel stellen wir ein Verfahren vor, mit dessen Hilfe zwei Knoten sich auch nach einem Identitätswechsel wiedererkennen können. Das Verfahren wurde bereits in [192] diskutiert und in verbesserter Fassung in [194] publiziert.

Ziel ist es, ein digitales Abbild des Wiedererkennens einer bekannten Person zu erreichen. Als Analogie soll hier ein Aufenthalt in einer dichten Menschenmenge dienen: Personen, die man nicht kennt, sind schlicht „irgendwelche“ (anonyme) Personen; einen Bekannten wird man jedoch wiedererkennen – an Merkmalen, die zwar für jedermann sichtbar sind, aber nur bei dessen Bekannten die gewünschte Assoziation hervorrufen. Ein direktes Anwendungsbeispiel für ein solches Szenario zur

Wiedererkennung ist das SmartReminder-Projekt aus unserer Arbeitsgruppe [132], bei dem ein digitaler Assistent beim Treffen anderer Personen an Aufgaben oder ausstehende Termine erinnern soll.

Das digitale Pendant soll noch einigen härteren Kriterien genügen: Im obigen Beispiel ist es möglich, die Merkmale einer anonymen Person beispielsweise durch ein Foto festzuhalten und sie daran wiederzuerkennen. Dies soll im hier vorgestellten Protokoll nicht möglich sein. Auch soll es möglich sein, nicht mehr gewünschte Bekanntschaftsbeziehungen aus eigener Initiative heraus aufzulösen.

Der Ansatz der Wiedererkennung bietet mehrere Vorzüge: Zunächst vermeidet das vorgestellte Verfahren die Verwendung von Public-Key-Operationen, welche sehr rechenaufwendig und auf Kleingeräten dementsprechend zeit- und energieintensiv sind. Des weiteren geschieht die Identifizierung paarweise, was bedeutet, daß keiner der Knoten hierdurch an eine einzelne Identität gebunden ist: Die „Identität“ ergibt sich aus den weiteren Kontextinformationen, welche die Knoten mit ihrem Gegenüber assoziieren – im Minimalfall ist dies lediglich die Information, diesem Knoten schon einmal begegnet zu sein.

5.6.1 Voraussetzungen und Ziele

Wir gehen davon aus, daß die Knoten bei ihrer vorherigen Begegnung über einen sicheren Kanal Informationen austauschen konnten. Diese Informationen sind die Grundlage für die Wiedererkennung. Diese Information bezeichnen wir im folgenden als *Sitzungsinformation*, die vorhergehende Kommunikation wird als *Sitzung* bezeichnet.

Des weiteren benötigen die Knoten einen persistenten Speicher, um diese Informationen vorzuhalten; die Größe des Speichers ist dem Szenario anzupassen. Für den Fall, daß der Speicher nicht ausreicht, muß eine geeignete Verdrängungsstrategie definiert sein. Das Ziel des Verfahrens läßt sich wie folgt definieren:

- Zwei Knoten, die bei einer vorherigen Begegnung Sitzungsinformationen ausgetauscht haben (und diese noch nicht verdrängt haben), sollen diese in ihrem Pufferspeicher identifizieren können.
- Das jeweilige Gegenüber darf keinerlei Information darüber erhalten, mit welchen weiteren Knoten Sitzungsinformationen ausgetauscht wurden. Das bedeutet auch, daß im Falle eines Nicht-Erkennens keinerlei Information über die Identität des Gegenübers bekannt werden darf.
- Durch Abhören der Kommunikation darf keine Information über die Identität der beteiligten Knoten (oder ihre Beziehung zueinander) bekannt werden.
- Wie bei klassischen Identifizierungsverfahren [216, Seite 283f] gilt auch hier, daß weder durch Abhören der Daten noch durch das Benutzen übertragener

Informationen eines früheren Protokollaufs die Möglichkeit bestehen darf, eine andere Identität vorzutäuschen.

5.6.2 Angriffsszenarien

Zu den in der Literatur allgemein anerkannten Eigenschaften von Sicherheitsarchitekturen (eine Beschreibung und formale Definition findet man bei [195, 196]) zählen Vertraulichkeit (confidentiality), Authentizität (authenticity), Integrität (integrity) und Verfügbarkeit (availability); je nach Autor werden hier auch Verbindlichkeit (accountability oder non-repudiation) und Zugriffskontrolle (access control) mit aufgeführt. Für die Wiedererkennung von Knoten sind Angriffe auf einige dieser Eigenschaften zu betrachten:

Verfügbarkeit: Der Angreifer kann versuchen, den Ablauf des Protokolls in einer Art und Weise zu stören, daß seine Funktion nicht mehr gewährleistet ist. Da Funkmedien ohnehin deutlich störanfälliger als kabelgebundene Netze sind, sind hier vor allem Angriffe von Interesse, die mit geringem Aufwand des Angreifers große Auswirkungen produzieren. Das Blockieren des Funkmediums z. B. durch nicht standardkonformen Medienzugriff oder den Einsatz von Störsendern ist protokollunabhängig immer möglich.

Identifizierung: Angreifer können darauf abzielen, entweder in einer laufenden oder einer späteren Verbindung als eine der beiden Parteien aufzutreten (Impersonation). Im Falle einer laufenden Verbindung könnte dies mit Hilfe eines „Man-in-the-Middle"-Angriffs realisiert werden, im verzögerten Fall sind Replay-Versuche hierfür in Betracht zu ziehen.

Anonymität: Angriffe auf die Anonymität der Knoten versuchen, einen Knoten zu einem späteren Zeitpunkt wiederzuerkennen. Dies kann mit passiven oder aktiven Mitteln geschehen: Im ersteren Fall würden charakteristische Merkmale im Protokoll die Anonymität gefährden, so daß ein Angreifer durch bloßes Abhören der Kommunikation einen Knoten wiedererkennen könnte. Im Aktiven Fall kann ein Angreifer (selbst generierte oder früher abgehörte) Pakete senden („Probing") und anhand bestimmter Reaktionen des Knotens Rückschlüsse auf dessen Identität ziehen.

Wir nehmen im folgenden einen Angreifer gemäß des Dolev-Yao-Modells [79] an. Von einem solchen Angreifer wird angenommen, daß er das gesamte Netz um einen Knoten herum beherrschen kann. Dies bedeutet, daß er nach Belieben Pakete verzögern, löschen, verändern oder erzeugen kann. Eine direkte Konsequenz daraus ist, daß er auch beliebig viele (nicht existente) Knoten simulieren kann, solange deren Identität nicht durch irgendeine Art Authentisierung bestätigt werden muß.

5.6.3 Einfaches Wiedererkennungsprotokoll

Im folgenden erklären wir zunächst die einfache Variante des Protokolls, um daran die Funktionsweise zu erläutern. In den darauffolgenden Abschnitten werden dann Performance-Verbesserungen eingeführt. Folgende Notation wird zur Beschreibung der Protokollvarianten benutzt:

- $E_k(v)$ ist die symmetrische Verschlüsselung der Daten v mit dem Schlüssel k.
- n steht für eine Nonce (siehe Anhang A.1.7). Zusätzlich zu den normalen Nonce-Eigenschaften fordern wir, daß die Werte von n nicht vorhersagbar sind. In der Praxis wird n daher mit einer Zufallszahl aus einem ausreichend großen Wertebereich gewählt; alternativ kann der im Anhang skizzierte verschlüsselte Zähler eingesetzt werden.
- T steht für ein Sitzungstoken, welches zur eindeutigen Identifizierung einer Sitzung bei allen beteiligten Parteien dient.
- *decoy* steht für eine Zufallszahl mit derselben Länge wie ein Block der Chiffre E. Da das Chiffrat einer idealen Chiffre von weißem Rauschen nicht unterscheidbar ist, dient dieser Wert der Verschleierung der Kommunikation gegenüber Mithörern.
- A und B bezeichnen die beiden beteiligten Parteien. A bezeichnet den Knoten, der das Protokoll initiiert. A hat zu Beginn keine Ahnung, mit wem er kommuniziert; analog gilt selbes für B.

Wir spezifizieren absichtlich keine Algorithmen und Bitlängen – diese sind je nach Szenario passend zu wählen. Wir gehen davon aus, daß die Zufallszahlen und Chiffren ideale Eigenschaften (siehe Anhang A.1 auf Seite 123) besitzen. Die Bitlängen von zufällig gewählten Elementen sind ausreichend groß zu wählen, daß die Wahrscheinlichkeit von Kollisionen im entsprechenden Szenario vernachlässigbar klein ist.

Für eine gemeinsame Sitzung zwischen zwei Knoten A und B werden bei beiden Knoten folgende Sitzungsinformationen gespeichert:

- $T_{A,B}$: Das Sitzungstoken muß sowohl bei A als auch bei B die Sitzung eindeutig identifizieren; es ist der Primärschlüssel für die Liste der Sitzungen.
- $k_{A,B}$: Das zugehörige Schlüsselmaterial.
- Weitere, mit der Sitzung assoziierte Informationen, wie beispielsweise Daten über die Identität des Gegenübers, dessen Zugriffsrechte auf eigene Daten oder Dienste oder Informationen über bereits ausgetauschte privacy-relevante Daten (welche später bei der Kontrolle des Informationsflusses berücksichtigt werden sollen – siehe Kapitel 6.4).

Das Protokoll basiert auf der Annahme, daß für eine Sitzung $T_{A,B}$ nur B in der Lage ist, Daten, die mit $k_{A,B}$ verschlüsselt sind, zu dechiffrieren. B kann eine erfolgreiche Dechiffrierung daran erkennen, daß diese ein bekanntes Resultat ergibt – hierfür wird $T_{A,B}$ verwendet.

Das Protokoll wird vom Knoten A gestartet. Da A zunächst nicht weiß, mit wem er kommuniziert, iteriert A über sämtliche gespeicherten Sitzungsinformationen:

$$A \rightarrow B : \forall B : \; E_{k_{A,B}}(T_{A,B}, n_1) \tag{2.1}$$

B wiederum versucht nun, jedes der empfangenen Pakete mit jeder bei ihm gespeicherten Sitzung zu entschlüsseln. Eine erfolgreiche Entschlüsselung erkennt B daran, daß die entschlüsselten Daten das mit dem Schlüssel korrespondierende Sitzungstoken $T_{A,B}$ enthalten. Im Falle eines solchen Erfolges sendet B das Paket 2.2:

$$B \rightarrow A : E_{k_{A,B}}(T_{A,B}, n_1 + 1, n_2) \tag{2.2}$$

Ansonsten generiert B eine Dummy-Nachricht 2.3 mit einem neuen *decoy*-Wert (wie auf der vorherigen Seite beschrieben):

$$B \rightarrow A : decoy \tag{2.3}$$

A kann eine Dummynachricht von einer Erfolgsbestätigung unterscheiden, indem A das Paket entschlüsselt; enthält das Ergebnis die (von A generierte) Nonce n_1+1, so war die Erkennung erfolgreich, ansonsten handelte es sich um eine Dummynachricht.

Um die Frische des Protokolls zu gewährleisten, muß A nach einer erfolgreichen Erkennung die Nonce von B beantworten:

$$A \rightarrow B : E_{k_{A,B}}(T_{A,B}, n_2 + 1, data) \tag{2.4}$$

Im Schritt 2.4 kann bereits mit der weiteren Kommunikation begonnen werden, was hier mit *data* angedeutet ist.

Beispiel

Der Ablauf des Protokolls soll am folgenden kleinen Beispiel exemplarisch illustriert werden. Zwei Knoten, Knoten 1 und Knoten 2, begegnen einander und möchten feststellen, ob sie sich bereits kennen[4]. Folgende Sitzungsinformationen seien bei den Knoten gespeichert:

Knoten 1 initiiert das Protokoll. Da beide Knoten nur unter zufällig gewählten Pseudonymen im Netz sichtbar sind, beginnt Knoten 1, die Liste seiner Sitzungsinformationen durchzuprobieren (Protokollschritt 2.1). Beginnend mit dem ersten Eintrag schickt Knoten 1 also an Knoten 2:

[4] Die Knoten wurden bewußt 1 und 2 genannt, um eine Verwechslung mit A und B in den Bezeichnern $T_{A,B}$ und $k_{A,B}$ zu vermeiden. Der Leser sollte im Hinterkopf behalten, daß weder Knoten 1 noch Knoten 2 zu Beginn des Protokolls weiß, wer sein Gegenüber ist (oder ob er sein Gegenüber überhaupt kennt).

$T_{A,B}$	88	42	7	50
$k_{A,B}$	k_{88}	k_{42}	k_7	k_{50}

Sitzungsinformationen Knoten 1

$T_{A,B}$	42	9	17	29
$k_{A,B}$	k_{42}	k_9	k_{17}	k_{29}

Sitzungsinformationen Knoten 2

Abbildung 5.1: Sitzungsinformationen der Knoten

$$E_{k_{88}}(88, n_1)$$

Knoten 2 versucht nun, das erhaltene Paket mit allen ihm bekannten Schlüsseln (also $k_{42}, k_9, \ldots$) zu entschlüsseln. Da der passende Schlüssel nicht in der Liste ist, sendet Knoten 2 eine *decoy*-Nachricht (Nachricht 2.3).

Knoten 1 überprüft nun, ob es sich bei den empfangenen Daten um eine Bestätigung handelt, indem er versucht, das Paket mit dem Schlüssel k_{88} zu entschlüsseln. Im Erfolgsfall muß das Paket die von ihm gesendete Nonce n_1 enthalten, was hier aber nicht der Fall ist. Daher fährt Knoten 1 mit dem nächsten Eintrag seiner Sitzungsliste fort:

$$E_{k_{42}}(42, n_1)$$

Nun hat Knoten 2 mit dem ersten Eintrag seiner Liste Erfolg: Die Entschlüsselung mit k_{42} ergibt 42, was laut dem Eintrag in der Sitzungsliste der zu k_{42} korrespondierende Wert ist. Damit hat Knoten 2 die passende Sitzung eindeutig erkannt. Als Bestätigung schickt Knoten 2 das Paket 2.2:

$$E_{k_{42}}(42, n_1 + 1, n_2)$$

Knoten 1 entschlüsselt das Paket mit k_{42} und erkennt die korrekte Operation auf der Nonce n_1, die er in der Anfrage mitgeschickt hatte. Damit hat Knoten 1 die Bestätigung, daß sein Gegenüber den passenden Schlüssel für diese gespeicherte Sitzung besitzt und diese somit eindeutig erkannt.

Beobachtbare Merkmale

Auch über komplett verschlüsselte Kommunikationen können durch Beobachtung von Seitenkanälen verschiedene Aussagen getroffen werden [181]: Länge und Anzahl der Pakete, Verarbeitungsdauer oder beobachtbare Reaktionen des Benutzers (wie z. B. der Abbruch der Kommunikation) können einem Angreifer Indizien liefern. In diesem Abschnitt diskutieren wir Merkmale, die beim oben vorgestellten Protokoll beobachtet werden können und wie dies (und zu welchem Preis) vermieden werden kann.

Für allgemeine Techniken der Verschleierung wie Padding (Auffüllen der Pakete auf gleiche Größe) oder das verzögerte Senden verweisen wir auf Arbeiten aus dem Mix-Kontext ([52, 24, 242]).

Die ersten beiden Merkmale haben deutliche Auswirkung auf die Verfolgbarkeit von Knoten, wohingegen die letzten Merkmale nur einen recht geringen Informationsgewinn erlauben.

Sofortiges Protokollende nach der Erkennung: Wenn für einen Beobachter das Ende des Erkennungsversuchs sichtbar ist (z. B. durch andere Paketlängen, Headerinformationen im Klartext, etc.), kann der Angreifer daraus die Zahl der vergeblichen Versuche vor der Erkennung ableiten. Je nach Implementierung ist dies die exakte Anzahl, oder zumindest eine Schätzung (werden beispielsweise k Anfragen 2.1 in einem Netzwerkpaket übertragen, so kann der Angreifer die Position auf k Versuche genau schätzen). Werden die Anfragen immer in derselben Reihenfolge gesendet, ist bei zwei Knoten die Zahl der Versuche bis zur Wiedererkennung ebenfalls immer identisch.

Diese Information kann sich ein Beobachter zu Nutze machen: Es seien n Knoten mit einer Listenlänge von l Einträgen unter Beobachtung; die Position i in der Liste bei der ersten Beobachtung nehmen wir als gleichverteilt an. Beobachtet der Angreifer nun eine weitere erfolgreiche Erkennung nach i Protokollschritten, so kann er mit einer Wahrscheinlichkeit von mindestens $P = (1-\frac{1}{l})^{(n-2)(n-1)}$ davon ausgehen, daß es sich um das Knotenpaar der ersten Beobachtung handelt[5]. Ist n ausreichend klein, ist also eine hohe Erkennungsrate möglich; kleine Werte für n können beispielsweise bei lokal begrenzten Beobachtungen wie z. B. beim Erstellen eines Bewegungsprofils in einem abgegrenzten Gebiet angenommen werden.

Um diesen Angriff zu vermeiden, gibt es zwei Lösungsmöglichkeiten: Entweder wird das Protokoll (mit Dummy-Daten) bis zum Listenende vollends durchgeführt (wobei auf eventuelle Seitenkanäle durch Timing zu achten ist), oder aber die Reihenfolge, in der die Anfragen gesendet werden, wird bei jedem Protokollauf zufällig verändert.

Länge der Session-Liste: Ist die aktuelle Länge der Session-Liste vom Angreifer beobachtbar, kann auch diese Information zur Verfolgung von Knoten genutzt werden. Angenommen, die Session-Liste besäße eine maximale Länge von l Einträgen und wäre gleichverteilt bei n Knoten. Würde ein Angreifer dieses Kriterium nutzen, um zwei aufeinanderfolgende (fehlgeschlagene) Wiedererkennungsversuche zu korrelieren, so wäre die Annahme, daß die beiden Protokolläufe mit identischer Listenlänge zum selben Knoten gehören, mit der Wahrscheinlichkeit $P = (1-\frac{1}{l})^{n-1}$ korrekt[6]. Auch hier gilt wiederum, daß der Angreifer für n nicht zwingend die Menge aller

[5] Die Wahrscheinlichkeit, daß ein Knoten mit einem anderen Knoten die Session-Daten an der Indexposition i abgelegt hat, beträgt $\frac{1}{l}$; die Chance, daß der Indexeintrag i bei $n-2$ weiteren Knoten *nicht* genutzt wird, ist somit $(1-\frac{1}{l})^{n-2}$. Damit ist die Wahrscheinlichkeit, daß keiner der $n-1$ verbleibenden Knoten den Indexeintrag i verwendet, $(1-\frac{1}{l})^{(n-2)(n-1)}$.
Diese Abschätzung geht davon aus, daß jeder der n Knoten mit jedem anderen eine gemeinsame Sitzung besitzt; ist dies nicht der Fall, steigt die Chance für eine erfolgreiche Korrelation weiter an: Mit einem Vernetzungsfaktor f (mit $0 \leq f \leq 1$) beträgt die Chance $(1-\frac{1}{l})^{f(n-2)(n-1)}$.

[6] Unter Annahme der Gleichverteilung beträgt die Wahrscheinlichkeit, daß ein weiterer Knoten dieselbe Listenlänge hat, $\frac{1}{l}$. Damit ist die Wahrscheinlichkeit, daß $n-1$ Knoten *nicht* dieselbe Listenlänge besitzen, $(1-\frac{1}{l})^{n-1}$.

existierenden Knoten annehmen muß, sondern für den Zweck des Trackings lediglich die Anzahl der Knoten in einem räumlich abgegrenzten Gebiet betrachtet.

Auch wenn die Annahme der Gleichverteilung der Länge der Session-Liste eine starke Vereinfachung darstellt, ist es wichtig, daß alle Knoten dieselbe Länge für ihre Session-Liste benutzen bzw. es einige wenige Listenlängen für verschiedene Knotentypen gibt. So wäre es beispielsweise denkbar, eine gemeinsame Listenlänge für kleine Knoten, eine für PDAs und Laptops sowie eine Länge für leistungsstärkere Geräte zu definieren. Ein Angreifer hat auf diese Weise ein wesentlich schwächeres Unterscheidungskriterium. Allerdings ist darauf zu achten, daß die Häufigkeitsverteilung der Geräte möglichst gleichmäßig ist (die Verfolgung eines Geräts, das einer selten auftretenden Klasse angehört, ist sonst mit recht hoher Wahrscheinlichkeit möglich). Nicht benutzte Listeneinträge sind während des Protokollaufs mit Zufallswerten zu füllen.

Klartextprotokoll nach dem Erkennungsversuch: Schlägt die Wiedererkennung fehl, so ist eine Schlüsselaushandlung der übliche nächste Schritt. Wird dieser Schritt durch Klartext-Protokollfelder eingeleitet, weiß der Beobachter, daß die Wiedererkennung fehlschlug. Will man dies vermeiden, bleiben zwei Möglichkeiten: Falls die Wiedererkennungsphase eine feste Länge hat, so kann auf die Headerinformation verzichtet werden; wird direkt mit einem Diffie-Hellman-Schlüsselaustausch fortgefahren, so wäre dies nicht von verschlüsselten Daten zu unterscheiden. Alternativ bestünde die Möglichkeit eines fingierten Schlüsselaustauschs, der trotz erfolgreicher Erkennung durchgeführt wird. Zwar könnte man hier Zufallswerte statt regulär berechneter Werte verwenden, jedoch müßte trotzdem die entsprechende Verzögerung simuliert werden, da die Finte ansonsten wegen des schnellen Ablaufs erkennbar wäre.

Verschlüsselte Kommunikation nach Ablauf des Erkennungsprotokolls: Analog zum vorigen Fall kann ein Beobachter folgern, daß sich zwei Knoten erkannt haben, wenn auf den Wiedererkennungsversuch direkt eine verschlüsselte Kommunikation folgt (und im Fehlerfall Protokollheader im Klartext zu erwarten wären). Die Gegenmaßnahmen sehen entsprechend aus.

Verdrängung von Sitzungsinformationen

Da die Liste der gespeicherten Sitzungsinformationen, die jeder Knoten besitzen muß, irgendwann voll belegt ist, muß jeder Knoten eine geeignete Strategie zur Verdrängung von Einträgen implementieren. Da die Knoten wegen räumlicher Trennung nicht in ständiger Kommunikation stehen können, kann keine gemeinsame Freigabe implementiert werden; dieser Abschnitt zeigt, daß hierdurch keine Fehlerfälle entstehen.

Für die Verdrängung bietet sich die least-recently-used-Strategie an: Es wird derjenige Eintrag ersetzt, dessen letzte Nutzung am längsten her ist. Zusätzlich ist es

denkbar, Sitzungsdaten für bestimmte Geräte (wie beispielsweise Geräte die man besitzt, besonders sensible Geräte oder solche, bei denen der initiale Schlüsselaustausch besonders aufwendig ist) gesondert zu behandeln, so daß andere Einträge bevorzugt verdrängt werden. Wegen der Möglichkeit der Verdrängung der Information beim Sitzungspartner (entweder durch reguläre Verdrängung oder aber durch Verlust wegen Strom- oder Hardwareausfalls) sollte davon abgesehen werden, Einträge als permanent unlöschbar zu markieren.

Wenn zwei Knoten A und B eine gemeinsame Sitzung vereinbart hatten, kann es bei der Verdrängung zu zwei möglichen Konstellationen kommen:

- B hat den Eintrag bereits verdrängt: Das bedeutet, daß A zwar das (früher passende) Anfragepaket sendet, da aber B nun nichts mehr damit anfangen kann, erhält A keine passende Antwort.
- A hat den Eintrag bereits verdrängt: B besäße zwar noch die alten Informationen, da A keine passende Anfrage hierzu mehr stellen kann, kommt es auch hier zu keiner Erkennung.

Wir sehen also, daß auch durch die einseitige Verdrängung von Daten keine Fehlerfälle auftreten. Der übriggebliebene Sitzungseintrag beim ehemaligen Sitzungspartner wird mangels Benutzung durch die Verdrängungsstrategie irgendwann freigegeben, weshalb dadurch auch kein Speicherleck entsteht.

5.6.4 Diskussion der Korrektheit

Um den korrekten Ablauf des Protokolls nachzuvollziehen, betrachten wir in diesem Kapitel die Formalisierung des Verfahrens mit Hilfe der BAN-Logik [40, 41]. Die Verwendung dieses Formalismus' bietet sich in sofern an, da die BAN-Logik ursprünglich zur Protokollverifikation von Authentisierungsprotokollen genutzt wurde, welche dieser Problemstellung sehr ähnlich sind. Eine knappe Einführung in die BAN-Logik, eine Übersicht der verwendeten Symbole und Schlußregeln sowie weiterführende Referenzen befinden sich im Anhang A.2.

Annahmen

Wir betrachten in dieser Analyse den Fall, daß zwischen A und B tatsächlich eine gespeicherte Sitzung existiert; anschließend überprüfen wir, ob auch ohne Kenntnis der folgenden Annahmen Informationen gewonnen werden können (das einfache Nichterkennen und ein passiver Angriff stellen dieselbe Situation dar).

A und B besitzen einen gemeinsamen Schlüssel:

A1 $\quad A \models A \overset{k_{A,B}}{\leftrightarrow} B$

A2 $\quad B \models A \overset{k_{A,B}}{\leftrightarrow} B$

A und B kennen das gemeinsame Token, das ihre Sitzungsdaten identifiziert:

A3 $\quad A \models A \overset{T_{A,B}}{\rightleftharpoons} B$

A4 $\quad B \models A \overset{T_{A,B}}{\rightleftharpoons} B$

Ziele

A und B sind davon überzeugt, daß der jeweils andere sich der eigenen Identität (spricht: des Sitzungstokens) vergewissert hat:

Z1 $\quad A \models B \models A \overset{T_{AB}}{\rightleftharpoons} B$

Z2 $\quad B \models A \models A \overset{T_{AB}}{\rightleftharpoons} B$

Da bei der Erstellung der Sitzungsinformationen das Sitzungstoken 1:1 mit dem korrespondierenden Schlüssel verknüpft ist, ist somit auch klar, daß der jeweils andere auch einen passenden Schlüssel besitzt.

Protokollablauf

In Nachricht 2.1 erzeugt A die Nonce n_1 und sendet diese zusammen mit einem verschlüsselten Token an B:

1	$A \models \sharp(n_1)$	Wurde von A selbst erzeugt
2	$B \lhd \left\{ A \overset{T_{AB}}{\rightleftharpoons} B, n_1 \right\}_{k_{A,B}}$	Token und Schlüssel aus A1 und A3

B kann das Paket entschlüsseln und sieht so das zu $k_{A,B}$ korrespondierende Token:

3	$B \lhd (A \overset{T_{AB}}{\rightleftharpoons} B, n_1)$	Aus Schritt 2, Annahme A2 und Regel A.9

Damit glaubt B auch, daß das Paket (zu irgendeinem Zeitpunkt – es könnte auch ein Replay-Angriff sein) von A gesendet wurde:

4	$B \models A \mid\sim (A \overset{T_{AB}}{\rightleftharpoons} B, n_1)$	Aus Schritt 2, Annahme A2 und Regel A.1

Die Schritte 3 und 4 werden zwar im folgenden nicht weiter verwendet; die hier gewonnene Information ist für B dennoch wichtig: Die Definition der BAN-Logik nimmt an, daß ein Knoten die erfolgreiche Entschlüsselung eines Pakets erkennt.

Da hier jedoch nur Nonces und ein (rein zufällig generiertes) Sitzungstoken ausgetauscht werden, ist eine erfolgreiche von einer fehlgeschlagenen Entschlüsselung nicht unterscheidbar. B kann diese Entscheidung dennoch treffen, da es zu einem $k_{A,B}$ nur ein korrespondierendes Token $T_{A,B}$ gibt – die Entschlüsselung mit $k_{A,B}$ muß also $T_{A,B}$ ergeben.

B generiert nun die Nonce n_2 und sendet diese in der Nachricht 2.2 an A (die Operationen auf den Nonces werden in dieser Notation der Übersichtlichkeit wegen weggelassen):

5	$B \models \sharp(n_2)$	Wurde von B selbst erzeugt
6	$A \lhd \left\{ A \overset{T_{AB}}{\rightleftharpoons} B, n_1, n_2 \right\}_{k_{A,B}}$	Token und Schlüssel aus A2 und A4

Hieraus läßt sich das Ziel Z1 ableiten:

7	$A \models B \mid\sim (A \overset{T_{AB}}{\rightleftharpoons} B, n_1, n_2)$	Aus Schritt 6, Annahme A1 und Regel A.1
8	$A \lhd (A \overset{T_{AB}}{\rightleftharpoons} B, n_1, n_2)$	Aus Schritt 6, Annahme A1 und Regel A.9
9	$A \models \sharp(A \overset{T_{AB}}{\rightleftharpoons} B, n_1, n_2)$	Aus Schritt 8, Schritt 1 und Regel A.8
10	$A \models B \models (A \overset{T_{AB}}{\rightleftharpoons} B, n_1, n_2)$	Aus Schritt 9, Schritt 7 und Regel A.4

Damit gilt auch

11	$A \models B \models A \overset{T_{AB}}{\rightleftharpoons} B$	Aus Schritt 10 und Regel A.6

womit Ziel Z1 erreicht ist. A sendet zuletzt an B Nachricht 2.4:

12	$B \lhd \left\{ A \overset{T_{AB}}{\rightleftharpoons} B, n_2 \right\}_{k_{A,B}}$	

Mit dieser Information kann auf das Ziel Z2 geschlossen werden:

13	$B \models A \mid\sim (A \overset{T_{AB}}{\rightleftharpoons} B, n_2)$	Aus Schritt 12, Annahme A2 u. Regel A.1
14	$B \lhd (A \overset{T_{AB}}{\rightleftharpoons} B, n_2)$	Aus Schritt 12, Annahme A2 u. Regel A.9
15	$B \models \sharp(A \overset{T_{AB}}{\rightleftharpoons} B, n_2)$	Aus Schritt 14, Schritt 5 und Regel A.8
16	$B \models A \models (A \overset{T_{AB}}{\rightleftharpoons} B, n_2)$	Aus Schritt 15, Schritt 13 und Regel A.4

Somit gilt auch

17	$B \models A \models A \overset{T_{AB}}{\rightleftharpoons} B$	Aus Schritt 16 und Regel A.6

Damit ist auch Ziel Z2 erreicht.

Wie bereits oben erwähnt, wird der Schlüssel $k_{A,B}$ zusammen mit dem Sitzungstoken ausgehandelt; außerdem ist die obige Kommunikation nur mit dem passenden Schlüssel möglich. Somit ist für A und B auch klar, daß der jeweils andere den Schlüssel zur gespeicherten Sitzung besitzt.

Wir stellen also fest, daß keine der übertragenen Informationen überflüssig ist: Jede Angabe wird für den Beweis benötigt (der Zweck der Schritte 3 und 4 wurde bereits oben erläutert). In den folgenden Abschnitten wird die Sicherheit gegen die in Kapitel 5.6.2 vorgestellten Angriffsszenarien betrachtet.

Das Protokoll ist robust gegen die erwähnten Angriffe auf die Identifizierung: Die Formalisierung schlägt fehl, wenn man eine der Anfangsbedingungen wegläßt. Dies bedeutet, daß alle diese Angaben nötig sind und somit ein Angreifer nicht in der Lage ist, sich ohne Kenntnis dieser (geheimen) Informationen als A oder B auszugeben. Ein Impersonation-Angriff ist also nicht möglich.

Analoges gilt für einen Man-in-the-Middle-Angriff: Einem aktiven, zwischengeschalteten Angreifer ist es zwar möglich, die Pakete weiterzuleiten; da er jedoch mangels Kenntnis des Schlüssels nicht in der Lage ist, die Pakete sinnvoll zu modifizieren, stellt dieser Angriff schlimmstenfalls einen Denial-of-Service-Angriff dar. Die Identifizierung ist hierdurch nicht gefährdet.

Die Schritte 9 und 15 sind nur erfüllt, wenn die Nonces n_1 und n_2 auch tatsächlich frisch sind; daraus folgt, daß das Präsentieren alter, gespeicherter Pakete erkannt wird und somit auch ein Replay-Angriff nicht möglich ist.

Ein Sonderfall ist ein Man-in-the-Middle M, der selbst eine gespeicherte Sitzung mit A besitzt. M wird A durch reines Abhören der Kommunikation zwischen A und B erkennen (sofern A die Sitzungsinformation noch nicht verdrängt hat), denn A iteriert über seine gesamte Sitzungsliste, welche neben $(T_{A,B}, k_{A,B})$ auch $(T_{A,M}, k_{A,M})$ enthält. Dies ist unvermeidlich, schließlich geht es A nicht darum, ausschließlich B wiederzuerkennen (A weiß zu diesem Zeitpunkt auch noch nicht, wer sein Kommunikationspartner ist), sondern um die Wiedererkennung *aller* A bekannten Knoten. Wenn wir das motivierende Beispiel von auf Seite 64 betrachten, ist dies da Analogon zu zwei Bekannten A und B, die sich in der Menschenmenge treffen, und dabei von einer dritten Person, welche Person A kennt, gesehen werden. Natürlich wird dieser Dritte A ebenfalls wiedererkennen. Sollte Knoten A nicht mehr wünschen, von M erkannt zu werden, so muß A lediglich den Sitzungseintrag $(T_{A,M}, k_{A,M})$ aus seiner Sitzungsliste entfernen.

Sollte M neben der Sitzung mit A auch eine Sitzung mit B besitzen, so kann sich M tatsächlich zwischen A und B klinken; allerdings wird er sowohl von A als auch von B als M identifiziert, es ist M nicht möglich, sich als A oder B auszugeben, da er nicht im Besitz von $(T_{A,B}, k_{A,B})$ ist (und sowohl Tokens als auch Schlüssel für jede gespeicherte Sitzung verschieden sind). Damit stellt dies keinen Angriff auf das Protokoll dar, da M sich nicht als jemand anderes ausgeben kann.

Angriffsmöglichkeiten auf die Anonymitätseigenschaft wurden teilweise bereits in Kapitel 5.6.3 diskutiert: Dort wurden Probleme betrachtet, die sich auf den gesamten Ablauf (und nicht nur einen einzelnen Protokollauf, wie hier in BAN-Logik modelliert) beziehen. Ebenso ist es nicht möglich, einen Knoten anhand der übertragenen Pakete wiederzuerkennen: Jedes Paket enthält frisch generierte Nonces, weshalb sich diese immer von früher gesendeten Paketen unterscheiden. Verbleibendes Problem ist also das Probing, sprich: Das Generieren oder Wiedereinspielen von Paketen, um erkennbare Reaktionen zu provozieren. Ohne Kenntnis des Schlüssels ist es nicht möglich, sinnvolle Pakete zu generieren, was diesen Angriffsvektor ausschließt.

Probing durch Replay bietet dem Angreifer ebenfalls keine Information. Idee des Angriffs ist folgende: Der Angreifer hört einen erfolgreichen Handshake ab und spielt zu einem späteren Zeitpunkt eines der Pakete erneut ins Netz ein. Anhand des darauf folgenden Antwortpakets versucht er durch Vergleich mit dem abgehörten Paket zu unterscheiden, ob dies die Antwort des Knotens aus der abgehörten Kommunikation ist.

Nutzt der Angreifer hierfür ein Paket aus Schritt 2.1, so ist die Antwort beim „falschen“ Knoten eine Nachricht aus Zufallszahlen (Nachricht 2.3); ist es der „richtige“ Knoten, so antwortet dieser mit Nachricht 2.2 – diese Nachricht enthält jedoch eine zweite, frisch generierte Nonce, was (zusammen mit der Annahme eines passenden Verwendungsmodus der Chiffre, siehe Anhang A.1.3) bedeutet, daß sich das Paket vom vorher abgehörten Paket komplett unterscheidet. Der Angreifer gewinnt hieraus keine Information.

Versucht der Angreifer, ein Paket aus Schritt 2.2 zu nutzen, so enthält dies eine ungültige Nonce n_1. Dadurch erkennt A den Angriffsversuch und behandelt das Paket wie eine Tarnnachricht 2.3. Auch hier gewinnt der Angreifer keine weitere Information, da sich A so verhält, als ob er mit der Antwort nichts anfangen könnte.

Die Verfügbarkeit kann durch das Stören des Protokolls beeinträchtigt werden: Fälscht der Angreifer Pakete, so daß diese die Netzwerkadresse des eigentlichen Kommunikationspartners besitzen, und antwortet er schneller als der eigentliche Adressat, so kann er durch Einspielen von Dummy-Nachrichten eine Erkennung unterbinden (dasselbe gilt, falls der Angreifer als Man-in-the-Middle agiert). Allerdings benötigt dieser Angriff pro Anfragepaket eine Antwort des Angreifers, der Angreifer muß für die gesamte Dauer des Protokolls aktiv bleiben. Somit ist dieser Angriff nicht effizienter als das bloße Stören der Funkschnittstelle durch einen Störsender oder nicht protokollkonformen Medienzugriff.

Wir stellen also fest, daß das Protokoll gegen die in Kapitel 5.6.2 vorgestellten Angriffsszenarien resistent ist.

5.6.5 Wiedererkennung mit Index-Optimierung

In den vorigen Abschnitten haben wir ein Protokoll zur Wiedererkennung von anonymen Knoten vorgestellt und dessen korrekte Funktionsweise belegt. In diesem Abschnitt soll nun der Aufwand betrachtet und optimiert werden.

Das vorgestellte Protokoll erfüllt zwar die Anforderungen, ist aber leider sehr aufwendig – bei einer Listenlänge l ergibt sich:

Netzwerkpakete gesendet von A : A sendet für jeden Listeneintrag in Schritt 2.1 ein Paket (Schritt 2.4 enthält bereits Nutzdaten) – ein Aufwand von $O(l)$.

Rechenzeit bei A**:** A muß für jeden Listeneintrag ein Paket verschlüsseln (Schritt 2.1) und eines entschlüsseln (Schritt 2.2 bzw. 2.3); damit werden $O(l)$ Krypto-Operationen (also Ver- und Entschlüsselungen) ausgeführt.

Netzwerkpakete gesendet von B : B sendet für jedes empfangene Paket eine Antwort (Schritt 2.2/2.3), was einen Aufwand von $O(l)$ bedeutet.

Rechenzeit bei B**:** B muß für jedes in Schritt 2.1 empfangene Paket alle in seiner Sitzungsliste gespeicherten Schlüssel durchprobieren (und zusätzlich die Nonce in Schritt 2.4 überprüfen). Damit sind bei B $O(l^2)$ Kryptooperationen erforderlich.

Für die Zahl der zu übertragenden Pakete gibt es keine Optimierungsmöglichkeit, da ja beiden Kommunikationspartnern die Identität ihres Gegenübers nicht bekannt ist. Ein schwerwiegendes Problem ist jedoch der hohe Aufwand bei B: Abgesehen vom Zeitaufwand bedeutet dies auch einen hohen Energiebedarf, welcher bei batteriebetriebenen Geräten die Einsatzzeit stark limitiert. Dies erleichtert Sleep-Deprivation-[214, 213], Power-Drain- oder Barrage-Attacks[177], bei denen ein batteriebetriebenes Gerät mit rechenintensiven Aufgaben so lange beschäftigt wird, bis dessen Energiereserven aufgebraucht sind.

Die Idee hinter dieser im folgenden dargestellten Verbesserung läßt sich folgendermaßen skizzieren: Wenn man B einen „Tipp“ gibt, wo in der Sessionliste nach dem Eintrag zu suchen ist, kann so die Anzahl der potentiellen Sessions reduziert werden. Da jedoch dieser Hinweis im Klartext übertragen werden muß, muß darauf geachtet werden, daß ein Beobachter diese Information nicht zur Identifizierung der Knoten benutzen kann.

In [194] stellten wir ein Verfahren vor, das einen nicht eindeutigen Index als Hinweis verwendete. Hier wurde ein Tradeoff zwischen Performance-Verbesserung durch die Einführung des Index und einer gewissen Privacy-Einbuße erreicht. Dieser Indexwert wird beim ersten Kontakt zusammen mit den Sitzungsinformationen gespeichert und im ersten Paket des Wiedererkennungsprotokolls mitgeschickt (der Rest des Protokolls bleibt unverändert):

$$A \rightarrow B : \forall B : \ index_B, E_{k_{A,B}}(T_{A,B}, n_1) \tag{3.1}$$

Beim Erhalt muß B nun nicht mehr alle Einträge seiner Liste prüfen, sondern nur noch die, bei denen $index_B$ mit dem gespeicherten Index-Wert der Listeneinträge übereinstimmt.

Um die Privacy-Einbuße bei einer Sitzungsliste mit Länge l und einem Index-Wertebereich r möglichst gering zu halten, wurde versucht, bei allen Knoten ein möglichst homogenes Verhalten zu erreichen (wir nehmen an, daß die Indexwerte gleichverteilt sind):

- Ist die Reihenfolge, in der die Sitzungsliste getestet wird, immer die selbe, so formt die Sequenz der Indexwerte ein identifizierendes Merkmal: Ein Knoten wäre mit einer Wahrscheinlichkeit von $P = 1 - \frac{1}{r^l}$ identifizierbar.
- Sortiert man die Liste nach dem Indexwert, so bleibt dem Angreifer als einzige Information die Anzahl der Vorkommen der einzelnen Indexausprägungen. Die Wahrscheinlichkeit, daß es sich bei zwei Knoten mit derselben Häufigkeit der Indexausprägungen um denselben Knoten handelt, beträgt somit $P = 1 - \frac{1}{\binom{r-1+l}{l}}$.[7]
- Sorgen die Knoten beim Aushandeln des gemeinsamen Indexwerts bewußt dafür, daß jede Indexausprägung möglichst gleich häufig vorkommt, so kann es maximal zu einer Verschiebung um ein Element (eine Gruppe mit gleichem Indexwert besitzt einen Wert mehr/weniger als der Rest) kommen; somit ist die Chance der Wiedererkennung etwa $P = 1 - \frac{1}{1+l*(l-1)}$.[8]

Die verbesserte Fassung, die im folgenden vorgestellt wird, verwendet zwei unabhängige Indices: Einen Sende-Index i_S und einen Empfangs-Index i_R. So sind zwei Kommunikationspartner A und B nicht gezwungen, sich auf eine gemeinsame Indexnummer zu einigen. Dies beseitigt die Fälle der letzten Variante, in denen einer der Indexwerte häufiger bzw. seltener als die restlichen vorkam.

Betrachten wir das Beispiel in Bild 5.2 auf der nächsten Seite, was die Konstellation vom Beispiel in Abbildung 5.1 auf Seite 69 um Indices erweitert darstellt: Sendet Knoten 1 das zweite Paket, so versieht es dieses mit dem Sende-Index $i_S = 2$ als Hinweis. Knoten 2 muß beim Empfang nun nur die Pakete mit Empfangs-Index $i_R = 2$ überprüfen – also die Einträge eins, vier und fünf (wobei Eintrag eins der zur Anfrage von Knoten 1 passende Eintrag ist). Wie man an diesem Beispiel sieht, muß die Umkehrung (dargestellt in Abbildung 5.3 auf der nächsten Seite) nicht symmetrisch sein: Sendet Knoten 2 das erste Paket an Knoten 1, so versieht

[7] Jeder Indexwert wird aus dem Wertebereich zufällig gewählt; jeder Indexwert kann mehrfach auftreten, durch die Sortierung wird die Reihenfolge gezielt verschleiert. Dies entspricht Ziehen mit Zurücklegen ohne Berücksichtigung der Reihenfolge.

[8] Unter der vereinfachten Annahme, daß $l = 0 \bmod r$. Die Zahl der Kombinationen ergibt sich aus der Möglichkeit, daß alle Indexwerte gleichhäufig vorkommen sowie der Tatsache, daß einer der verbleibenden $l - 1$ Werte um eins kleiner sein muß, falls einer der l Werte um eins größer als der Mittelwert ist.

Knoten 2 dies mit dem Sende-Index $i_S = 1$. Knoten 1 testet nun alle Einträge mit Empfangs-Index $i_R = 1$, also Eintrag eins und zwei.

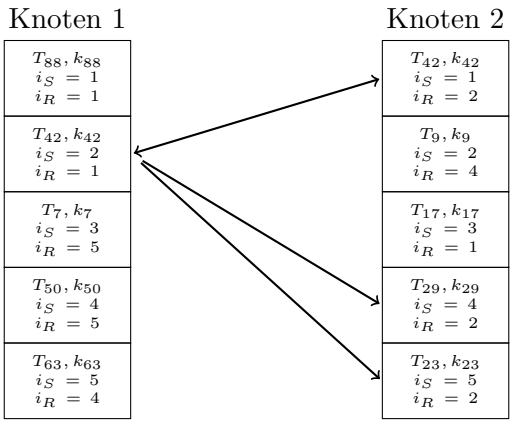

Abbildung 5.2: Beispiel mit Indices. Korrespondierende Einträge bei Knoten 2 zum zweiten Listenelement von Knoten 1 ($i_S = 2$).

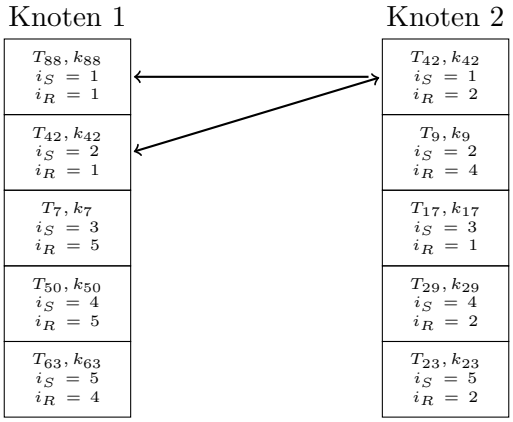

Abbildung 5.3: Beispiel mit Indices. Korrespondierende Einträge bei Knoten 1 zum ersten Listenelement von Knoten 2 ($i_S = 1$).

Wie im Abschnitt über beobachtbare Merkmale auf Seite 69 bereits erwähnt, ist es möglich, den Abbruch des Wiedererkennungsprotokolls zum Verfolgen von Knoten zu nutzen, falls das Ende des Protokolls von der folgenden Kommunikation für einen Beobachter unterscheidbar ist. Es wurde vorgeschlagen, entweder die Liste der Sitzungen unabhängig von der Erkennung komplett durchzugehen, oder aber die Reihenfolge zu randomisieren. Letzteres ist beim Einsatz des Index nur noch begrenzt möglich: Die Reihenfolge kann nur innerhalb desselben Indexwerts geändert werden. Das bedeutet, daß die Gefahr durch Beobachter steigt, je feingranularer der Index ist. Mit feinerem Index steigt jedoch auch die Effizienz des Verfahrens.

Im folgenden betrachten wir den Aufwand für B beim kompletten Durchlauf des Verfahrens (also bei einem Fehlschlag). Wie oben sei l die Länge der Sitzungsliste und r der Wertebereich des Index. n_i sei die Anzahl der Listeneinträge bei B mit Indexwert i. Unter der Annahme, daß $l = 0 \bmod r$, sendet A für jeden Wert des Index $\frac{l}{r}$ Anfragen. Damit läßt sich die Zahl der Entschlüsselungsoperationen c bei B bestimmen:

$$c = \sum_{i=1}^{r} \frac{l}{r} n_i = \frac{l}{r} \sum_{i=1}^{r} n_i$$

Da die Summe der Listeneinträge n_i die gesamte Listenlänge l ergeben muß, folgt somit:

$$c = \frac{l}{r} \sum_{i=1}^{r} n_i = \frac{l^2}{r}$$

Verwendet man eine Indexlänge mit Länge der Liste ($r = l$), so folgt:

- B benötigt somit $c = l$ Kryptooperationen. Man erreicht für B einen linearen Aufwand von $O(l)$.
- Der Indexwert i_S muß nicht mehr mitübertragen werden, diesen kann B durch Mitzählen der Pakete (für das j-te empfangene Paket wird $i_S = j$ angenommen) selbst bestimmen.

Mit Hilfe des Index kann der Aufwand bei B auf ein lineares Maß ($O(l)$) reduziert werden. Wie oben beschrieben, muß dafür auch im Falle des Erkennens die gesamte Liste abgearbeitet werden, falls die Folgekommunikation vom Erkennungsprotokoll unterscheidbar ist. Dies ist im Vergleich zur Aufwandsersparnis eine geringe Einbuße.

5.6.6 Optimierung durch Reduktion der Anzahl der Antwortpakete

Kommt die Indexoptimierung des vorigen Kapitels zum Einsatz (oder eine der in Kapitel 5.6.3 auf Seite 69 diskutierten Fälle zum tragen), so muß in jedem Fall die gesamte Sitzungsliste von A durchgegangen werden. In diesem Fall lassen sich die Antwortpakete von B reduzieren: B antwortet erst am Ende der gesamten Liste im Falle eines Erkennens mit Nachricht 2.2, ansonsten mit Nachricht 2.3.

Dadurch sendet B nur ein einzelnes Paket (anstelle von l Paketen). A weiß zwar nun nicht, zu welchem Listeneintrag die Antwort gehörte und muß deshalb sämtliche l Listeneinträge durchprobieren. Dafür muß A dies nur mit einem einzelnen Paket tun, und nicht mehr mit l Paketen (bei denen der zugehörige Listeneintrag klar gewesen wäre). Damit bleibt der Aufwand für A identisch.

5.6.7 Nutzung des Verfahrens zum schnellen Neuaufsetzen einer Sitzung

Neben der Wiedererkennung von Knoten, denen man vor längerer Zeit begegnet ist, kann das Verfahren auch in einer anderen Situation zum Einsatz kommen: Wenn zwei Knoten eine längere Kommunikation führen, kann einer der Knoten den Wunsch haben, die Verbindung temporär zu unterbrechen, um sein Pseudonym zu wechseln um so für Beobachter schwerer verfolgbar zu sein. Da nach dem Pseudonymwechsel eine Wiedererkennung erfolgen muß, bietet sich auch für dieses Szenario die Verwendung des vorgestellten Algorithmus' an.

Die Kommunikationspartner handeln hierzu Sitzungsdaten für die Wiedererkennung aus und assoziieren diese mit allen Informationen der laufenden Kommunikation. Nach der Unterbrechung (und einer Pause) wechseln die Knoten ihr Pseudonym und erkennen sich mit Hilfe des Protokolls wieder. Anhand der Daten, die mit der Sitzung assoziiert wurden, können sie ihre Kommunikation fortsetzen.

Zur Optimierung kann hierfür eine zweite, sehr kurze Liste (beispielsweise mit fünf Einträgen) von Sitzungen benutzt werden; beim Beenden der Kommunikation kopieren sie die ausgehandelte Sitzung zusätzlich in diese Liste. Optimal ist hierfür die Sortierung der Liste nach der letzten Benutzung (die neueste Sitzung zuerst), dafür wird auf die Nutzung eines Index verzichtet. Beim Wiedererkennungsversuch durchlaufen beide Partner zunächst diese kurze Liste. Findet keine Erkennung statt, wird wie gehabt die lange Liste getestet. Da die Sitzungsinformation kopiert wurde, bedeutet im Zweifelsfall eine Verdrängung aus der kurzen Liste nicht, daß die Sitzung verloren ist, da die Daten noch in der langen Liste vorhanden sind (die Chance, daß die Daten auch aus einer Liste mit mehreren hundert Einträgen verdrängt wurden, ist vernachlässigbar).

Auch hier sind natürlich die beobachtbaren Merkmale (siehe 5.6.3 auf Seite 69) zu beachten: Sieht ein Beobachter zwei kommunizierende Knoten und erkennt nach einer kurzen Pause, daß zwei Knoten (mit neuen Pseudonymen) sich mit diesem „Quick Session Resume“ wiedererkennen, so kann er mit hoher Wahrscheinlichkeit folgern, daß es sich um dieselben Knoten wie vorher handelt. Entsprechende Gegenmaßnahmen zur Ununterscheidbarkeit der Kommunikationen wurden bereits in Abschnitt 5.6.3 diskutiert.

5.6.8 Erweiterung für Multicasts

Das bisherige Protokoll war für die Wiedererkennung von genau zwei Knoten bei direkter Kommunikation ausgelegt. Für die (im vorigen Absatz erläuterte) schnelle Wiederaufnahme von Sitzungen ist dies zwar ausreichend, jedoch wäre es von Vorteil, mit einer Variante des Protokolls eine Möglichkeit zu besitzen, mit der jeder Knoten jeden weiteren Knoten in seiner Umgebung wiedererkennen kann.

Im Kapitel über die Korrektheit des Verfahrens (Seite 75) beschrieben wir, daß ein weiterer Zuhörer M den Knoten A ebenfalls erkennt, so er mit diesem ebenfalls eine gemeinsame Sitzung besitzt. Zusammen mit der Optimierung aus Kapitel 5.6.6 auf Seite 80 läßt sich hieraus eine einfache Multicast-Variante des Protokolls konstruieren: Jeder Knoten verschickt seine gesamte Liste an Anfragen per Multicast; jeder umliegende Knoten sendet jeder Anfrage (per Unicast) sein Antwortpaket.

Ein Nachteil des Verfahrens ist, daß es teilweise auf der Kooperation der umliegenden Knoten basiert; der Schlüssel ist hierbei die fingierte Antwort im Falle des Nichterkennens. Angenommen, Knoten würden in diesem Fall überhaupt keine Antwort (anstatt des fingierten Antwortpakets) senden, so könnte ein Angreifer mit Hilfe eines Replay-Angriffs den Listeneintrag identifizieren, auf den ein Knoten mit einer korrekten Antwort reagiert: Der Angreifer verändert den Mitschnitt, indem er sukzessive die Anfragen durch Veränderung ungültig macht; antwortet der Knoten noch immer, so ist das korrekte Anfragepaket noch dabei. Dies würde letztlich einen Probing-Angriff erlauben. Die Schwierigkeit besteht darin, daß Knoten, die nur mit einer *decoy*-Antwort reagieren müßten, selbst keinen unmittelbaren Nachteil haben, wenn sie diese nicht senden. Ein Grund für eine solche Handlungsweise kann egoistisches Verhalten (um Energie zu sparen) sein. Verhalten sich aber alle (oder zumindest der überwiegende Teil) der Knoten so, so hat ein Angreifer eine hohe Chance, einen Probing-Angriff zu starten.

Um dieses Problem zu umgehen, muß sichergestellt werden, daß schon die Anfragepakete frisch (also kein Replay) sind. Wenn man davon ausgeht, daß eine erfolgreiche Wiedererkennung ohnehin an der folgenden Menge übertragener Daten erkannt werden kann, genügt es dann auch, wenn die Knoten nur eine erfolgreiche Identifizierung melden.

In der folgenden Beschreibung gehen wir davon aus, daß Knoten A die umliegenden Knoten $B_1, \ldots B_i$ wiedererkennen möchte. Das Protokoll läuft symmetrisch ab – jeder der umliegenden Knoten durchläuft dieselben Schritte. Das angepaßte Protokoll lautet wie folgt:

A generiert eine Nonce n_A und sendet diese per Multicast an alle Nachbarn:

$$A \rightarrow all : n_A \tag{4.1}$$

A empfängt die Nonces $n_{B_1}, \ldots, n_{B_i}$, da alle Nachbarn genauso verfahren. Hierzu wartet A eine gewisse Zeitspanne t_Δ, um den umliegenden Knoten die Chance zur Reaktion zu geben.

Analog zum Unicast-Protokoll sendet A nun die Pakete zur Wiedererkennung. Den Paketen vorangestellt ist eine Liste der empfangenen Nonces – da Multicasts typischerweise unzuverlässig sind (und sich Knoten durch Bewegung aus der Funkreichweite hinausbewegt haben), ist es möglich, daß der sendende Knoten nur einen Teil der Nonces empfangen hat, die A erhalten hat. Ebenfalls neu ist der Hashwert aller

Nonces $H(n_{B_1}, \ldots, n_{B_i})$ im verschlüsselten Teil.

$$A \rightarrow all : \forall S : \; n_{B_1}, \ldots, n_{B_i}, E_{k_{A,S}}(T_{A,S}, n2_A, H(n_{B_1}, \ldots, n_{B_i})) \tag{4.2}$$

A empfängt die Listen der umliegenden Knoten. Zur Wiedererkennung überprüft A bei jeder Liste, ob die eigene Nonce n_A in der Liste enthalten ist. Falls dem so ist, verfährt A wie beim Unicast-Protokoll; bei einer erfolgreichen Erkennung prüft A zusätzlich die Korrektheit des Hash-Wertes, indem A selbst den Hash der Nonces bestimmt und mit dem Wert des verschlüsselten Teils vergleicht. So stellt A die Frische des Pakets sicher.

Die Verwendung eines Hashwertes sorgt für eine konstante Größe der Pakete (ansonsten müßten alle Nonces einzeln im Paket aufgeführt werden). Trotz dieser Platzoptimierung entsteht für einen Angreifer keine Replay-Möglichkeit: Zwar könnte ein Angreifer aufgezeichnete Pakete wiedergeben und dabei die vorangestellte Liste der Nonces verändern; da er den Schlüssel nicht kennt, kann er den inneren Hashwert nicht ändern, weshalb er passende Nonces $n'_1, \ldots, n'_j$ bestimmen müßte, so daß $H(n_A, n'_1, \ldots, n'_j) = H(n_1, \ldots, n_i)$ wäre – dies ist jedoch aufgrund der Eigenschaften einer kryptographischen Hashfunktion (Siehe Anhang A.1.4) praktisch ausgeschlossen.

Bei einer erfolgreichen Wiedererkennung mit Nachbar B_k bestätigt A die Frische seiner Kommunikation mit einer Operation auf der Nonce $n2_{B_k}$ (analog zum Unicast-Protokoll):

$$A \rightarrow B : E_{k_{A,B_k}}(T_{A,B_k}, n2_{B_k} + 1) \tag{4.3}$$

Damit ist die beidseitige Wiedererkennung abgeschlossen. Der Aufwand für jeden beteiligten Knoten beträgt somit:

- Senden der Liste: l Pakete
- Erstellen der Liste, überprüfen der i empfangenen Listen (mit Indexoptimierung): $l(i+1)$ Ver- und Entschlüsselungs-Operationen sowie $i+1$ Hashoperationen. Im Falle einer Wiedererkennung zusätzlich die Überprüfung des Pakets 4.3.

Die Synchronisierung des Protokolls kann folgendermaßen geschehen: Jeder Knoten besitzt zwei Timeouts, t_r und t_i mit $t_r < t_i$. Die Timeouts bestimmen die minimale und die maximale Zeit, die ein Knoten zwischen zwei Erkennungsversuchen vergehen lassen will. Erhält ein Knoten eine Nachricht 4.1, bevor der Timeout t_r abgelaufen ist, reagiert er nicht. Ist t_r hingegen abgelaufen, beteiligt sich der Knoten an dem Protokoll, indem er ebenfalls Nachricht 4.1 sendet. Läuft der Timeout t_i ab, so ergreift der Knoten die Initative und startet das Protokoll mit Nachricht 4.1, ohne daß eine Nachricht von außen erhalten wurde.

Das anfangs geschilderte egoistische Verhalten hat nun auch keine weiteren Konsequenzen: Beteiligt sich ein Knoten von vornherein nicht am Protokoll, so bleibt er unsichtbar. Sendet er Nachricht 4.1, nicht aber die Liste (Nachricht 4.2), so wird

seine Nonce zwar in den Listen der benachbarten Knoten berücksichtigt; er selbst kann aber nicht erkannt werden, was aber den Protokollablauf der anderen Knoten nicht weiter beeinträchtigt.

5.6.9 Optimierung für den praktischen Einsatz

Um die Übersichtlichkeit zu wahren, wurden in den obigen Protokolldiagrammen immer einzelne verschlüsselte Pakete versandt. Natürlich macht es Sinn, die Netzwerkpakete beispielsweise mit mehreren Identifizierungsanfragen bis zur MTU[9] zu füllen. Dadurch wird die Netzlast reduziert und der Protokollvorgang beschleunigt, da seltener auf den Medienzugriff gewartet werden muß.

Im Abschnitt über die beobachtbaren Merkmale auf Seite 69 wurden verschiedene Aspekte diskutiert, die ein Angreifer in Abhängigkeit von Szenario und Implementierung beobachten kann. Für den praktischen Gebrauch ist abzuwägen, welche der Informationen für die Anwendung tatsächlich kritisch sind; des weiteren sind die Hinweise dort zu beachten, wie die verbleibenden Probleme umgehbar sind. Durch geeignetes Design des Anwendungsprotokolls wie beispielsweise dem Padding auf feste Paketlängen können dem Angreifer viele Ansatzpunkte genommen werden.

Wie eingangs erwähnt, definiert dieses Kapitel keine Vorgaben für Algorithmen oder Schlüssellängen. Diese müssen entsprechend den verwendeten Geräten (und dem Angriffsszenario) gewählt werden. Insbesondere bei den symmetrischen Chiffren ist jedoch zu beachten, daß einige von ihnen ein relativ ungleichmäßiges Laufzeitverhalten besitzen. So gibt es Verfahren (wie beispielsweise den Twofish-Algorithmus [199]), die eine sehr aufwendige Schlüsselinitialisierung besitzen; das bedeutet, daß der Algorithmus unter Umständen zwar große Datenmengen sehr schnell verschlüsseln kann, jedoch überproportional lange für die Aufbereitung des Schlüsselmaterials braucht. Ein solches Vorgehen ist in manchen Situationen durchaus von Vorteil, da das Durchprobieren von Schlüsseln (brute force attack, dictionary attack) dadurch gebremst wird, ohne daß dem legitimen Nutzer beim Verschlüsseln großer Datenmengen dadurch signifikante Nachteile entstehen. In unserem Algorithmus werden jedoch nur kurze Datenpakete mit vielen verschiedenen Schlüsseln bearbeitet, weshalb ein solches Design hier von Nachteil ist (einen Vergleich des Initialisierungsaufwands mehrerer moderner symmetrischen Chiffren findet man in [200]).

5.7 Fazit

In diesem Kapitel haben wir den Interessenskonflikt zwischen Anonymität und Verschlüsselung diskutiert. Durch die Fallunterscheidung aus Kapitel 5.4 auf Seite 62

[9] MTU: Medium Transfer Unit. Maximale Paketlänge auf dem Transfermedium.

konnte die Häufigkeit des initialen Schlüsselaustauschs, welcher das größte Problem hierbei darstellt, möglichst weit reduziert werden.

Für Situationen, in denen zunächst ein Man-in-the-Middle toleriert werden kann, haben wir in Kapitel 5.5 auf Seite 63 das verzögerte Station-to-Station-Protokoll vorgeschlagen. Hiermit kann erst ab dem Punkt, ab dem eine der beiden Parteien die gesicherte Verbindung höher bewertet als die eigene Anonymität, die Authentisierung erfolgen.

Zuletzt wurde in Kapitel 5.6 das neue Problem der Wiedererkennung anonymer Knoten vorgestellt, womit sich Knoten, die zu einem früheren Zeitpunkt auf gesichertem Weg Informationen ausgetauscht haben, wiedererkennen können, obwohl sie nun unter einem neuen Pseudonym agieren. Neben dem Basisalgorithmus haben wir eine in ihrer Performance verbesserte Fassung und eine Multicast-Adaption vorgestellt.

Zusammen mit den vorigen Kapiteln sind wir nun in der Lage, Systeme so zu konzipieren, daß sie anonym aber abhörsicher kommunizieren können. Je nach Situation ist das Etablieren eines authentischen Schlüssels nicht nötig oder kann bei Bedarf nachgeholt werden. Ist ein solcher authentischer Schlüssel einmal etabliert, kann er bei Bedarf wiederverwendet werden. Ein Wiederverwenden eines Schlüssels erlaubt es jedoch zumindest den beiden Kommunikationspartnern, die damit geführten Sessions zu assoziieren.

Die hier vorgestellten Mechanismen können fast vollständig unabhängig von der darüberliegenden Anwendung agieren. Die Schnittstelle zur Anwendungsschicht kann wie folgt aussehen (wobei nicht alle Funktionen im nächsten Kapitel Verwendung finden):

setBackoffTime(t) Erlaubt die Konfiguration der Backoff-Zeit.

getSessionDuration() Liefert die Zeit, wie lange die aktuelle Sitzung bereits dauert.

resetSession() Erzwungenes Zurücksetzen der Sitzung und sofortiger Eintritt in die Backoff-Zeit. Hiermit werden alle offenen Verbindungen unterbrochen.

allowSessionReidentification() Gibt an, daß die aktuelle Sitzung im Cache zur Wiedererkennung aufgenommen werden soll, also eine gegenseitige Wiedererkennung mit dem aktuellen Gegenüber erwünscht ist.

reidentifiedNodeFound(callback) Wird bei der periodischen Wiedererkennung ein befreundeter Knoten gefunden, wird die Anwendung über den Callback benachrichtigt.

isEncryptionAuthentic() Abfrage, ob die Verschlüsselung authentisch ist, oder ob mit einem Man-in-the-Middle-Angriff gerechnet werden muß.

getSessionID() Abfrage der Session-ID; diese ID ist für wiedererkannte Sessions

identisch. So kann das Anwendungsprotokoll Daten aus mehreren Sitzungen korrekt assoziieren.

isIdentityConfirmed() Gibt an, ob die Identität des Gegenübers in irgendeiner Form bestätigt wurde.

getIdentity() Aufforderung an den Kommunikationspartner, sich zu identifizieren. Diese Funktion liefert entweder die Identität (samt entsprechender Zertifikate) oder einen Fehlerstatus (entweder, weil die Signaturenkette nicht geprüft werden konnte, oder weil der Kommunikationspartner die Identifikation verweigert hat).

identifySelf() Sendet die eigene Identität (samt Zertifikaten) an den Kommunikationspartner.

checkIdentificationPermission(callback) Mit Hilfe der übergebenen Callback-Prozedur kann die Anwendung entscheiden, ob bei einer Anfrage des Kommunikationspartners die eigene Identität übertragen werden soll, oder ob die Anfrage abgelehnt wird.

Auch wenn Mithörer wegen der Verschlüsselung keine Daten gewinnen, kann es dennoch im Interesse der beteiligten Geräte sein, die übertragenen Informationen im Blick zu behalten. Da diese Daten in der Anwendungsschicht übertragen werden, stellen wir im folgenden Kapitel den Ansatz vor, diese Kommunikation durch eine standardisierte Schnittstelle zu führen, so daß an dieser über den Informationsfluß Buch geführt werden kann. Daran anknüpfend kann mit Hilfe von Zugriffsregeln eine Entscheidung getroffen werden, ob eine Kommunikation fortgesetzt, oder ob sie wegen einer Verletzung der Privacy-Präferenzen des Benutzers abgebrochen werden soll.

6 Privacy auf der Applikationsebene

> The digital revolution has changed everything...
> What was once hard to copy is now trivial to duplicate.
> What was once forgotten is now stored forever.
> What was once private is now public.
>
> *Ronald Rivest, 2001 [186]*

6.1 Einleitung

In den vorigen Kapiteln wurden Wege beschrieben, einem Knoten im Netz weitestgehende Anonymität einzuräumen. Auf der Anwendungsebene soll nun entschieden werden, welche Informationen weitergegeben werden.

In unserer Architektur gehen wir davon aus, daß einzelne Knoten verschiedene Dienste anbieten können. Für solche Dienstaufrufe findet man in klassischen Computersystemen viele Entsprechungen, sie sind vergleichbar mit den Netzwerk-RPCs oder den SOAP[1]-Aufrufen der Web-Services.

Für spezielle oder sehr einfache Anwendungen mögen einzelne Diensteaufrufe genügen; komplexere Abläufe erfordern jedoch mehrfache Interaktion mit dem Diensteanbieter. Um für solche Abläufe eine Struktur vorzugeben, ist es in unserer Architektur möglich, diese in Form von einfachen Skripten mitzuliefern. So ist es möglich, für spezielle Anwendungen nach wie vor die Primitive einzeln aufzurufen (was letztendlich einem selbst erstellten Skript entspricht) – oder aber für Standardaufgaben die vorgefertigten Skripte zu benutzen.

Die Bewertung des Informationsflusses und darauf basierend das Treffen der Entscheidung, ob das Programm fortgesetzt, abgebrochen oder die Fortsetzung vom Benutzer bestätigt werden soll, ist ebenfalls nicht einfach zu treffen. Verschiedene Möglichkeiten und Ansätze werden vorgestellt.

6.2 Vorarbeiten

Die meisten Vorarbeiten zum Thema Privacy sind aus dem Forschungsgebiet der Sicherheit entwachsen. Privacy wurde (und wird) häufig nur als Spezialfall der Vertraulichkeit behandelt (so schreiben beispielsweise Myers und Liskov [161]: „privacy,

[1] SOAP: Simple Object Access Protocol

confidentiality, and secrecy will be considered synonymous“), welche über Mechanismen der Zugriffskontrolle implementiert wird: Die Technik zur Realisierung ist dieselbe, lediglich die Motivation ist eine andere. So ist es nicht weiter verwunderlich, daß viele Arbeiten auf den Ideen und Prinzipien verschiedener Zugriffs-Policies aufbauen.

6.2.1 Zugriffskontrolle

Einen Überblick über verschiedene Modelle zur Zugriffskontrolle und eine kurze Erklärung der wichtigsten Vertreter findet man in [93, Kapitel 9] und [176, Kapitel 5.2–5.3]; einen konzeptuellen und eher theoretischen Überblick bietet [8, Kapitel 8]. Detailliertere Erklärungen, übersichtlich zusammengestellt und mit Beweisen und Beispielen versehen, findet man in [29, Teil 3 (Policy)].

Das wohl bekannteste Verfahren zur Modellierung von Zugriffsrechten ist das Bell-LaPadula-Modell (BLP-Modell) [19, 20] . Das Modell stammt ursprünglich aus der militärischen Forschung und diente der Regelung des Zugriffs auf Dokumente und Akten. Die Neuerung des Modells war die mathematische Formulierung und damit die Beweisbarkeit bestimmter Eigenschaften.

Das BLP-Modell betrachtet Klassifizierungen (levels L) und Kategorien (categories C). Die Klassifizierung ist eine vollständig geordnete Liste von Bezeichnern für Freigaben (z. B. öffentlich, vertraulich, geheim, streng geheim). Über dieser Liste ist der Vergleichsoperator $<$ definiert: $l_1 < l_2$ (mit $l_1, l_2 \, \epsilon \, L$) bedeutet hierbei, daß l_1 eine Freigabe mit weniger Privilegien als l_2 ist. Die Kategorien dienen der weiteren Einteilung der Dokumente, beispielsweise nach Projekten oder Abteilungen. Jeder Benutzer und alle Dokumente erhalten eine Sicherheitsstufe (security level), bestehend aus einer Klassifizierung und einer Menge von Kategorien.

Zur Bestimmung der Zugriffsrechte benutzt das Modell eine Vergleichsoperation: Eine Sicherheitsstufe (L, C) *dominiert* über einer Sicherheitsstufe (L', C') (kurz: (L, C) *dom* (L', C')) dann und genau dann wenn $L' \leq L$ und $C' \subseteq C$. Das Modell erlaubt per Definition alle möglichen Kombinationen von Kategorien (d. h. $C \in \mathcal{P}(categories)$), daher induziert der Ordnungsoperator *dom* eine Ordnung als Verband (lattice; siehe A.3 auf Seite 131) über die Sicherheitsstufen [72].

Für den Lesezugriff gilt folgende Bedingung (simple security property): Eine Person mit Sicherheitsstufe P hat Zugriff auf ein Objekt mit Sicherheitsstufe O wenn gilt: P *dom* O. Umgekehrt wird Schreibzugriff dann gewährt, wenn gilt: O *dom* P (*-property). Informell ausgedrückt können Personen also nur Dokumente lesen, die gleich oder kleiner ihrer eigenen Sicherheitsstufe sind (no read-up); Dokumente, die sie verfassen, erhalten eine Sicherheitsstufe, die gleich oder höher ihrem eigenen Status ist (no write-down). Dadurch können neue Erkenntnisse in der Hierarchie nur nach oben wandern – es kann also zu keinen Informationslecks kommen.

Das BLP-Modell wurde in der Literatur oft diskutiert und auch kritisiert, denn es lassen sich keineswegs alle Anwendungsszenarien mit ihm modellieren. Dies ist nicht weiter verwunderlich, wenn man sich den militärischen Ursprung des Modells ins Gedächtnis ruft. Um dieses Mißverständnis zu vermeiden, wird das BLP-Modell als *confidentiality policy system* bezeichnet – im Gegensatz zu den *integrity policy systems*, deren Fokus weniger auf der präzisen Modellierung des Datenflusses, sondern dem Erhalt der Datenintegrität liegt [30]. Lipner sagt hierzu in [155] sinngemäß, daß es einer Firma relativ egal ist, wer welche Daten modifiziert, solange gewährleistet ist, daß die Veränderungen korrekt sind und Sinn machen (und nicht eine Testversion der verwendeten Software aufgrund eines Programmfehlers die Firmendaten verwüstet).

Zu den Vertretern der Integritätsmodelle gehört das Biba-Modell [28]. Die Mechanismen ähneln dem BLP-Modell: Die Klassifizierung wird nun jedoch nicht mehr als Sicherheitsrang betrachtet, sondern gibt das Vertrauen in die korrekte Funktionsweise wieder (integrity level). Die Integritätsstufen sind auch hier wieder in einem Verband geordnet (die Orginalveröffentlichung kennt im Gegensatz zum BLP-Modell keine Kategorien, eine entsprechende Erweiterung ist jedoch trivial zu bewerkstellingen). Die Regeln für den Zugriff funktionieren analog zum BLP-Modell, jedoch sind die Bedingungen hier genau umgekehrt:

Sei $i(O)$ die Integritätsstufe des Objekts O. Ein Programm P darf ein Objekt O lesen, wenn $i(P) \leq i(O)$. Schreibzugriff und Ausführung sind gestattet, falls $i(O) \leq i(P)$. Ein Programm kann also nur Daten verändern, die den gleichen oder einen geringeren Integritätslevel wie es selbst besitzen. Umgekehrt ist durch die Lesebedingung sichergestellt, daß ein Programm immer mit integeren Daten arbeitet (Daten mit einem höheren Integritätslevel können ja vom Programm nicht verändert werden).

Um die Ansätze des Schutzes vertraulicher Daten mit dem Integritätsschutz zu vereinen, wurden entsprechende hybride Modelle entwickelt. Zu diesen zählt das Chinese-Wall-Modell [35]. Designidee war hier, Datenabfragen bei Interessenskonflikten zu unterbinden. Das Modell ähnelt dem BLP-Modell, jedoch werden als weiterer Zugriffsschutz Konfliktklassen eingeführt. Diese Mengen beinhalten Bezeichner, die nicht gemeinsam auftreten dürfen, weil sie zu einem Interessenskonflikt führen würden.

Weitere relativ verbreitete Schutzmodelle, die insbesondere für den Einsatz in Firmen angepaßt wurden, sind das Modell nach Clark-Wilson [59], das Graham-Denning-Modell [94] (welches die Vererbung von Rechten in klassischen Zugriffsmatrizen modelliert) oder das Harrison-Ruzzo-Ullman-Modell [107].

Im Kontext des Ubiquitous Computing wurden abseits der abstrakten Modelle verschiedene praktische Techniken für die Zugriffskontrolle vorgestellt. In [81] werden Datenströme von Videokameras verschlüsselt zugriffsberechtigten Nutzern zur Verfügung gestellt. Die Informationen für den Zugriff werden in den Datenstrom ein-

gebettet und sind für die Nutzer individuell verschlüsselt, weshalb ein Beobachter nicht entscheiden kann, welcher Benutzer Zugriff hat.

Ebenfalls aus dem Kontext des Ubiquitous Computing rührt diese Arbeit: Gunter et al. [98] übertragen das Prinzip von Zugriffsmatrizen (nach Graham und Denning [94]) auf die Privacyproblematik. Der von den Autoren „Privacy System“ genannte Formalismus erlaubt die Vergabe von Privilegien: Ein Benutzer kann einem anderem erlauben, Daten über ihn zu sammeln oder Objekte mit ihm als Privacy-Bezeichner anzulegen; auch die Weitergabe eines Rechts kann modelliert werden. Die Arbeit schlägt vor, daß in verteilten Systemen diese Rechte mit Hilfe von Techniken des DRM[2] durchgesetzt werden sollen – die DRM-Implementierung ist sozusagen ein Ausführungsmonitor, der den Zugriff auf die Daten überwacht.

6.2.2 Information flow

Die bisher vorgestellten Verfahren dienten lediglich der Spezifikation von Sicherheitsrichtlinien. Für die eigentliche Überwachung der Einhaltung wurden allenfalls Vorschläge gemacht: Im Falle des militärischen Kontexts des BLP-Modells sorgen die Verwalter der Akten für eine ordnungsgemäße Weitergabe; das Verfahren von Gunter et al. empfiehlt den Einsatz eines DRM-Systems. Die Kontrolle, daß eine Richtlinie jederzeit eingehalten wird, entspricht der Überwachung des Informationsflusses (information flow). Um diesen Fluß innerhalb eines Programmes zu beobachten, werden im Allgemeinen zwei grundlegende Ansätze diskutiert: Die Beobachtung des Entropiegehalts jeder Zuweisung sowie das Einhalten von Rechtehierarchien (Sicherheitsstufen geordnet zu einem Verband, weshalb diese Modelle in der Literatur meist als „lattice-based models“ bezeichnet werden). Besonderes Augenmerk gilt hierbei *impliziten Informationsflüssen*, d. h. Rückschlüsse, die ein Außenstehender auf höher privilegierte Variablen durch Beobachtung unprivilegierter Werte ziehen kann. Die beiden Ansätze werden im folgenden kurz angerissen, eine ausführlichere Erläuterung findet man in [93, Kapitel 9.5].

Im ersten Ansatz wird die Entropie als Maß für den Informationsgewinn benutzt. Für eine Variable x, die die Werte $\{x_1, \ldots, x_n\}$ mit den Wahrscheinlichkeiten $p(x_i)$ annehmen kann, ist die Entropie definiert als

$$H(x) = -\sum_{i=1}^{n} p(x_i)\ log_2\ p(x_i)$$

Der Informationsfluß von einer Variable $x \in \{x_1, \ldots, x_n\}$ zu einer Variable $y \in \{y_1, \ldots, y_m\}$ entspricht dem Grad der Mehrdeutigkeit (Transinformation, also die Entropie von x in Abhängigkeit von y):

[2] DRM: Digital Rights Management

$$H_y(x) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\ log_2\ p(x_i|y_j)$$

Da aus dem Aspekt der Geheimhaltung jeder Wert $H_y(x) < 1$ kritisch ist, wird in nahezu allen Verfahren der Ansatz der der Rechtehierarchien verfolgt. Diese gehen zurück auf die Arbeit von Denning [72]. Hier wird die Einhaltung einer Sicherheitspolicy, definiert nach einer Variante des BLP-Modells (ohne Kategorien), überwacht. Die Arbeit betrachtet *statische Bindungen* von Rechten an Konstanten, als auch *variable Bindungen*, nämlich die Übertragung von Rechten bei der Operation mit variablen Objekten. Hierbei wird die Einhaltung der Rechtehierarchie überwacht. Ein formaler Nachweis der Korrektheit der Arbeit wurde von Volpano et al. [229] mit Hilfe von Typbarkeits-Regeln durchgeführt. Als Weiterentwicklung ihrer ersten Veröffentlichung zeigten Denning und Denning [73], daß eine statische Analyse des Programmcodes explizite und implizite Verstöße gegen die Sicherheitsrichtlinien aufzeigen kann.

Eine alternative Betrachtungsweise des durch einen Verband beschränkten Informationsflusses ist die *Nichtbeeinflussung* (noninterference) [92]. Im Gegensatz zu den Definitionen des BLP-Modells wird der Informationsfluß nicht explizit verboten, vielmehr ist er eine direkte Konsequenz der folgenden Nichtbeeinflussungs-Bedingung:

Ein Programm C wird als Funktion betrachtet, welche einen Startzustand in einen Endzustand überführt. Der Startzustand eines Programms $C(s)$ sei $s = (s_h, s_l)$, wobei s_h die privilegierten (für den Benutzer klassifizierten) und s_l die sichtbaren Variablen seien (in der Literatur wird von *high values* und *low values* gesprochen). Die Äquivalenzrelation $=_l$ bezeichnet identische Programmaufrufe, was bedeutet: $s =_l s' \ \Leftrightarrow s_l = s'_l$. Die Relation $\approx_l$ bezeichnet die Fähigkeit des Angreifers, die Ausgabewerte zu beobachten (ein Angreifer kann unter Umständen nicht alle unprivilegierten Variablen beobachten; im worst case entspricht $\approx_l$ der Relation $=_l$). Damit gilt ein Programm als sicher, wenn gilt:

$$\forall s_1, s_2 :\ s_1 =_l s_2\ \Rightarrow\ C(s_1) \approx_l C(s_2)$$

Informell ausgedrückt bedeutet dies, daß eine Veränderung der privilegierten Variablen s_h keine Veränderung der unprivilegierten Variablen zur Folge haben darf. Daraus folgt, daß es weder explizite noch implizite Informationsflüsse geben kann, die klassifizierte Informationen preisgeben – was der simple security property des BLP-Modells entspricht.

Eine Übersicht über die auf der Arbeit von Denning basierenden Weiterentwicklungen findet man in [191]. Der Fokus der Entwicklungen liegt auf der statischen

Überprüfung, da nur auf diese Weise die Verletzung der Sicherheits-Policy ohne Seitenkanal behandelt werden kann: Erfolgt ein Programmabbruch zur Laufzeit, erkennt der Angreifer, daß seine Eingabe eine Sicherheitsverletzung provoziert haben muß. Dadurch gewinnt er indirekt Informationen über privilegierte Daten. Die einzige Möglichkeit, diesem Problem zu begegnen, besteht darin, die Ausführung von Programmen zu verweigern, wo es zu einem solchen Fall kommen könnte.

Es gibt mehrere Implementierungen, die eine solche statische Prüfung vornehmen. Viele wurden mit Hilfe von Theorem-Beweisern realisiert (z. B. [87, 157]). Besonders hervorzuheben ist JIF (Java Information Flow; früher: JFlow) [2], eine Erweiterung der Programmiersprache Java, um zur Compilezeit eine solche Überprüfung vornehmen zu können [160]. JIF zählt zu den Typen-basierten Systemen: Variablen erhalten neben dem Datentyp eine weitere Typinformation, welche die Sicherheitsklassifizierung repräsentiert. Widersprüche zur Policy führen beim Übersetzen zu inkompatiblen Typzuweisungen.

Ein Problem, mit dem sowohl statische Prüfer als auch Laufzeit-Monitore zu kämpfen haben, ist der sogenannte *Label Creep*: Bei jeder Variablenzuweisung „erbt“ die Variable die Sicherheitsannotation der für die Zuweisung verwendeten Werte. Noch stärker ist der Effekt von bedingten Programmflüssen. Dies kann vor allem bei Laufzeitsystemen zu einer stetigen Verbreitung von Labels führen, welche sich übermäßig restriktiv auswirken.

Alternativ zu diesem Ansatz lassen sich illegale Datenflüsse auch mit Hilfe von Programmabhängigkeitsgraphen aufspüren [103] (in dieser Arbeit wurde ebenfalls die Programmiersprache Java entsprechend erweitert). Der Vorteil hierbei ist, daß einige Fehlalarme, die bei typbasierten Systemen auftreten können, vermieden werden. Aufgrund dieses Vorteils verfolgen auch andere Autoren (z. B. [122]) diesen oder ähnliche Ansätze. Diese detaillierte Analyse fordert jedoch einen drastisch höheren Aufwand an Speicher und Rechenzeit.

O'Neill et al. [171] diskutieren das Problem von Benutzereingaben, durch welche der Programmablauf nichtdeterministisch wird. Das Programmodell macht Ausgaben über verschiedene Kanäle an unterschiedlich priorisierte Benutzer; die Arbeit erweitert die Sicherheitsdefinition, daß keine Information über das Verhalten eines privilegierten Benutzers an einen nichtprivilegierten fließen darf.

In [161] erörtern die Autoren von JIF die Nutzung des Systems für Privacy-Applikationen. In ihrem Szenario mißtrauen sich zwei Parteien (Principals) – es gibt Informationen, die der eine vor dem anderen geheimhalten möchte; der Programmcode wird von beiden Parteien verifiziert und läuft anschließend auf einer vertrauenswürdigen Plattform. Diese JIF-Erweiterung entspricht der Erweiterung der Rechtehierarchie um Kategorien im BLP-Modell. Durch eine explizite Declassify-Anweisung im Programmcode kann jede der beiden Parteien einen Informationsfluß der eigenen Daten zur jeweils anderen Partei zulassen, obwohl dieser nicht der Policy entsprechen würde.

Hayati und Abadi [108] bauen auf dieser Arbeit auf, nutzen allerdings die Kategorien nicht für Personen, sondern schlagen vor, hiermit Verwendungszwecke zu modellieren. Diese könnten dann mit einer separaten P3P-Policy [65] abgeglichen werden. Die Autoren räumen allerdings ein, daß dieser Ansatz lediglich als Unterstützung für eine Funktionsgarantie (assurance) dienen kann, aber kein Beweis hierfür ist.

Jiang et al. [125] beschreiben, wie in einem kontrollierten Informationsfluß Privatsphäre erhalten werden kann. Hierzu fordern sie die Einhaltung ihres „Principle of Minimum Asymmetry“, was bedeutet, daß es kein Ungleichgewicht zwischen den kommunizierenden Parteien geben darf. Hierdurch versuchen sie, sozialverträgliche Privacyziele und -anforderungen zu beschreiben.

In der Arbeit von Hong und Landay [118] werden sämtliche Daten mit Privacy Tags versehen (Signaturen sollen die nachträgliche Veränderung des Privacy Tags verhindern). Als Datenspeicher kommen sogenannte „Info Spaces“ zum Einsatz, wo die Daten kontrolliert abgelegt werden. Diese Middleware sorgt für ordnungsgemäße Weitergabe und die Löschung der annotierten Daten. Beim Empfang von Daten von einem anderen Info Space wird das Privacy Tag überprüft; falls ein Policyverstoß vorliegt, wird der ursprüngliche Eigner der Daten informiert.

6.2.3 Beschreibungssprachen und Ontologien

Um die Beschreibung von Privacy-Präferenzen zu realisieren, wurden in der Literatur verschiedene Anläufe genommen. Im Vergleich zu den Anforderungen an ein reines Zugriffsschutzverfahren sind diese hierbei deutlich höher, da der Zugriff vom Kontext und verschiedensten Bedingungen abhängen kann.

Der bekannteste Versuch, Privacyziele mit einer Beschreibungssprache zu erfassen, ist wohl die *Platform for Privacy Preferences* (P3P) [66, 65]. Mit P3P können Webseiten maschinenlesbar beschreiben, welche Daten sie erfassen, wie lange sie gespeichert werden, und wie die Daten verwendet und an wen sie weitergegeben werden. Benutzer sollten in einem entsprechenden Browser wiederum ihre Präferenzen einstellen; wegen der Maschinenlesbarkeit können die Browser die Benutzerpräferenzen automatisch mit den Angaben der Webseite abgleichen und gegebenenfalls den Benutzer warnen.

P3P-Beschreibungen werden als XML-Dokumente verfaßt. Die Details können im Standard eingesehen werden, das Kernelement `POLICY` sei im folgenden grob vorgestellt. Es beschreibt eine Regel durch die Angabe folgender Elemente:

- `ENTITY`: Textuelle Beschreibung der (juristischen oder natürlichen) Person, welche die von der Policy beschriebene Anwendung betreibt.
- `ACCESS`: Aussage, welche personenbezogenen Daten gespeichert werden. Hierfür stellt der Standard eine Menge von vordefinierten Beschreibungen bereit:

Kontaktinformation, zusätzliche Information (wie z. B. der Lieferstatus einer Bestellung), beides oder keines von beidem.

- `DISPUTES-GROUP`: Enthält mindestens ein `DISPUTES`-Element. Diese beinhalten Kontaktinformationen für Nachfragen oder Verstöße gegen die Policy.
- `STATEMENT`: Aussage, wie mit den erwähnten Daten verfahren wird. Es können mehrere solche Elemente auftreten.

Das `STATEMENT` besteht wiederum aus folgenden Elementen:

- `PURPOSE`: Enthält ein oder mehrere Elemente, die den Grund für die Datenspeicherung beschreiben. Beispielsweise besagt `current`, daß Daten kurzfristig gespeichert werden (z. B. um Suchergebnisse vorzuhalten), `admin` bedeutet, daß die Daten aus administrativen Gründen gesammelt werden, etc.
- `RECIPIENT`: Beinhaltet ein oder mehrere Elemente, welche die Datenweitergabe spezifizieren. `ours` besagt z. B., daß die Daten nicht weitergegeben werden, `delivery` bedeutet eine Weitergabe zu Auslieferungszwecken, etc.
- `RETENTION`: Beschreibung, wie lange die Daten vorgehalten werden. Die vordefinierten Werte beschreiben Zeiträume wie „keine persistente Datenhaltung“ über „gesetzliche Vorgaben“ und „übliche Geschäftspraktiken“ bis hin zu einem unbestimmten Zeitraum.
- `CONSEQUENCE` (optional): Ein Freitextfeld, das den Grund der Datenhaltung näher beschreibt.

Der P3P-Ansatz provozierte reichlich Kritik [64, 51]. Zum einen wurde dem Konzept vorgeworfen, daß es für den durchschnittlichen Benutzer zu undurchsichtig und zu komplex sei und er deshalb nicht in der Lage wäre, seine Präferenzen entsprechend einzustellen bzw. die Warnungen des Systems richtig zu interpretieren. Das schwerwiegendste Problem ist jedoch die Tatsache, daß eine P3P-Policy vom Restsystem vollkommen entkoppelt ist: Es handelt sich quasi um ein Versprechen des Betreibers, sich entsprechend zu verhalten – daß dem dann auch tatsächlich so ist, ist nicht kontrollierbar.

Trotzdem wird P3P in verschiedenen Arbeiten verwendet. So wird in [149] P3P um Beschreibungskonstrukte erweitert, um Sensor-, Orts- und Positionierungsinformationen in einer ubiquitären Umgebung zu erfassen. Digitale Signaturen solcher „privacy contracts“ sollen die Einhaltung der Angaben beglaubigen.

[25] integrieren P3P-Präferenzen in ihr Trust-System, um so beim Aushandeln von Trust-Beziehungen die weitergegebenen Daten zu limitieren. Auch das weiter oben kurz angesprochene Verfahren von Gunter et al. [98] benutzt eine an P3P angelehnte Beschreibungssprache zur Spezifikation der Privacy-Richtlinien.

Die *Enterprise Privacy Authorization Language* (EPAL) [202, 10] hat das Ziel, den Schutz personenbezogener Daten innerhalb eines Unternehmens (und auch über

Unternehmensgrenzen hinweg) darzustellen und eine Möglichkeit zur Durchsetzung des Schutzes zu bieten. Die Einhaltung der Richtlinien soll entweder mit Hilfe direkter EPAL-Unterstützung in den einzelnen Applikationen erfolgen, oder mit Hilfe eines Anwendungsmonitors, welcher der Schnittstelle einer nicht-EPAL-sensitiven Anwendung vorgeschaltet wird. Die Definitionen sind wesentlich feingranularer als die von P3P – hier wurden nur Defaultwerte von Archetypen zur Verfügung gestellt. Um eine sehr feine Definition für möglichst viele Anwendungen zu erreichen, ist eine EPAL-Definition zweigeteilt: In einem Vokabular-Teil werden zunächst die verschiedenen Elemente definiert, über die dann im Policy-Teil Regeln verfaßt werden.

Eine einzelne EPAL-Policy besteht aus ähnlichen Angaben wie eine P3P-Policy: User category (Kategorie, wo die Daten verwendet werden), action (was mit den Daten getan wird), data category (Kategorisierung, um was für Daten es sich handelt, also z. B. Benutzerdaten, etc.), purpose (der Verwendungszweck), condition (wann die Regel greift) und obligation (zwingende Handlungsfolgen wie beispielsweise die Löschung nach einer bestimmten Zeit). Die hierfür verwendeten Angaben stammen aus den Definitionen des Vokabularteils.

Stufflebeam et al. [218] versuchen, P3P und EPAL anhand eines Anwendungsbeispiels zu vergleichen. Hierbei fiel insbesondere die leicht unterschiedliche Ausrichtung der Beschreibungssprachen auf: EPAL ähnelt beinahe den Regeln eines Zugriffskontrollsystems, wohingegen P3P auf einer höheren, deskriptiven Ebene bleibt. Allgemein ist P3P deutlich weniger ausdrucksstark als EPAL.

Um in der Lage zu sein, eine Privacy Policy aufzustellen, muß zunächst das Anwendungsumfeld und die damit verbundenen Risiken evaluiert werden. Das Privacy Risk Model von Hong et al. [119] bietet für die Designkriterien eine Systematik. Diese dient allerdings nur dem Entwurfsprozeß, da es kein maschinenlesbares Äquivalent gibt.

Ein Problem ist die Organisation und Kategorisierung von personenbezogenen Daten; diese ist in den beschriebenen Arbeiten entweder vorgegeben oder muß wie bei EPAL für jede Spezifikation separat erstellt werden. Die Forderung nach einer Privacy-Ontologie [138] ist deshalb nicht überraschend. Für den Security-Bereich gibt es verschiedene Arbeiten (wie die Ontologien von [139]), nicht jedoch für das Thema Privacy.

Einige Arbeiten erstellen für Beispiele und Spezialfälle separate Ontologien. [240] benutzt die durch eine Ontologie implizierte hierarchische Gliederung der Daten, um so ein Vergröbern der Daten einfach zu ermöglichen: Je nach Anwendung und benutztem Dienst werden entsprechend diesen Regeln Angaben über Position, etc. mit einer bestimmten Menge an Unschärfe versehen, um so die preisgegebene Datenmenge zu reduzieren.

6.3 Bewertung

6.3.1 Information flow

Die in den Kapiteln 6.2.1 und 6.2.2 vorgestellten Verfahren lassen sich nur bedingt auf die hier vorliegende Problemstellung übertragen. In den Vorarbeiten, die die Privacy-Problematik ansprechen, wird Privatsphäre mit Geheimhaltung gleichgesetzt. In unserem Szenario geht es nicht zwingend um die Geheimhaltung sämtlicher Privacy-relevanter Daten, sondern um eine Kontrolle des Flusses dieser Informationen – aus diesem Grund lösen die beschriebenen Mechanismen das Problem nicht.

Die in [161] verfolgte Idee setzt voraus, daß beide Parteien das Programm verifizieren, mit entsprechenden Privacy-Typisierungen versehen und es anschließend auf einer vertrauenswürdigen Plattform zur Ausführung bringen. Auch diese Annahme ist in unserem Szenario nicht gegeben.

Die statische Überprüfung, wie sie in den obigen Arbeiten favorisiert wird, impliziert, daß sich die Policy zur Laufzeit nicht mehr ändern darf. Dies ist bei einer interaktiven Anwendung nicht unbedingt gegeben, Verhalten des Gegenübers oder nicht modellierte Faktoren (z. B. der Empfang einer authentischen Nachricht eines Bekannten, der eine Empfehlung für das gerade laufende Skript ausspricht) können die Präferenzen eines Nutzers auch zur Laufzeit beeinflussen.

Da bei den Arbeiten in 6.2.2 die Geheimhaltung, also das rigorose Verbot von Datenflüssen im Vordergrund stand, brachte eine statische Überprüfung einen Performancevorteil: Einmal (aufwendig) überprüft konnte das Programm anschließend ohne Performanceverlust beliebig oft ausgeführt werden. Die individuellen Präferenzen der Nutzer, die sich sogar von Aufruf zu Aufruf ändern können, erlauben diesen Tradeoff nicht.

Ein Nachteil einer Überprüfung zur Laufzeit ist der Informationsfluß, der aus einem Programmabbruch aufgrund einer Policyverletzung entsteht. Bei der Betrachtung von Seitenkanälen wird davon ausgegangen, daß das entsprechende Programm automatisch und beliebig oft ausgeführt werden kann. Dies ist in unserem Szenario nicht der Fall, da der Benutzer das Programm zur Ausführung bringt; allein die manuelle Interaktion limitiert die Zahl der Programmläufe und die Ausführungsgeschwindigkeit so stark, daß bekannte Angriffe auf solche Seitenkanäle nicht genug Informationen sammeln können, um eine Aussage treffen zu können. Der Abbruch des Programms zur Laufzeit offenbart einem Angreifer also lediglich, daß an diesem Punkt bestimmte Präferenzen in Bezug auf die Privatsphäre verletzt wurden. Dieser Informationsgewinn ist in Relation zum Szenario vernachlässigbar, der Usability-Gewinn ist hingegen erheblich. Aus diesem Grund nehmen wir den Programmabbruch zur Laufzeit bewußt in Kauf.

6.3.2 Beschreibungssprachen und Ontologien

Unser Ziel ist es, den Informationsfluß eines Anwendungsprotokolls zu überprüfen, um so die Kumulation von weitergegebenen Daten zu beobachten und auf Verstöße gegen Privacy-Präferenzen zu überprüfen. Um die Anzahl von Fehlalarmen gering zu halten, sollte die Menge der beobachteten Merkmale möglichst feingranular sein.

Für dieses Ziel ist die Ausdrucksstärke P3P zu gering: Ein P3P-Dokument beschreibt die Privacy-Policy einer gesamten Webseite und verwendet ein sehr einfaches Vokabular, das nur relativ allgemeine Aussagen treffen kann. Erschwerend kommt hinzu, daß die Signatur (und damit verbunden ein entsprechendes Zertifikatsmanagement) von einer vertrauenswürdigen Instanz nötig ist, um der P3P-Beschreibung eine verbindliche Aussagekraft zu verleihen. Die Vokabulardefinition von EPAL ist zwar sehr ausdrucksstark, jedoch erkennt man bereits hier, daß EPAL für einen anderen Einsatzzweck, nämlich die innerbetriebliche Kontrolle der Daten, entworfen wurde. Sämtliche Regeln über die Weitergabe von Daten müssen vorformuliert werden. Dies ist sicherlich ein valider Ansatz für die Datenflußkontrolle in Behörden und großen Konzernen, allerdings eignet er sich kaum für die spontanen und häufig nur kurz andauernden Verbindungen von mobilen Geräten.

Der Einsatz von Ontologien sowohl zur Organisation der Daten, zur Kategorisierung bezüglich ihrer Sensitivität als auch als Hilfsmittel zur Vergröberung der Daten erscheint sinnvoll. Die Verwendung derselben Datenstruktur, um verschiedene Ziele zu erreichen reduziert die erforderliche Arbeit bei der Einrichtung.

6.4 Simple Service Description and Scripting Language

Wie in den vorigen Kapiteln gezeigt wurde, läßt sich der Grad der Privatsphäre auf den unteren Netzwerkschichten gut kontrollieren, da diese ein vorgegebenes Format besitzen. Anwendungsprotokolle hingegen besitzen keine solche Struktur, welche aber erforderlich ist, um eine generische Überprüfung des Informationsflusses modellieren zu können. Die in diesem Abschnitt vorgestellte Skriptsprache soll diese Strukturierung einführen. Für reale Implementierungen ist es natürlich genauso möglich, die hier vorgestellten Konzepte auf bekannte Programmiersprachen zu übertragen; um den Fokus auf der Privacy-Problematik zu halten, werden wir die Konzepte an dieser einfachen Skriptsprache vorstellen.

Wir nehmen an, daß es in der ubiquitären Umgebung *Diensteanbieter* gibt, die den ubiquitären, mobilen Geräten erlauben, bestimmte Dienste in Anspruch zu nehmen. Diese Dienstleistungen können einfache Funktionsaufrufe sein (beispielsweise ein öffentlicher Temperatursensor, der die aktuell gemessene Temperatur liefert); häufig werden diese Dienste aber komplexere Abläufe sein (z. B. der Kauf einer Kinokarte nach dem Studieren des aktuellen Programms). Solche komplexeren Abläufe lassen sich aber in einfachere Primitive unterteilen: Programmteile, die einzig die Daten

des mobilen Geräts oder die Interaktion des Benutzers benötigen, sowie Abschnitte, welche Interaktion mit dem Rechner des Diensteanbieters erfordern.

Dieses Client-Server-Modell wird in der hier vorgestellten Skriptsprache benutzt: Der Client (das mobile Gerät) erhält vom Server (dem Diensteanbieter) das Skript und führt dieses aus. Dabei kann es beim Interpretieren von jedem Befehlsschritt Zugriffsrechte und Privacy-Policies überprüfen. Die Interaktion mit dem Diensteanbieter entspricht einem RPC[3] aus klassischen verteilten Systemen. Diese definierten *Diensteschnittstellen* bieten mehrere Vorteile: Zum einen sind dies die einzigen, wohldefinierten Punkte, an denen ein Informationsaustausch mit dem Diensteanbieter stattfindet; zum anderen eröffnet dies die Möglichkeit, auf den Client-Geräten alternative Skripte einzusetzen, welche alternative Anwendungen implementieren oder die Dienste mehrerer Diensteanbieter kombinieren.

Dieses Kapitel stellt zunächst die Skriptsprache und die Beschreibung der Diensteschnittstellen vor. Anhand dieser Sprache zeigen wir, wie es möglich ist, eine obere Schranke für den Informationsgewinn des Diensteanbieters beim Ausführen eines Skriptes anzugeben. Anhand dieser Schranke kann der Benutzer (oder ein entsprechendes Softwaremodul) entscheiden, wie lange mit der Ausführung eines Skriptes fortgefahren werden soll.

6.4.1 Voraussetzungen und Annahmen

Während des Ablaufs eines Skripts greift dieses auf Daten des Benutzers zu; die Daten mit einem bestimmten Privacy-Wert werden im folgenden als *Privacy-relevante Daten (PRD)* bezeichnet. Diese Daten dienen als „automatische Eingabewerte“, d. h. sie ersetzen Benutzerinteraktion durch automatische Vorgaben. Um die Privatsphäre des Nutzers zu wahren, wird der Informationsfluß dieser Werte verfolgt.

Definition 6.1 (Privacy-annotierte Variablen) *Die in einem Gerät abgelegten Privacy-relevanten Daten werden in Variablen mit wohlbekannten Bezeichnern gespeichert, um eine automatische Zuordnung zu ermöglichen. Zur Unterscheidung von anderen Variablen werden diese Variablen im folgenden* Privacy-annotierte Variablen (PAV) *genannt.*

Es ist möglich, daß manche Daten weitere Informationen als Teilmenge enthalten. Ein Beispiel für eine solche Abhängigkeit ist die „machine-readable zone“ des Personalausweises, die als Ziffernfolge den Name und das Geburtsdatum enthält [124]. Für die folgenden Betrachtungen gehen wir davon aus, daß dies in den gespeicherten PAV nicht der Fall ist:

[3] RPC: Remote Procedure Call. Prozeduraufruf auf einem anderen Rechner über das Netzwerk in verteilten Systemen.

Definition 6.2 (Unabhängigkeit der PAVs) *Wir nehmen an, daß die PAVs* voneinander unabhängig *sind, d. h. daß die Kenntnis einer PAV keine Schlußfolgerungen auf weitere PAV zuläßt.*

Das Skript wird auf dem Gerät des Benutzers ausgeführt. Kommunikation mit dem Diensteanbieter findet über die definierten Schnittstellen statt. Daher gilt:

Definition 6.3 (Informationsfluß-Möglichkeiten) *Die Serviceaufrufe sind die einzigen Stellen im Skript, an denen Privacy-relevante Informationen fließen können.*

In den vorigen Kapiteln dieser Arbeit wurde argumentiert, daß eine anonyme Kommunikation gewährleistet werden kann. Wir gehen bei der Betrachtung der Skriptsprache davon aus, daß zwischen zwei Diensteaufrufen von diesen Techniken Gebrauch gemacht wird:

Definition 6.4 (Anonymität der Diensteaufrufe) *Die einzelnen* Diensteaufrufe erfolgen anonym. *Die einzigen Informationsquellen für Diensteanbieter sind der Name des aufgerufenen Dienstes sowie dessen Parameter.*

6.4.2 Definition der Skriptsprache

Die Skriptsprache enthält die Konstrukte einer klassischen WHILE-Sprache (siehe z. B. [166]), erweitert um Befehle für die Ein- und Ausgabe und den Aufruf der Dienste. Um die Übersichtlichkeit zu erleichtern, wurde auf weitere Sprachkonstrukte wie Prozeduren verzichtet; eine entsprechene Erweiterung ist jedoch problemlos möglich.

Die Sprache besitzt folgende Syntax:

script	::=	script *scriptname block*
block	::=	{ *statements* }
statements	::=	*statement* | *statement* ; *statements*
statement	::=	skip | letvar *varname* := *expression* |
		if *bexpression* then *block* [else *block*] |
		while *bexpression* do *block* |
		call | output(*expression*)
expression	::=	*aexpression* | *bexpression* |
		(*expression*) | *input* | *call*
call	::=	call *servicename*({ *expression* { , *expression* } })
input	::=	input()
aexpression	::=	*varname* | *constant* | *PAV* |
		aexpression [+,-,*,/] *aexpression* |
		$-$*aexpression*
bexpression	::=	true | false | not *bexpression* |
		aexpression [=,!=,<,>] *aexpression*

		bexpression [and,or,xor] *bexpression*
scriptname	::=	*identifier*
servicename	::=	*identifier*
varname	::=	*identifier*
PAV	::=	PAV.*identifier*
identifier	::=	*alpha* { *alpha* }
constant	::=	" { *alpha* }" | true | false | [-]{*num*}[.*num*{*num*}]
alpha	::=	A | B | ... | Z | a | b | ... | z | _
num	::=	0 | 1 | ... | 9

Neben den bekannten Programmkonstrukten gibt es Anweisungen zum Aufruf von Diensten (call), zur Ausgabe von Informationen (output) und zur Benutzerinteraktion (input). Die genaue Ausführung der Ein- und Ausgabe ist (sofern überhaupt möglich) vom jeweiligen Gerät abhängig – auf die Problematik generischer Ein- und Ausgabeschnittstellen soll hier nicht weiter eingegangen werden; aus diesem Grund sind die Befehle *input*() und *output*(...) in der Skriptsprache nur grob umrissen und für eine reale Implementierung als Platzhalter zu verstehen.

6.4.3 Definition der Diensteschnittstellen

Die Syntax der Diensteschnittstellen entspricht weitestgehend normalen Prozeduren, mit dem Unterschied, daß hier zwischen Parametern und PAV-Angaben unterschieden wird:

interface	::=	def *servicename*({ *param* ... })
param	::=	*varname* | *PAV*
varname, *PAV*		(siehe oben)

Das bedeutet, daß Dienste, deren Parameter nur PAV enthalten, direkt aufgerufen werden können. Der Fluß an privacy-relevanten Informationen ist so direkt ersehbar und kann automatisch verarbeitet werden.

Durch die Verknüpfung zwischen Parameter und PAV ist es nicht möglich, die Privacy-Richtlinien des Benutzers zu umgehen; andere (Meta)sprachen wie beispielsweise WSDL[4] [56] erlauben es lediglich, die Typen der Parameter festzulegen. Eine von der Schnittstellendefinition losgelöste Privacy-Definition ohne Kontrolle der eigentlichen Aufrufe ist jedoch lediglich eine Absichtserklärung – ähnlich der P3P-Dokumente für Webseiten [65].

Diensteschnittstellen, die freie Variablen beinhalten, können nicht automatisch aufgerufen werden, sondern benötigen entweder entsprechende Benutzereingaben oder die Einbettung in ein geeignetes Skript. Letzterer Fall wird im folgenden Kapitel erörtert.

[4] WSDL: Web Service Description Language

6.4.4 Verfolgung des Informationsflusses

Das hier vorgestellte Verfahren erlaubt es, eine obere Schranke für den Informationsfluß zum Diensteanbieter zu bestimmen. Diese Schranke wird in Form von Mengen von PAVs repräsentiert. Ist eine PAV in einer solchen Menge enthalten, bedeutet dies, daß der Diensteanbieter Information über diese PAV gewinnen kann. Dies kann im ungünstigsten Fall der exakte Wert sein.

Um dem Problem des Label Creep, also der schleichenden Fortpflanzung von Annotationen zu begegnen, teilen wir den Informationsfluß in verschiedene Typen, welche dann gesondert behandelt werden.

In Programmen kann es zu zwei Arten von Informationsfluß kommen:

- *Expliziter Fluß* durch direkten Datenaustausch bei Variablenzuweisungen, etc.
- *Impliziter Fluß* durch Seitenkanäle wie bedingte Verzweigungen, Schleifen, Programmabbruch, etc.

Expliziter Informationsfluß

Expliziter Informationsfluß kommt in der oben beschriebenen Sprache nur bei Variablenzuweisungen vor. Im folgenden sei:

P_i — Menge aller PAVs, die in die Berechnung der Variable i eingeflossen sind, oder kurz der *Privacy-Wert der Variable i*. Dabei gilt für alle PAVs p:

$$\forall p : PAV \mid p \notin P_i \Rightarrow \text{Das Wissen über } i \text{ verrät nichts über } p \tag{6.1}$$

Bei jeder Wertebelegung einer Variable i muß der korrespondierende Privacy-Wert P_i die Vereinigung aller Privacy-Werte der verwendeten (terminalen) Ausdrücke ($PV(i)$) enthalten. Dies sind:

$$\bar{P_i} \quad = \quad PV(expr.) := \bigcup_{e \,\in\, expr.} \begin{cases} e & e \text{ ist } PAV \\ P_e & e \text{ ist Variable } (identifier) \\ \emptyset & e \text{ ist Konstante } (constant) \\ P(call) & e \text{ ist Diensteaufruf (call)} \\ input_n & e \text{ ist Benutzereingabe (input)} \end{cases} \tag{6.2}$$

Dies sind nur die Informationen des expliziten Informationsflusses; wir notieren dies daher als $\bar{P_i}$; das endgültige neue P_i muß zusätzlich die Informationen des impliziten Informationsflusses berücksichtigen. Dies betrachten wir im folgenden Abschnitt.

Der Privacywert eines Diensteaufrufs $P(call)$ kann erst nach Berücksichtigung des impliziten Informationsflusses und der möglichen Zusammenführbarkeit von mehreren Diensteaufrufen (siehe Kapitel 6.4.5 auf Seite 105) bestimmt werden.

Über Benutzereingaben können keine weiteren Aussagen getroffen werden, weil hier die Privacy-Annotation fehlt; eine Angabe hierüber im Skript wäre keine vertrauenswürdige Informationsquelle. Die Idee, bei einer Eingabe den Privacywert aller Ausgabebefehle (und damit der dort verwendeten Variablen) seit der letzten Eingabe zu verwenden, greift leider zu kurz, da nicht bekannt ist, welche Anweisungen dem Benutzer gegeben werden; eine böswillige Anzeige „geben Sie zur Sicherheit den letzten Wert nochmals ein“ würde einen solchen Mechanismus aushebeln. Die Alternative wäre die Vereinigung sämtlicher Ausgaben, die das Programm getätigt hat – dies wäre allerdings insbesondere bei umfangreicheren Skripten viel zu restriktiv.

Um die Information nicht völlig zu verwerfen, wird ein spezielles Label $input_n$ mit laufender Nummer n gespeichert. Eine automatische Evaluation ist zwar nicht möglich, doch kann im Fall von Benutzerrückfragen dieses Label genutzt werden, um z. B. die direkt davor getätigten Ausgaben des Programms dem Nutzer als Entscheidungshilfe nochmals anzuzeigen.

Impliziter Informationsfluß

Wir betrachten den impliziten Informationsfluß, der am meisten Rückschlüsse über die PRD zuläßt, nämlich die bedingten Verzweigungen eines Programms. Weitere bekannte Seitenkanäle der Kryptographie, wie z. B. Timing [82] oder Stromverbrauch [143] sind in diesem Szenario nicht relevant, sie zielen auf Tamper-Resistant Devices, die einem Angreifer in die Hände fallen.

Ein weiterer möglicher Seitenkanal ist der Programmabbruch. Im Secrecy-Fall der Informationsflußkontrolle darf keinerlei Information dem Angreifer in die Hände fallen – auch dann nicht, wenn das Programm beliebig oft ausgeführt wird (es wird davon ausgegangen, daß dies automatisiert möglich ist). In unserem Szenario hingegen werden die Skripte nur selten ausgeführt; obendrein ist der Informationsgewinn sehr gering, insbesondere wenn man bedenkt, daß im Gegensatz zur Information Secrecy bewußt in Kauf genommen wird, daß Information fließt – der Fluß soll lediglich überwacht werden.

Wir unterscheiden für den impliziten Informationsfluß zwei Fälle: Den *Programmzustand* und den *potentiellen Programmfluß*. Dies läßt sich an folgendem einfachen Beispiel sehen:

```
1 letvar x := 0;
2 if PAV.age<18 then
3    { letvar x := 1; };
4 call result(x);
```

Die Variable x hängt offensichtlich vom Alter ab, obwohl dies von $\bar{P}_x$ (siehe Formel 6.2 auf Seite 101) nicht erfaßt wird. Im Fall, daß $PAV.age < 18$ ist, wird die Zuweisung in Zeile 3 ausgeführt. An dieser Stelle muß nun $PAV.age$ in die Menge P_x aufgenommen werden, was wir als Abhängigkeit aufgrund des Programmzustands bezeichnen.

Im umgekehrten Fall, also wenn $PAV.age \geq 18$, wird die Zuweisung nicht ausgeführt; trotzdem gewinnt der Diensteanbieter die Information, daß das Alter mindestens 18 Jahre sein muß: Dies folgt aus der Tatsache, daß x nur im anderen Fall verändert wird. Dies nennen wir eine Abhängigkeit vom potentiellen Programmfluß, also von Programmteilen, die bei anderer Wertekonstellation ausgeführt werden könnten (es im konkreten Fall aber nicht sind).

Behandlung des Programmzustands Der Programmzustand sind die (erfüllten) Bedingungen, die zum Erreichen der aktuellen Codezeile geführt haben. Hierzu führen wir PS_{state} ein, einen Stack, der die PAV aller Bedingungen enthält. Beim Eintritt in einen if- oder while-Block wird also $push(PS_{state}, PV(expression))$ ausgeführt.

Am Ende des Blocks wird dieser Zustand wieder mit $pop(PS_{state})$ entfernt. Innerhalb des Blocks muß bei allen Variablenzuweisungen der Programmzustand mit im entsprechenden P_i aufgenommen werden; dies ist die Vereinigung aller Mengen des Stapels:

$$P_{state} = \bigcup_{i \in PS_{state}} i \tag{6.3}$$

Behandlung des potentiellen Programflusses Der kritische Teil ist die Betrachtung der Programmteile, die aufgrund von bedingten Verzweigungen *nicht* ausgeführt werden. Eine Evaluation durch Ausführen dieser Programmteile, um festzustellen, was passiert wäre, ist selbst bei unserer einfachen Sprache nicht möglich, da Benutzerinteraktion und Rückgabewerte von Diensteaufrufen aus Sicht des Algorithmus nicht deterministisch sind. Allgemeiner gesprochen wäre dies ein Programm, welches das Ergebnis eines Programms bestimmen soll; dies entspricht jedoch dem speziellen Halteproblem, welches erwiesenermaßen nicht entscheidbar ist [203, Seite 119] (der Widerspruch läßt sich dadurch herführen, daß das Programm sein eigenes Ergebnis bestimmen soll).

Implizite Informationen können abgeleitet werden, wenn Variablen nicht zugewiesen (wie im Beispiel auf der vorherigen Seite), oder wenn Dienste nicht aufgerufen werden. Dieser Fall tritt bei if-Abfragen auf – dabei wird der jeweils andere Programmzweig nicht ausgeführt; dasselbe gilt aber auch für while-Schleifen, wenn die Abbruchbedingung von vornherein mit $false$ evaluiert. Die gewonnene Information

ist wiederum der Programmzustand P_{state}, der dazu geführt hat, daß die entsprechenden Passagen nicht ausgeführt werden. Im folgenden bezeichnet B den Programmzweig, der zur Ausführung kam, und $\bar{B}$ den Abschnitt, der nicht ausgeführt wurde (wobei B oder $\bar{B}$ leer sein können).

Als obere Schranke für den Informationsfluß betrachten wir alle Befehle von $\bar{B}$, unabhängig davon, ob diese wiederum an Bedingungen geknüpft sind oder nicht. Der Informationsfluß findet beim Evaluieren der Bedingung statt, die folgende Regel muß also beim Eintritt in B ausgeführt werden. Um den Informationsfluß von nicht ausgeführten Variablenzuweisungen zu erfassen, wird bei allen auftretenden Variablenzuweisungen „letvar i“ ausgeführt:

$$P_i = P_i \cup P_{state} \tag{6.4}$$

Im Falle von Diensteaufrufen treffen wir die worst-case-Annahme, daß der Anbieter den ausgelassen Aufruf und den nächsten Diensteaufruf hätte zusammenführen können. Um nicht aufgerufene Dienste zu erfassen, führen wir den Zustand P_{flow} ein. Diese Menge erfaßt analog zu den Variablenzuweisungen den Programmzustand, der zur Verzweigung geführt hat:

$$P_{flow} = P_{flow} \cup P_{state} \tag{6.5}$$

P_{flow} wird beim Verlassen von B gesetzt. Nach dem nächsten tatsächlich durchgeführten Diensteaufruf wird P_{flow} wieder gelöscht (entsprechend der Annahme 6.4, daß Diensteaufrufe nicht zusammengeführt werden können).

Privacy-Werte von Variablen und Diensteaufrufen

Zusammenfassend muß also als Privacy-Annotation bei einer Variablenzuweisung der Privacy-Wert des Ausdrucks (6.2) und der Programmzustand (6.3) zusammengeführt werden:

$$\text{letvar } i := expression \Rightarrow P_i = PV(expression) \cup P_{state} \tag{6.6}$$

Bei einem Diensteaufruf wird somit davon ausgegangen, daß die Informationen über die Parameter, den Programmzustand sowie den potentiellen Programmfluß ableitbar sind:

$$P(call(params)) = \bigcup_{p \in params} P_p \cup P_{state} \cup P_{flow} \cup \{session_n\} \tag{6.7}$$

Nach einem Aufruf wird, wie bereits erwähnt, $P_{flow} = \emptyset$ gesetzt. Das Pseudolabel $session_n$ dient der Verfolgung zusammenführbarer Aufrufe (siehe Kapitel 6.4.5); analog zu $user_n$ ist n ein laufender Zähler, um die einzelnen Aufrufe zu unterscheiden. Um den Datenfluß über mehrere Aufrufe hinweg zu beobachten, wird jeder Diensteaufruf in einem neuen Array $P_{call}[]$ protokolliert: $P_{call}[n] = P(call(params))$

6.4.5 Zusammenführbarkeit mehrerer Diensteaufrufe

Bis dato gingen wir davon aus, daß die Diensteaufrufe voneinander unabhängig und anonym erfolgen. Dies bedeutet, daß der Diensteanbieter die einzelnen Aufrufe nicht einzelnen Knoten zuordnen kann. Ein Zusammenführen kann aber durch folgende Ansätze erfolgen:

- Vergabe einer Session-ID per Rückgabewert, die anschließend bei den folgenden Diensteaufrufen verwendet wird.
- Verwendung ausreichend vieler PAVs in jedem Aufruf, so daß diese eine eindeutige Signatur bilden.

Der erste Fall läßt sich programmtechnisch gut verfolgen – wie am Ende des vorigen Kapitels bereits beschrieben, wird jedem Funktionswert das Privacy-Label $session_n$ angefügt; dies folgt der Worst-Case-Annahme, daß jeder Rückgabewert einer Funktion ein potentieller eindeutiger Identifier ist. Anhand dieser Labels lassen sich im Array P_{call} alle Informationen sammeln, die mit diesem Label verknüpft sind.

Der zweite Fall ist bedeutend komplizierter und läßt sich nicht auf rein algorithmischem Wege beschreiben; einfache Kriterien, um eine Beurteilung zu treffen, können sein:

- Wertebereich der PAV
- Verteilung der Werte

So hätte beispielsweise eine PAV $PAV.weiblich$ einen kleinen Wertebereich („ja“ und „nein“) und eine relativ unkritische Verteilung (ca. 51% der Deutschen sind Frauen). $PAV.geburtsdatum$ hingegen bietet einen deutlich größeren Wertebereich (womit die Chance der Wiedererkennbarkeit steigt). Ebenso kritisch sind PAVs, bei denen manche Werte extrem selten vorkommen.

Erschwerend kommt noch hinzu, daß letzterer Fall von äußeren Gegebenheiten abhängen kann, die in diesem System nicht modelliert sind: So ist beispielsweise der Nachname „Häberle“ in Stuttgart nichts ungewöhnliches, in Hamburg hingegen ein herausragendes Merkmal. Weitere solche Effekte gibt es in verschiedenen Größenordnungen und Ausprägungen.

Um die Ableitbarkeit zu modellieren, benutzen wir eine Funktion $Infer(A, B)$ mit der Signatur $(PAV*, PAV*) \rightarrow [0; 1]$; die Funktion liefert 1 (true), falls davon

ausgegangen wird, daß der Diensteanbieter A und B dem selben Benutzer zuordnen kann.

Betrachtet man nur die Session-Annahme, sieht die Funktion wie folgt aus:

$$Infer(A,B) = \begin{cases} 1 & \exists \text{ Session-Pseudolabel } s \mid s \in A \wedge s \in B \\ 0 & \text{sonst} \end{cases} \tag{6.8}$$

Um alle Informationen zu bestimmen, die ein Diensteanbieter einem Diensteaufruf $P_{call}[i]$ direkt oder indirekt zuordnen kann, bildet man die transitive Hülle über $P_{call}[i]$, $P_{call}[]$ und $Infer(A,B)$:

$$P_{call}[i]^+ = \bigcup \{c \in P_{call}[] \mid P_{call}[i]\ Infer^+\ c\} \tag{6.9}$$

6.4.6 Beispiele

Der Ablauf des beschriebenen Verfahrens soll am folgenden Beispielen verdeutlicht werden.

Speisekarte Eisdiele (1)

An einer Eisdiele befindet sich ein Temperatursensor, welcher die Lufttemperatur mißt. Das Skript soll ab einer gewissen Temperatur auf der Anzeige des mobilen Geräts die Eiskarte anzeigen.

```
1  def aussentemperatur()
2  def werbung_fruchteis()
3  def werbung_milcheis()
4
5  script eiskarte {
6    letvar temp := call aussentemperatur();
7    if (temp > 24) {
8      letvar info := call werbung_fruchteis();
9      output(info);
10     letvar info := call werbung_milcheis();
11     if (not PAV.laktoseallergie) {
12       output(info);
13     }
14   }
15 }
```

Zunächst wird in Zeile 6 die Temperatur abgefragt. Somit könnte es zum ersten Informationsfluß (nach Regel 6.7) kommen:

$P_{call}[1] = \emptyset$ (keine Parameter) $\cup\, P_{state} \cup P_{flow} \cup \{session_1\} = \{session_1\}$

Die Semantik des Rückgabewerts ist der Skriptsprache unbekannt – es könnte sich prinzipiell um einen eindeutigen Identifier handeln; daher wird das Symbol $session_1$ vergeben. Nach Regel 6.6 erhält P_{temp} nun den Privacy-Wert des Aufrufs und von P_{state}:

$P_{temp} = P_{call}[1] \cup P_{state} = \{session_1\}$

Die bedingte Verzweigung in Zeile 7 legt den Privacywert der Verzweigungsbedingung auf den Zustandsstapel:

$push(PS_{state}, \{session_1\}) \Rightarrow PS_{state} = [\emptyset, \{session_1\}]$

Wir nehmen an, daß die Außentemperatur über 24 °C liegt – die Eiskarte soll also angezeigt werden. Die Abfragen der Speisekarte in Zeile 8 wird (nach Regel 6.7) folgendermaßen behandelt:

$P_{call}[2] = \emptyset$ (keine Parameter) $\cup\, P_{state} \cup P_{flow} \cup \{session_2\} = \{session_1, session_2\}$

Damit erhält die Variable $info$ den Privacywert

$P_{info} = P_{call}[2] \cup P_{state} = \{session_1, session_2\}$

Der erneute Funktionsaufruf in Zeile 10 läuft analog ab: Die Variable $info$ wird neu zugewiesen (was ihren alten Privacywert verwirft), und der Funktionsaufruf ist von keinen neuen Nebenbedingungen abhängig. Damit wird der Informationsfluß mit $session_1$, dem Ergebnis der Temperaturabfrage, abgeschätzt: Die Funktionsaufrufe sind von der Temperaturabfrage abhängig, der Diensteanbieter ist potentiell in der Lage, die Folgeaufrufe (Zeilen 8 und 10) mit dem Pseudonym des ersten Aufrufs zusammenzuführen.

An diesem Beispiel sieht man, daß die Abschätzung nur eine obere Schranke darstellt: Normalerweise ist es nicht möglich, anhand der Außentemperatur einen Benutzer wiederzuerkennen. Die Information selbst kommt jedoch aus einer potentiell nicht vertrauenswürdigen Quelle, ebenso wie die Beschreibung des Dienstes. Es ist also nicht möglich, mit Hilfe von weiteren Metadaten eine knappere Abschätzung zu treffen, da diese ja ebenfalls vom (potentiell nicht vertrauenswürdigen) Diensteanbieter stammen.

Speisekarte Eisdiele (2)

Das folgende Skript erfüllt nochmals dieselbe Funktionalität, allerdings wurde hier versucht, unnötige Kommunikation einzusparen. Diese Einsparung hat aber Auswirkungen auf die Privatsphäre des Nutzers:

```
1 def aussentemperatur()
2 def werbung_fruchteis()
3 def werbung_milcheis()
```

```
4
5  script eiskarte {
6    letvar temp := call aussentemperatur();
7    if (temp > 24) {
8      letvar info := call werbung_fruchteis();
9      output(info);
10     if (not PAV.laktoseallergie) {
11       letvar info := call werbung_milcheis();
12       output(info);
13     }
14   }
15 }
```

Der Ablauf ist bis zu Zeile 10 identisch; die Abfrage der Milcheissorten (Zeile 11) ist nun aber in die Allergieabfrage hineingezogen. Beim Betreten dieses Programmzweigs wird folgende Information in PS_{state} gespeichert:

$$push(PS_{state}, \{laktoseallergie\}) \Rightarrow PS_{state} = [\emptyset, \{session_1\}, \{laktoseallergie\}]$$

Damit kann aus dem Abruf des Angebots an Milcheissorten auf das Nichtvorhandensein einer Laktose-Allergie geschlossen werden:

$$\begin{aligned} P_{call}[3] &= \emptyset \text{ (keine Parameter)} \cup P_{state} \cup P_{flow} \cup \{session_3\} \\ &= \{session_1, session_3, laktoseallergie\} \end{aligned}$$

Kinokasse

Dieses Beispiel schildert einen komplexeren Sachverhalt: Ein Kino möchte einen Dienst anbieten, der es Kunden ermöglicht, über ihr mobiles Gerät das Kinoprogramm anzusehen, Details zu einzelnen Filmen abzurufen und auch Kinokarten zu kaufen. Dabei soll das Alter des Kartenkäufers berücksichtigt werden, d. h. es soll nur möglich sein, Kinokarten gemäß der FSK-Freigabe zu erwerben. Die entsprechende Dienstedefinition sieht wie folgt aus:

```
1  def kinoprogramm()
2  def film_details(nr)
3  def film_fsk(nr)
4  def kartenkaufen(filmnr,anzahl)
5
6  script kino {
7    letvar p := call kinoprogramm();
8    while true {
9      output(p);
10     letvar nr := input();
11     letvar fsk := call film_fsk(nr);
```

```
12        if (PAV.age >= fsk) {
13          output(call film_details(nr));
14          output("1 − Karten kaufen");
15          output("2 − Übersicht");
16          letvar wahl := input();
17          if (wahl==1) {
18            output("Anzahl der Karten: ");
19            letvar anz := input();
20            call kartenkaufen(nr,anz);
21          }
22        }
23      }
24    }
```

Zu Beginn des Programms wird in Zeile 7 das Kinoprogramm abgerufen. Somit könnte es zum ersten Informationsfluß (nach Regel 6.7) kommen:

$P_{call}[1] = \emptyset$ (keine Parameter) $\cup\, P_{state} \cup P_{flow} \cup \{session_1\} = \{session_1\}$

Nach Regel 6.6 erhält P_p nun den Privacy-Wert des Aufrufs und von P_{state}:

$P_p = P_{call}[1] \cup P_{state} = \{session_1\}$

Die While-Schleife in Zeile 8 sorgt dafür, daß der Privacy-Wert der Schleifenbedingung auf den Stack PS_{state} gelegt wird – in diesem Fall die leere Menge.

$push(PS_{state}, \emptyset) \Rightarrow PS_{state} = [\emptyset]$

Durch die Benutzereingabe in Zeile 10 wird der Variable nr die Privacy-Annotation $P_{nr} = \{input_1\}$ zugewiesen. In der darauffolgenden Zeile wird die Altersfreigabe des gewählten Films angefragt:

$P_{call}[2] = P_{nr} \cup P_{state} \cup P_{flow} \cup \{session_2\} = \{input_1, session_2\}$
$P_{fsk} = P_{call}[2] \cup P_{state} = \{input_1, session_2\}$

Wir sehen, daß diesmal der Diensteanbieter Information über die Benutzereingabe gewinnt, da diese als Parameter beim Aufruf verwendet wurde. Die Überprüfung des Alters in Zeile 12 sorgt dafür, daß der Privacywert der bedingten Verzweigung auf den Zustandsstapel gelegt wird:

$push(PS_{state}, P_{fsk} \cup \{age\}) \Rightarrow PS_{state} = [\emptyset, \{input_1, session_2, age\}]$

Anschließend werden die Filmdetails abgefragt und ausgegeben. Hierbei wird der Informationsfluß wie folgt abgeschätzt:

$P_{call}[3] = P_{nr} \cup P_{state} \cup P_{flow} \cup \{session_3\} = \{input_1, age, session_2, session_3\}$

In Zeile 16 wird die Benutzereingabe für die nächste Aktion entgegengenommen. Ihr wird der folgende Privacywert zugeordnet:

$P_{wahl} = \{input_2\} \cup P_{state} = \{input_1, input_2, session_2, age\}$

Nehmen wir an, der Benutzer hätte sich für den Kauf einer Karte entschieden. Das bedeutet, daß der folgende bedingte Programmblock betreten wird:

$push(PS_{state}, P_{wahl}) \Rightarrow$
$PS_{state} = [\emptyset, \{input_1, session_2, age\}, \{input_1, input_2, session_2, age\}]$

Die Eingabe der Kartenanzahl in Zeile 19 wird mit folgendem Datenfluß abgeschätzt:

$P_{anz} = \{input_3\} \cup P_{state} = \{input_1, input_2, input_3, session_2, age\}$

Der folgende Diensteaufruf erhält somit diese Abschätzung:

$P_{call}[4] = P_{nr} \cup P_{anz} \cup P_{state} \cup P_{flow} \cup \{session_4\} =$
$\{input_1, input_2, input_3, age, session_2, session_4\}$

Er ist also abhängig vom Alter, den Benutzereingaben und von $session_2$, nämlich der FSK des gewählten Films.

6.4.7 Mögliche Erweiterungen

In diesem Kapitel möchten wir weitere Möglichkeiten vorschlagen, wie mit Hilfe einer solchen Skriptsprache die Privatsphäre weiter geschützt werden kann. Da diese den Formalismus zur Verfolgung von Datenflüssen nicht mehr verändern, wird hier auf eine Spezifikation der Syntax verzichtet.

Vergröbern von Informationen

Eine weitere Möglichkeit, das Zusammenführen von Daten zu verhindern, ist die gezielte Vergröberung oder Verfälschung von PAVs. Dieser Ansatz wird insbesondere bei der Nutzung von Location-based Services genutzt (z. B. [117]). Am Beispiel von Positionsdaten wurde aber gezeigt, daß reines Randomisieren durch Hinzufügen einer gleichverteilten Schwankung die erhoffte Anonymisierung unter Umständen nicht erreicht wird [97, 116]. Daher ist es sinnvoller, innerhalb einer Kette von zusammenführbaren Diensteaufrufen denselben (veränderten) Wert zu verwenden.

Ein weiterer Ansatz, die Informationen gezielt zu Vergröbern, ist das Einteilen eines Werts in bestimmte Intervalle; der Dienst erfährt lediglich, in welchem Intervall die zugehörige PAV liegt. Das Alter eines Benutzers ist ein typisches Beispiel für diesen Ansatz: Im obigen Beispiel könnten das die Intervalle der Altersfreigaben sein. In anderen Fällen interessiert möglicherweise lediglich, ob der Benutzer volljährig ist.

Die Verwendung solcher Intervallbezeichnungen machen die Abläufe für den Benutzer transparenter und die dahinterliegenden Absichten leichter erfaßbar. Da die

preiszugebende Informationsmenge sinkt, dürfte die Bereitschaft von Benutzern, den Dienst zu nutzen, ansteigen.

Verwendung authentisierter Werte

In bestimmten Situationen kann eine der Parteien auf der Verwendung authentisierter Werte bestehen. Dabei unterscheiden wir *verbindliche Aussagen* und *beglaubigte Aussagen*. Verbindliche Aussagen werden von der Person selbst getätigt; beglaubigte Aussagen hingegen sind Aussagen dritter, die von der Person nur weitergegeben werden. Ein Beispiel für eine verbindliche Aussage ist die Zusage zu einem Kaufgeschäft. Im obigen Beispiel könnte der Kinobetreiber an einer Stelle eine beglaubigte Aussage über das Alter des Kunden fordern, um sicherzustellen, daß die Bestimmungen zur Altersfreigabe eingehalten werden.

Eine verbindliche Aussage kann mit Hilfe digitaler Signaturen getätigt werden. Das für die Unterschrift benutzte Schlüsselpaar muß dem Empfänger bekannt oder durch eine (vom Empfänger als vertrauenswürdig betrachtete) Signaturkette bestätigt sein. Zusätzlich sollte eine verbindliche Aussage eine Frischeinformation beinhalten, um Replay-Angriffe auszuschließen.

Eine bestätigte Aussage kann durch zwei Schritte erreicht werden: Eine vertrauenswürdige Instanz erstellt einen signierten (verbindlichen) Datensatz, welcher die Aussage sowie die Identität der Person beinhaltet. Der Empfänger prüft zunächst die Gültigkeit der Signatur und sein Vertrauensverhältnis zum Aussteller. In einem zweiten Schritt prüft er die Identität seines Gegenübers.

Beide Fälle benötigen also die Identifierung des Senders: Die Aussagen sind an eine feste Identität gebunden. Es ist zwar möglich, daß ein Knoten mehrere Identitäten (Pseudonyme) besitzt, jedoch kann er diese nicht nach Belieben selbst erzeugen oder austauschen. Sie stellen für die Wiedererkennung also ein eindeutiges identifizierendes Kriterium dar.

In unserer einfachen Skriptsprache können verbindliche Aussagen durch Einführen einer Funktion *sign*() modelliert werden; beglaubigte Aussagen lassen sich nicht auf Bedarf generieren – sie können jedoch in PAVs (mit passender Privacy-Bewertung) für einige wichtige Werte vorgehalten werden.

6.4.8 Diskussion

In diesem Kapitel haben wir eine Möglichkeit gezeigt, den Informationsfluß innerhalb eines Programms zu verfolgen. Durch die Trennung in Programmzustand, Programmfluß und die Zusammenführbarkeit von einzelnen Diensteaufrufen wurde das Problem des Label Creep reduziert. Die Korrektheit der Aussage einer oberen

Schranke läßt sich per Induktion über die Programmzeilen nachvollziehen.[5]

Wie schon im vorigen Abschnitt angedeutet, läßt sich die hier skizzierte Skriptsprache leicht um Prozeduren und Funktionen erweitern. Die Sprache wurde hier bewußt einfach gehalten, um die wesentlichen Sachverhalte zu illustrieren; eine Übertragung auf gängige Skriptsprachen oder spezialisierte Skriptingsysteme für ubiquitäre Umgebungen (wie z. B. [141]) sollte kein Problem darstellen.

Wie bereits auf auf Seite 102 angesprochen, stellen Benutzereingaben ein Problem dar. Sie sind aus Privacy-Sicht „ungetypt“ und können dementsprechend nicht automatisch ausgewertet werden; lediglich ihr Vorhandensein im Informationsfluß kann durch die Pseudo-Labels $input_i$ verfolgt werden. Für eine Behandlung als PAV fehlen die hinterlegten Privacy-Informationen für die Auswertung – diese können nicht vom Diensteanbieter mitgeliefert werden (ansonsten wäre der Mechanismus beliebig umgehbar). Ein möglicher Ausweg wären zertifizierte Skripte: Eine vom Benutzer als vertrauenswürdig eingestufte Instanz würde das Skript überprüfen, die Eingabevariablen mit entsprechender Privacy-Annotation versehen und die gesamten Informationen signieren. Durch Prüfen der Signatur könnte der Benutzer sich versichern, ein korrektes „Privacy-Gütesiegel“ erhalten zu haben.

Die Annahme der Unabhängigkeit der PAVs (Definition 6.2 auf Seite 99) ist in sofern vereinfachend, da es neben expliziten Abhängigkeiten auch implizite bzw. sehr wahrscheinliche Abhängigkeiten geben kann. Diese sehr wahrscheinlichen Abhängigkeiten werden z.B. von Online-Warenhäusern für Kundenempfehlungen (im Stile von „Sie interessieren sich für Artikel X. Viele Kunden, die sich für X interessierten, hatten auch Interesse an Artikel Y.“) verwendet. Wie beispielsweise in der Arbeit von Frankowski et al. [90] gezeigt, lassen sich mit Hilfe solcher Charakteristika Daten zusammenführen. Andererseits gilt auch hier, analog zu den verfeinerten Verfahren zum Bewegungstracking ([22], siehe Kapitel 4.3 auf Seite 36), daß solche Angriffe bei konsequenter Nutzung von Privacy-Mechanismen ein Henne-Ei-Problem darstellen: Die für die Korrelationen nötigen Statistiken können nur erhoben werden, wenn die einzelnen Beobachtungen den richtigen Individuen zugeordnet werden können – und genau dies sollen die hier vorgestellten Maßnahmen verhindern.

6.5 Bewertung von Privacy-Aussagen

Das vorige Kapitel hat eine Möglichkeit zur Verfolgung des Datenflusses vorgestellt. Auf diese Weise kann festgestellt werden, von welchen PAVs ein Dienstaufruf abhängt. Um den Benutzer möglichst automatisch zu unterstützen, ist eine entsprechende Bewertung und eine darausfolgende Reaktion nötig. Diese ist stark

[5] Die in Kapitel 5.6.4 verwendete BAN-Logik läßt sich hier nicht einsetzen: BAN dient dazu, das Wissen über etwas zu beweisen. Die negative Aussage – daß jemand etwas nicht wissen kann – läßt sich hiermit nicht modellieren.

szenarioabhängig; dieses Kapitel soll einen kurzen Ausblick über mögliche Ansätze geben.

6.5.1 Regeln und Regelsysteme

Durch die Verfolgung des Informationsflusses kann bestimmt werden, über welche PAVs der Dienstenabieter direkt (aus PAVs selbst, Berechnungen mit ihnen oder bedingten Programmverzweigungen) oder indirekt (aus der Zusammenführbarkeit mehrerer Diensteaufrufe) Informationen gewinnen kann. Bei jedem Diensteaufruf muß eine Bewertungsfunktion entscheiden, ob der Aufruf durchgeführt werden soll. Dabei sind folgende Fragen zu beantworten:

- Ist es in Ordnung, daß jemand die Daten in dieser Kombination (also dem Wissen, daß sie zur selben Person gehören) erfährt und möglicherweise speichert?
- Wie groß ist die Wahrscheinlichkeit, daß man anhand dieser Daten eindeutig identifizierbar wird?

Identity-Management-Systeme [4, 104, 45] adressieren üblicherweise nur die erste Frage: Durch das Einführen von Rollen für bestimmte Aktivitäten wird nur noch ein Teil der persönlichen Daten unter einer einzelnen Identität preisgegeben. Die Abtrennung und die Kontrolle, welche Daten dies letztendlich sind, liegt nach wie vor beim Nutzer.

Das Problem, daß man auch bei Abwesenheit eines einzelnen identifizierenden Merkmals über eine Reihe von Kriterien identifiziert werden kann, wurde zwar insbesondere in Deutschland im Kontext der Volkszählung 1983 angeprangert [39], wird aber in aktuellen Systemen zum Identitätsmanagement nicht weiter betrachtet.

Für das Aufstellen eines Regel- und Präferenzensystems sind folgende Ansätze denkbar:

Zugriffsrechte für einzelne Dienste: Analog zu Access Control Lists (ACLs) werden jedem einzelnen Dienst der Zugriff auf bestimmte PAVs gewährt. Dies bedeutet zwar eine feine Granularität, jedoch einen hohen Konfigurationsaufwand. Hinzu kommt noch das Problem, daß die Rechtelage unklar ist, wenn ein Knoten auf einen neuen Dienst stößt, für den noch keine Konfiguration vorliegt.

Zuweisen von PAVs zu bestimmten Rollen: Auch hier werden (ähnlich wie beim vorigen Ansatz) Zugriffsrechte für einzelne PAVs bestimmt, jedoch gelten diese Rechte für ganze Klassen von Diensten; der Nutzer schlüpft je nach Dienst in eine andere Rolle. Schwierigkeiten treten auf, wenn sich die Rolle des Nutzers bei einem Dienst nicht klar bestimmen läßt (aufgrund des Fehlens einer passenden Rolle oder wegen Überlappungen von Rollen).

Allgemeine Regeln zur kontextabhängigen Einschätzung: Dieser Ansatz verfolgt nicht die Beziehung des Nutzers zum Dienst, sondern setzt die PAVs untereinander in gegenseitigen Bezug. Solche Regeln können allgemein vorschreiben, daß ein Dienst beispielsweise nur eine bestimmte, maximale Anzahl von PAVs präsentiert bekommt, bestimmte Gruppen von PAVs nicht zusammen übertragen werden dürfen, etc.

Offensichtlich ist der letzte Ansatz wohl der flexibelste; allerdings ist seine saubere Einrichtung auch am kompliziertesten. Hilfreich hierfür ist es, wenn die PAVs nach verschiedenen Kriterien in Gruppen zusammengefaßt werden und anhand bestimmter Eigenschaften durch die Regeln selektierbar sind. Die Forderung nach einer allgemeinen Klassifizierung und maschinenlesbaren Beschreibung von Privacy (und damit auch von Privacy-relevanten Daten) ist nicht neu – bereits Kim et al. [138] stellen die Notwendigkeit einer Privacy-Ontologie fest.

Diese Forderung nach einer allgemeinen Privacy-Ontologie blieb bis heute jedoch unerfüllt; für einzelne Szenarien hingegen stellen verschiedene Autoren (z. B. [130]) solche Kategorisierungen auf.

6.5.2 Privacy-Wert von Daten

Im Zuge solcher Bewertungsregeln wäre es von Vorteil, wenn man den Daten einen „Privacy-Wert“ zuweisen könnte (anstatt die Regeln nur mit logischen Verknüpfungen zu erstellen). Wir stellen in diesem Abschnitt eine Reihe möglicher Aspekte solcher Bewertungen vor.

Analog zu den beiden Fragestellungen für Regelsysteme kann sich ein solcher Wert aus zwei Aspekten zusammensetzen:

- Wie sensitiv schätzt der Benutzer die Aussage ein
- Wie hoch ist das Risiko, anhand eines Wertes wieder identifiziert zu werden

Die erste Frage („was geht das jemand anderes an“) ist vom Benutzer nur subjektiv zu beantworten. Aber auch die Risiko-Abschätzung ist stark vom Kontext der Information abhängig und damit schwer zu bestimmen [226]. Wir geben im folgenden einige Kriterien an, die auf das Identifizierungsrisiko Einfluß haben und sich zumindest teilweise automatisiert behandeln lassen (ein Teil hiervon wurde bereits in Abschnitt 6.4.5 angesprochen):

Kardinalität des Wertebereichs: Je größer die Anzahl der möglichen Ausprägungen eines Werts, desto besser eignet sich der Wert als Selektionskriterium in einer Datenbasis.

Werteverteilung: Gibt es Wertausprägungen, die deutlich seltener vorkommen als andere, so sind diese besonders geeignet, die Zahl möglicher Kandidaten in einer Datenbank zu reduzieren.

Validitätsdauer: Je länger der Wert einer PAV unverändert bleibt, desto länger kann er einem Angreifer als Selektionskriterium dienen. So bleiben beispielsweise Name und Vorname (fast) lebenslänglich konstant, Daten über Hobbies oder Arbeitsstelle ändern sich nur selten, wohingegen Kontextinformationen wie „Benutzer ist in Eile“ sich sehr häufig ändern.

Genauigkeit der Daten: Wurden die Informationen mit einer gewissen zufälligen Schwankung versehen, so sinkt ihr Privacy-Wert. Ebenso sinkt der Privacy-Wert, wenn nur auf bestimmte Intervalle und Wertebereiche geprüft wird.

Kontextabhängige Einflüsse: Manche der oben genannten Kriterien besitzen einen weiteren Kontextbezug. So ändert sich beispielsweise die Werteverteilung des Nachnamens „Häberle“ in Abhängigkeit des aktuellen Aufenthaltsortes – in Stuttgart tritt der Name vergleichsweise häufig auf, wohingegen er in Hamburg im Telefonbuch nur wenige Male zu finden ist. Analoge Beispiele findet man in größerem Maßstab bei ausländischen Namen („Kovacz“ ist in Ungarn sehr häufig, wohingegen man den Name in Deutschland vergleichsweise selten antrifft).

Die ersten vier Punkte lassen sich bei der Definition der PAVs noch vergleichsweise gut modellieren; ernsthafte Schwierigkeiten bereitet hingegen der kontextabhängige Einfluss. Hierfür ist beim Belegen der PAVs die Unterstützung des Benutzers vermutlich häufig erforderlich.

6.5.3 Unterstützung durch Trust-Beziehungen

Eine Möglichkeit, die Schwelle für die Bereitschaft zur Preisgabe personenbezogener Daten zu bestimmen, ist die Verwendung von Trust-Mechanismen. Dies können im einfachsten Fall „Gütesiegel“ in Form von Zertifikaten sein, mit welchen eine (vom Benutzer als vertrauenswürdig eingestufte) Institution bestätigt, daß der Dienstanbieter korrekt mit den Benutzerdaten umgeht.

Alternativ sind Systeme basierend auf Bewertungen oder ähnlichem denkbar: Laut Glaeser et al. [91] entspricht die Annahme, von Vertrauensaussagen in der Vergangenheit auf die Gegenwart schließen zu können, der menschlichen Psyche. Selbiges gilt auch für „Web of Trust“-Annahmen: Die Vertrauenswürdigkeit einer Person kann überprüft werden, indem man dessen Vertrauensbeziehungen betrachtet (einen Überblick über solche komplexeren, teils dezentral funktionierenden Trust-Systeme findet man in [34, 220]).

Privacy und Trust bilden eine Tradeoff-Beziehung [206], wie man bereits an den obigen Beispielen sehen kann: Ein Zertifikat kann nur ausgestellt und überprüft werden, wenn die Identität des Zertifizierten bekannt ist. Allgemein ist es schwer, jemandem zu vertrauen, den man nicht kennt.

Bei näherer Betrachtung stellt man fest, daß sich Ratingverfahren für die Bewertung des korrekten Umgangs mit personenbezogenen Daten nur schlecht eignen. Um die Validität einer Bewertung einschätzen zu können, müßte man Informationen über deren Aussteller besitzen – und dies ist als allererstes dessen Identität, was dem Wunsch nach Anonymität der Dienstenutzer widerspricht. Zum anderen kann jeder Nutzer nur das bewerten, was er auch beobachten kann; der Mißbrauch von Daten ist jedoch nur sehr schwer zu beobachten, da die Auswirkungen erst mit großer Verzögerung zu spüren und nur in seltenen Fällen einer genauen Quelle zuzuordnen sind.

Kritische Stimmen [148] stellen die Grundsatzfrage, ob Trustmechanismen in ubiquitären Szenarien überhaupt sinnvoll sind: Es sei schließlich sinnvoller, Transparenz zu schaffen und die Vertrauens-Thematik dem Mensch zu überlassen, als zu versuchen, diese Mechanismen zu imitieren. Einfache vertrauensbildende Maßnahmen (wie die oben erwähnten Zertifikate) können einen positiven Effekt auf ubiquitäre Umgebungen haben; in unserem Szenario sollen diese Maßnahmen die Benutzerentscheidung auch nicht ersetzen, sondern lediglich für eine Entlastung des Nutzers sorgen: Die Schwelle, ab welcher der Benutzer mit einer Rückfrage behelligt wird, läßt sich mit Hilfe einfacher vertrauensbildender Mechanismen etwas hinausschieben. Kommt es zur Rückfrage beim Benutzer, sorgt der Rest unseres Systems für die gewünschte Transparenz.

6.5.4 Integration der Schnittstellen der unteren Netzwerkschichten

Auf einen Zugriff aus der Skriptsprache heraus auf die Schnittstellen des Privacy-Frameworks unterhalb der Anwendungsebene (siehe Kapitel 5.7 auf Seite 85) haben wir bewußt verzichtet: Die Skripte sollen einzig dazu da sein, bestimmte Anwendungen aus angebotenen Diensteschnittstellen zu modellieren. Vielmehr sollten die Möglichkeiten, welche die Schnittstelle bietet, bei der Bewertung der Privacy-Aussagen genutzt werden: So sind hier stärkere Aussagen als ein pauschales „ja“ und „nein“ möglich.

Ab einer bestimmten Schwelle kann so beispielsweise gefordert werden, daß sich die Gegenstelle identifiziert – entweder durch Austausch eines Zertifikats oder durch einen entsprechend markierten Eintrag in der Liste wiedererkennbarer Geräte. Mit dieser Information sind dann weitere Entscheidungen möglich; beispielsweise kann gefordert sein, daß sich die Gegenstelle auf einer Liste vertrauenswürdiger Personen befindet, oder aber daß das Zertifikat die Signatur einer vertrauenswürdigen Prüfstelle (sozusagen ein „Privacy-TÜV“) trägt.

Umgekehrt kann auch die Gegenstelle die Identifizierung fordern: Die Entscheidung, ob dieser stattgegeben wird, ist letztendlich ebenfalls eine Privacy-Entscheidung – mit dem Unterschied, daß diese nicht aufgrund des Programmverlaufs getroffen wird. Bedingung für die Herausgabe der eigenen Identität kann wiederum die Iden-

tifizierung des Gegenübers sein, welche dann wie im vorigen Abschnitt zur Entscheidungsfindung herangezogen werden kann.

6.6 Fazit

Ohne geeignete Mechanismen für Security und Privacy dürften es ubiquitäre Umgebungen schwer haben, sich aus eigener Kraft zu etablieren. Um jedoch eine Durchdringung des Alltags zu erlangen, wie es die Vision des Ubiquitous Computings vorsieht, sind gute Usability-Eigenschaften für diese Mechanismen erforderlich [13].

In diesem Kapitel haben wir ein Verfahren vorgestellt, mit dem es möglich ist, daß ein Benutzer die vollständige Kontrolle über seine Daten behält: Mit Hilfe der Annotation privacy-relevanter Daten und der Verfolgung des Datenflusses ist es möglich, die Weitergabe personenbezogener Daten zu kontrollieren und auf Wunsch zu verhindern.

Die Kontrolle ist durch einen applikationsspezifischen Monitor zu implementieren. Je nach Mächtigkeit der Hardware reicht die Bandbreite hier von einem einfachen regelbasierten Monitor bis hin zu einer Implementierung, die die Regelkonformität durch automatisches Schließen aus einer Regelbasis bestimmt. Dementsprechend muß die zugrundeliegende Beschreibungssprache für die Policies gestaltet sein. Die momentan bekannten Beschreibungssprachen für Privacy-Präferenzen greifen für einen solchen Einsatz zu kurz; vorgestellt wurde ein Ansatz, mit dem eine Bewertung der Privacy unabhängig vom Kommunikationspartner oder einer selbstdefinierten Rolle erfolgen kann.

Weiterhin wurden einige Erweiterungen der vorgestellten Skriptsprache aufgezeigt; solche Erweiterungen sind sinnvoll, wenn die Modellierung bestimmter Sachverhalte mit Hilfe der Skriptsprache zu restriktiv wäre oder schlicht nicht möglich ist. So können beispielsweise spezielle Befehle eingeführt werden, die besondere Anforderungen an die tieferliegenden Schichten (wie Vorhandensein eines authentischen Schlüssels oder eine überprüfte Identität der Gegenstelle) stellen. Eine solche Erweiterung könnten zum Beispiel spezielle Befehle für anonyme Bezahlmechanismen (z. B. [207]) sein.

7 Fazit

> Das Problem kennen ist wichtiger, als die Lösung zu finden, denn die genaue Darstellung des Problems führt automatisch zur richtigen Lösung.
>
> *Albert Einstein (1879–1955)*

7.1 Zusammenfassung

Wir haben in dieser Arbeit das komplexe Themengebiet des Schutzes der Privatsphäre im Ubiquitous Computing umfassend beleuchtet. Nach einer Motivation und Begründung, weshalb eine solche Kontrolle auch bei jedem einzelnen Gerät vonnöten ist, wurden entlang des klassischen Netzwerk-Protokollstapels einzelne Problemfelder identifiziert und Lösungen hierfür vorgeschlagen.

Das hieraus entstandene Gerüst bietet jedem mobilen Knoten zunächst ein größtmögliches Maß an Anonymität, welche das Gerät dann stückweise und kontrolliert aufgeben kann. Hierbei setzt das Gesamtgerüst keine globale, ständig verfügbare Infrastruktur voraus.

Die wesentlichen Beiträge, die diese Arbeit auf diesem Forschungsgebiet leistet, sind im folgenden nochmals zusammengefaßt.

- Nach unserem Kenntnisstand ist dies die erste Arbeit im Bereich des Ubiquitous Computing, die das Thema Privacy in diesem Umfang und durch alle Schichten hindurch beleuchtet.
- Die Arbeit liefert eine schlüssige Argumentationskette (basierend auf historischen und philosophischen Aspekten), weshalb Privacy im Ubiquitous Computing, insbesondere auch technische Schutzmaßnahmen, essentiell notwendig sind.
- Der Anonymitätsgewinn bei mobilen Knoten durch wechselnde Pseudonyme wurde ausführlich untersucht. Hierfür kam in Form des Multi-Hypothesen-Trackings ein moderner Trackingalgorithmus zum Einsatz, welcher in nur sehr wenigen vorhergehenden Veröffentlichungen zum Einsatz kam. Nach unserem Wissen kamen bei bisherigen Publikationen nur einfache Bewegungsmodelle zur Erzeugung synthetischer Bewegungsdaten zum Einsatz; die Verwendung von komplexen Modellen wie dem hier verwendeten Socialforce-Modell ist neu.
- Die widersprüchlichen Ziele von Anonymität und sicherem Schlüsselaustausch wurden detailliert diskutiert und ein Lösungsansatz vorgeschlagen.

- Das Problem der Wiedererkennung anonymer Knoten ist nach unserem Wissen eine komplett neue und bis dato unbehandelte Problemstellung. Hierfür stellten wir ein effizientes Protokoll vor, welches das Problem sowohl bei paarweiser als auch bei Multicast-Kommunikation löst. Um die Korrektheit des Verfahrens zu untermauern, wurde das Protokoll mit Hilfe der BAN-Logik formal untersucht.
- Am Beispiel einer abstrakten Servicebeschreibungs- und Skriptsprache wurde aufgezeigt, wie der Fluß von sensiblen Informationen verfolgt werden kann. Mit Hilfe des hier vorgestellten Verfahrens ist es möglich, für jeden entfernten Diensteaufruf eine obere Schranke für die Menge der übertragenen Daten (sowohl durch direkten Informationsfluß als auch durch indirekte Abhängigkeiten oder implizite Informationen durch Verzweigungen im Programm) zu bestimmen.

7.2 Ausblick, Übertragbarkeit der Ergebnisse

Das hier vorgestellte Gerüst bietet ein robustes Fundament für ubiquitäre Systeme, welche den Schutz der Privatsphäre respektieren. Das Forschungsfeld selbst ist jedoch breiter – je nach Anwendung, Szenario und zur Verfügung stehender Rechenleistung können sich entsprechende weitere Forschungsarbeiten anschließen.

So bietet die Skriptsprache der Anwendungsebene nun eine Möglichkeit, den Informationsfluß zu verfolgen; jedoch möchte man wohl kaum jede Kombination von Informationen einzeln mit einer Regel versehen. Ebenso greift ein einfacher Punkteansatz („PAV x ist y Punkte wert; ab einer Summe von z Punkten wird die Übertragung unterbrochen“) sicherlich zu kurz. Im Themenkomplex des automatischen Schließens und der künstlichen Intelligenz ließe sich hier sicherlich ein intelligenteres, flexibles Verfahren finden, welches einen Großteil der Benutzerentscheidungen übernehmen kann. Die Rechenleistung der Kleingeräte des Ubiquitous Computings dürften hierfür eine weitere, besondere Herausforderung darstellen.

Die allgemeine Frage nach einer Ontologie für Privacy, personenbezogene Daten und den hier verwendeten PAVs, wurde zwar schon vor längerer Zeit gestellt [138], aber bis heute nicht generisch bearbeitet. Eine solche Ontologie wäre für die Beschreibung der Regeln für die Weitergabe der PAVs sicherlich eine große Hilfestellung.

Darüber hinaus gibt es weitere Themengebiete, die in der Forschungslandschaft des Ubiquitous Computing zwar angesprochen wurden, aber bis dato den Privacy-Aspekt noch nicht oder nur bedingt berücksichtigen. Im Kontext der Bezahlprotokolle gibt es zwar kryptographische Verfahren, welche anonyme, kryptographische Münzen realisieren, jedoch sind diese vom Rechenaufwand für Kleingeräte kaum einsetzbar. Eine ebenfalls interessante Fragestellung ist die der semantischen Adressierung, also dem Kommunikationsaufbau mit Geräten in der Umgebung anhand

bestimmter Eigenschaften (jenseits der Netzwerkadresse, beispielsweise „der Radler vor mir“, „die blonde Dame zwei Sitze weiter“, „die rote Anzeigetafel“, etc.) – eine sicherlich sehr interessante Technik, die jedoch wiederum die Privatsphäre tangiert.

Der Schutz der Privatsphäre ist jedoch kein Problem, welches dem Ubiquitous Computing allein anheim ist. Es tritt vielmehr an den verschiedensten Stellen in unserem Alltag und bei der Benutzung elektronischer Kommunikation auf.

Dank der Untergliederung entlang des Schichtenmodells sind jedoch einzelne Komponenten dieser Arbeit auf andere Bereiche übertragbar. So wäre es beispielsweise denkbar, das Konzept der Informationsverfolgung aus Kapitel 6 auf WSDL zu übertragen und so den Fluß von personenbezogenen Daten zwischen Web Services zu verfolgen.

7.3 Schlußwort

Das Themengebiet der Privatsphäre ist ein sehr interessantes und forschungstechnisch anspruchsvolles Thema – nicht zuletzt deswegen, weil es sich der Ideen, Techniken und Algorithmen vieler verschiedener Bereiche der klassischen Informatik bedient und mitunter sogar Themen aus anderen Disziplinen wie beispielsweise der Psychologie streift. Aber auch gegenwärtigen Entwicklungen in Politik und Gesellschaft geben diesem Thema Aktualität und Praxisbezug, was ebenfalls zur Faszination dieses Forschungsgebietes beiträgt.

Die Vision des Ubiquitous Computing ist gerade in der Geburtsphase; Wirtschaft und Politik sind gerade dabei, ubiquitäre Techniken zu entdecken und beginnen damit, diese auf breiter Front einzusetzen. Ob sich die Modellprojekte von intelligenten Häusern, dem mitdenkenden Kühlschrank oder anderen, den Alltag durchdringenden Ideen so weit entwickeln, daß sie von der Mehrheit der Bevölkerung akzeptiert, gekauft und verwendet werden, wird die Zukunft zeigen müssen. Mit dem wachsenden öffentlichen Interesse für die Privatsphäre und der Angst vor einem Big-Brother-Staat könnte Privacy hierbei ein entscheidender Faktor werden.

Die ausgiebige Auseinandersetzung mit dem Thema Privacy hat uns jedoch eines sehr deutlich gemacht: Eine Welt, vollkommen durchdrungen von einem Ubiquitous Computing ohne Schutzmaßnahmen für die Privatsphäre, könnte George Orwells Visionen wie harmlose Kinderphantasien aussehen lassen.

A Anhang

A.1 Kryptographische Grundlagen

Einige in der Arbeit verwendeten Begriffe sollen hier kurz erläutert werden. Detaillierte Grundlagen und Informationen zu einzelnen Algorithmen finden sich in [201]; Ferguson und Schneier [88] beschreiben detailliert Hinweise zur sicheren Implementierung kryptographischer Protokolle. Stinson [216] bietet eine mathematische Einführung in die Kryptographie. Im Buch von Stajano [213] werden die speziellen Probleme im Umfeld des Ubiquitous Computings ausführlich vorgestellt und für einige Problemfelder mögliche Lösungen aufgezeigt.

A.1.1 Symmetrische Chiffren

Verschlüsselung allgemein dient dazu, Informationen so zu codieren, daß diese Information ohne Kenntnis des verwendeten Schlüssels (de facto) nicht wiederhergestellt werden kann. Eine ideale Chiffre hat die Eigenschaft, daß das Chiffrat von zufälligen Bits (weißem Rauschen) nicht zu unterscheiden ist.

Bei symmetrischen Chiffren ist der Schlüssel zum Ver- und Entschlüsseln identisch, bzw. es ist möglich, den einen Schlüssel auf einfache Weise vom anderen Schlüssel abzuleiten. Populäre Vertreter dieser Klasse sind AES [68] oder Twofish [199, 198].

A.1.2 Asymmetrische Chiffren

Die Anforderungen an eine asymmetrische Chiffre sind identisch zu denen, die wir bei den symmetrischen Chiffren beschrieben haben. Der Unterschied zu den symmetrischen Chiffren besteht darin, daß es statt einem Schlüssel ein Schlüsselpaar gibt: Eine Nachricht, die mit einem der beiden Schlüssel chiffriert wurde, kann nur mit dem anderen entschlüsselt werden. Ein übliches Vorgehen ist, einen der beiden Schlüssel öffentlich zu verteilen, während man den zweiten Schlüssel geheim hält; von diesem Vorgehen rührt der Name *Public-Key-Kryptographie* her.

Jeder kann mit Hilfe des öffentlichen Schlüssels eine Nachricht versenden, die nur der Empfänger lesen kann. Umgekehrt kann der Besitzer des geheimen Schlüssels Nachrichten signieren, indem er sie mit seinem geheimen Schlüssel chiffriert: Nur mit dem korrespondierenden Public Key erhält man ein sinnvolles Ergebnis, weshalb

die Empfänger sicher sein können, daß die Nachricht vom Inhaber des geheimen Schlüssels sein muß.

Asymmetrische Chiffren benötigen einen deutlich höhreren Rechenaufwand als symmetrische Chiffren (meist in der Dimension mehrerer Zehnerpotenzen). Populärster Vertreter dieser Klasse von Chiffren ist wohl das RSA-Verfahren [185].

A.1.3 Modi für Chiffren

Prinzipiell unterscheidet man zwei Arten von Chiffren: *Stromchiffren* und *Blockchiffren*. Bei Stromchiffren wird die Nachricht Bit für Bit vom Algorithmus verarbeitet; bei Blockchiffren werden immer Datenblöcke einer bestimmten Länge auf einmal verarbeitet (was bedeutet, daß man den Klartext bis zur Blocklänge auffüllen muß).

Verwendet man eine Chiffre ohne weitere Vorkehrungen, so erzeugt derselbe Klartext auch immer denselben Chiffretext. So kann ein Angreifer erkennen, daß zweimal dieselbe Nachricht verschickt wurde, selbst wenn er nicht in der Lage ist, sie zu entschlüsseln. Deshalb verwendet man einen *Initialisierungsvektor*, der aus zufälligen Bits besteht und zusammen mit der Nachricht übertragen wird. Er fließt in die Verschlüsselung mit ein (z. B. durch Voranstellen vor die eigentliche Nachricht) und sorgt somit dafür, daß zwei identische Nachrichten im chiffrierten Zustand unterschiedlich aussehen (vorausgesetzt, daß verschiedene Initialisierungsvektoren verwendet wurden).

Werden Blockchiffren naiv Block für Block benutzt, so nennt man diese Verwendung *Electronic Code Book Mode*. Vorteil ist dabei, daß Übertragungsfehler nur die Entschlüsselung eines einzelnen Blocks verhindern; der Nachteil dieses Modus ist jedoch, daß jeder Block einzeln angegriffen werden kann, da keinerlei Abhängigkeit zu den restlichen Daten besteht.

Zu den häufig verwendeten Modi gehört beispielsweise das *Cypher Block Chaining* (schematisch Dargestellt in Abbildung A.1).

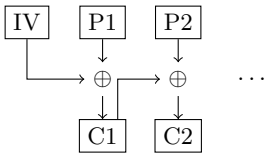

Abbildung A.1: Cipher Block Chaining

Für unsere ideale Chiffre gehen wir davon aus, daß die Chiffren so verwendet werden, daß kein Teilblock wiedererkannt werden kann, selbst wenn sich die Nachricht nur an einem einzelnen Bit an beliebiger Position unterscheidet.

A.1.4 Kryptographische Hashfunktionen

Hashfunktionen bilden ein Datenwort mit beliebiger Länge auf ein Datenwort mit fester (und damit potentiell kürzerer) Länge ab. Im Kontext der Datenbanken werden Hashfunktionen genutzt, um so einen Indexwert zu erhalten, mit dem schnell auf einen Datensatz zugegriffen werden kann.

Im Kontext der Kryptographie dienen Hashfunktionen dazu, einen „Fingerabdruck“ einer Nachricht zu erstellen. Dementsprechend haben kryptographische Hashfunktionen die Eigenschaft, daß ihre Umkehrbarkeit schwer sein soll: Es soll einen technisch nicht realisierbaren Aufwand erfordern, zu einem Hashwert eine zugehörige Nachricht zu konstruieren. Aus diesem Grund findet man Hashfunktionen mitunter auch unter dem Begriff *Falltürfunktion*.

Da der Wertebereich des Hashes kleiner als der Wertebereich der Nachrichten ist, kann es zu Kollisionen kommen. Solche Kollisionen sind aus Sicherheitssicht natürlich problematisch; hinzu kommt noch, daß es einem Angreifer unter Umständen genügt, zwei Nachrichten beliebigen Inhalts zu erzeugen, die aber beide auf denselben Hashwert abgebildet werden. Letzteres ist aufgrund des *Geburtstags-Paradoxons* einfacher als das Finden einer Kollision zu einer fixen Nachricht (Erläuterung siehe Kapitel 4.2.1 auf Seite 27). Dementsprechend definiert man die Anforderungen an eine kryptographische Hashfunktion:

Weiche Kollisionsresistenz: Das Problem, zu gegebenem Klartext p einen weiteren Klartext p' finden, so daß $H(p) = H(p')$, muß einen nicht realisierbaren technischen Aufwand erfordern.

Harte Kollisionsresistenz: Das Problem, ein beliebiges Paar p, p' mit $H(p) = H(p')$ zu finden, muß einen nicht realisierbaren technischen Aufwand erfordern. Da diese Aufgabe aufgrund des Geburtstagsparadoxons die leichter zu bewältigende Aufgabe ist (die Wahrscheinlichkeit für einen zufälligen Treffer ist höher), ist dies das härtere Kriterium.

A.1.5 Hash Chains

Unter einer Hash Chain versteht man das mehrfache Anwenden einer Hashfunktion auf einen Startwert: $C_0 = seed$, $C_{n+1} = H(C_n)$. Solche Hashketten können beispielsweise benutzt werden, wenn eine Person rückwirkend die Zusammengehörigkeit von Werten beweisen will: Dazu berechnet er zunächst k Werte der Hashchain und gibt nun der Reihe nach $C_k, C_{k-1}, \ldots$ preis. Durch Anwenden der Hashfunktion kann jeder Beobachter von C_{i-1} auf C_i schließen, jedoch ist es ihm nicht möglich, den nächsten Wert der Hash Chain vorherzusagen: Dazu müßte er die Hashfunktion umkehren, um an den ursprünglichen Startwert zu gelangen – und genau dies soll eine kryptographisch starke Hashfunktion verhindern.

A.1.6 Kryptographisch starke Zufallszahlengeneratoren

An verschiedenen Stellen (wie beispielsweise bei der Wahl von Initialisierungsvektoren oder dem Erzeugen von Sitzungsschlüsseln) werden Zufallszahlen benötigt. Gerade am Beispiel von Sitzungsschlüsseln wird klar, daß die Sicherheit des Verfahrens nicht nur vom verwendeten Verschlüsselungsalgorithmus, sondern auch von der Qualität der Zufallszahlen abhängig ist. Echte Zufallsbits sind in (per se deterministischen) Computern nur schwer zu bekommen.

Daher bedient man sich Pseudo-Zufallszahlengeneratoren (Pseudorandom number generators, PRNGs), welche aus einem Startwert (seed) eine längere Bitfolge erzeugen. Bei einem idealen PRNG besitzen die erzeugten Bits folgende Eigenschaften: Die Bits müssen gleichverteilt sein. Außerdem darf das Wissen über die Vergangenheit (Bits $b_0, \ldots b_n$) keine Rückschlüsse auf folgende Bits (Bit b_{n+1}) erlauben:

$$P(b_{n+1} \mid b_0, \ldots, b_n) = P(b_{n+1}) = \frac{1}{2}$$

Aus solchen Algorithmen läßt sich direkt eine Stromchiffre erzeugen (der Schlüssel ist der Startwert des PRNGs); umgekehrt läßt sich aus einer Chiffre ein kryptographisch starker Zufallszahlengenerator erzeugen.

A.1.7 Nonces

Das Wort *Nonce* ist im Ursprung ein Kunstwort, zusammengesetzt aus dem Englischen „n once“. Jeder Wert von n soll also nur einmal verwendet werden. Im einfachsten Fall erfüllen Sequenzzähler oder Zeitstempel diese Eigenschaft.

Im Kontext kryptographischer Protokolle wird meist zusätzlich gefordert, daß die Folge der Nonces nicht vorhersagbar ist – ähnlich der Forderung an PRNGs (siehe oben). Dies kann entweder durch Verwendung von (Pseudo-)Zufallszahlen erreicht werden, wobei der Wertebereich der Nonces groß genug sein muß, so daß die Kollisionswahrscheinlichkeit hinreichend klein wird.

Alternativ können nicht vorhersagbare Nonces auch durch einen Sequenzzähler und einen zufällig gewählten, geheimen Schlüssel erzeugt werden: Wegen der Bijektivitätseigenschaft einer Chiffre gibt es zu jedem Sequenzwert ein unterschiedliches Chiffrat und somit keine Kollisionen. Das Vorhersagen der so entstehenden Folge entspricht einem Angriff auf die verwendete Chiffre (bzw. dem verwendeten Schlüssel). Um einen Angriff zusätzlich zu erschweren, sollte der Sequenzzähler mit einer Zufallszahl initialisiert werden, um das Erraten des Sequenzwerts zu erschweren und so eine Known Plaintext Attack[1] zu verhindern.

[1] *Known Plaintext Attack*: Eine Angriffssituation, in welcher der Angreifer zu einem abgehörten Chiffretext auch den zugehörigen Klartext besitzt und darauf aufbauend versucht, den verwendeten Schlüssel zu rekonstruieren.

A.2 BAN-Logik

Die Arbeiten von Burrows, Abadi, und Needham [40] stellt die BAN-Logik vor (der Name rührt von den Anfangsbuchstaben der Autoren her; die häufig verwendete Deutung als Akronym für „*Belief And kNowledge*“ ist allenfalls als Bonmot zu betrachten). Diese ermöglicht eine formale Analyse von Authentisierungsprotokollen und soll dabei helfen, überflüssige Operationen oder unnötige Protokollschritte zu entdecken, potentielle Schwachstellen aufzudecken sowie die Annahmen, Vorbedingungen und Schlußfolgerungen zu konkretisieren. Sie stellt somit keinen formalen Beweis der Unangreifbarkeit dar.

Die BAN-Logik wurde 1989/1990 publiziert [40, 41]. Eine Einführung in den Formalismus samt einer Beispielsanalyse findet man bei [237]. Die Vorstellung der BAN-Logik blieb nicht ohne Kritik (z. B. [164]): Hauptkritikpunkt ist der Idealisierungsschritt der Protokolle, Vorbedingungen und Ziele, da dieser lediglich informell definiert ist. Ebenfalls fehlt der BAN-Logik ein definiertes Angreifermodell, weshalb einzelne Angriffe auf ein Protokoll individuell zu modellieren sind. Dennoch eignet sich die BAN-Logik, um Protokolle einer ersten abstrakten Prüfung zu unterziehen, was in verschiedenen publizierten Verfahren Lücken aufgezeigt hat [74, 41, 209]. Die Kritik an der BAN-Logik förderte die Entwicklung weiterer formaler Ansätze zur Überprüfung kryptographischer Protokolle; einen Vergleich verschiedener Verfahren zur Protokollverifikation findet man in [134].

A.2.1 Notation

Die BAN-Logik kennt Protokollteilnehmer, Schlüsselmaterial sowie Aussagen (Formeln). In der folgenden Übersicht bezeichnen wir analog zur Originalpublikation [40] Teilnehmer mit P, Q und R, kryptographische Schlüssel mit K und sonstige Aussagen mit X und Y.

$P \models X$: P glaubt X
: P geht davon aus, daß X wahr ist.

$P \lhd X$: P sieht X
: P hat die Nachricht X erhalten bzw. (beispielsweise nach einer Entschlüsselung) kann X lesen.

$P \mid\sim X$: P sagte X
: P sendete zu einem früheren Zeitpunkt die Aussage X; dieser Zeitpunkt kann früher im aktuellen Protokollauf gewesen sein oder von einem anderen Protokollauf stammen. Zum Sendezeitpunkt von X galt: $P \models X$.

$P \Rightarrow X$: P hat Autorität über X
: P ist vertrauenswürdig in Bezug auf X; P kann z. B. ein Server mit spezieller Funktionalität bezüglich X (wie beispielsweise Erzeuger einer Signatur) sein.

$\sharp(X)$: X ist frisch
X wurde noch in keiner früheren Nachricht (also auch in keinem früheren Protokollauf) verwendet; die Aussage $\sharp(Nonce)$ ist also immer wahr.

$P \overset{K}{\leftrightarrow} Q$: P und Q besitzen einen gemeinsamen (geheimen) Schlüssel K
Niemand sonst kennt K oder kann K erlangen, außer P oder Q vertrauen ihm.

$\overset{K}{\mapsto} P$: K ist öffentliche Schlüssel von P
Den entsprechenden geheimen Schlüssel K^{-1} kennt nur P (sowie gegebenenfalls dessen Vertraute).

$P \overset{X}{\rightleftharpoons} Q$: P und Q besitzen ein gemeinsames Geheimnis X
Eventuell kennen noch weitere vertrauenswürdige Instanzen X, doch nur P und Q dürfen X zur gegenseitigen Identifikation verwenden.

$\{X\}_K$: X ist mit K verschlüsselt
Abhängig vom Typ von K wird symmetrisch oder asymmetrisch verschlüsselt. Es wird angenommen, daß jeder Teilnehmer seine eigenen Nachrichten erkennt und diese ignoriert (sie reagieren also nicht auf ihre eigenen Nachrichten mit dem nächsten Protokollschritt).

$\langle X \rangle_Y$: X kombiniert mit der Formel Y
Y ist üblicherweise ein Geheimnis, so daß X zusammen mit Y die Identität von X beweist. Y kann beispielsweise ein Paßwort sein, mit welchem durch eine schlüsselgebundene Hashfunktion eine Signatur von X erzeugt wird.

A.2.2 Ableitungsregeln

Mit den oben vorgestellten Aussagen werden die Protokollnachrichten formuliert; mit Hilfe der Schlußregeln können aus diesen neue, gültige Aussagen abgeleitet werden. Die verwendete Notation der Form $\frac{X}{Y}$ besagt, daß Y gefolgert werden darf, sofern X erfüllt ist. Eine logische und-Verknüpfung mehrerer Aussagen wird mit einem Komma notiert.

Die sogenannten *Message-Meaning-Regeln* erlauben Rückschlüsse auf den Absender einer Nachricht. Die folgenden beiden Regeln beschreiben (einmal für symmetrische und einmal für asymmetrische Verfahren), daß P glaubt, daß die Nachricht X von Q stammt, falls P und Q einen gemeinsamen Schlüssel besitzen und P eine mit diesem Schlüssel chiffrierte Nachricht empfängt.

$$\frac{P \models Q \overset{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \models (Q \mid\sim X)} \tag{A.1}$$

$$\frac{P \models \overset{K}{\mapsto} Q, \quad P \lhd \{X\}_{K^{-1}}}{P \models (Q \mid\sim X)} \tag{A.2}$$

Entsprechendes gilt für den Besitz eines gemeinsamen Geheimnisses:

$$\frac{P \models Q \overset{Y}{\rightleftharpoons} P, \quad P \lhd \langle X \rangle_Y}{P \models (Q \mid\sim X)} \tag{A.3}$$

Die Nonce-Verification-Regel stellt die Überprüfung der Frische einer Nachricht dar: Falls die Nachricht frisch ist, glaubt der Absender noch an deren Gültigkeit:

$$\frac{P \models \sharp(X), \quad P \models (Q \mid\sim X)}{P \models (Q \models X)} \tag{A.4}$$

Die Jurisdiction-Regel setzt die Semantik von $\models\!\!\Rightarrow$ um: Wenn P glaubt, daß Q Kontrolle über X hat, so vertraut P auch Q bezüglich des Wahrheitsgehalts von X:

$$\frac{P \models (Q \models\!\!\Rightarrow X), \quad P \models (Q \models X)}{P \models X} \tag{A.5}$$

Eine nötige Eigenschaft von $\models$ ist, daß das Vertrauen in eine zusammengesetzte Nachricht das Vertrauen in jeden einzelnen Teil der Nachricht impliziert:

$$\frac{P \models X, \quad P \models Y}{P \models (X,Y)} \qquad \frac{P \models (X,Y)}{P \models X} \qquad \frac{P \models (Q \models (X,Y))}{P \models (Q \models X)} \tag{A.6}$$

Analoges gilt auch für $\models$ und $\lhd$:

$$\frac{P \models (Q \mid\sim (X,Y))}{P \models (Q \mid\sim X)} \qquad \frac{P \lhd (X,Y)}{P \lhd X} \tag{A.7}$$

Ähnlich hierzu gilt für die Zusammensetzung mit Nonces, daß eine zusammengesetzte Nachricht frisch ist, sofern ein Teil von ihr frisch ist:

$$\frac{P \models \sharp(X)}{P \models \sharp(X,Y)} \tag{A.8}$$

Die folgenden Regeln beschreiben die Entschlüsselung von Nachrichten: Ist P im Besitz des passenden Schlüssels (bzw. Geheimnisses), so kann P auch die entschlüsselte Nachricht sehen.

$$\frac{P \lhd \langle X \rangle_Y}{P \lhd X} \qquad \frac{P \models (Q \overset{K}{\leftrightarrow} P), \quad P \lhd \{X\}_K}{P \lhd X} \tag{A.9}$$

Analoges gilt natürlich für asymmetrische Verfahren:

$$\frac{P \models (\overset{K}{\mapsto} P), \quad P \lhd \{X\}_K}{P \lhd X} \qquad \frac{P \models (\overset{K}{\mapsto} Q), \quad P \lhd \{X\}_{K^{-1}}}{P \lhd X} \tag{A.10}$$

Die letzten Regeln beschreiben die Kommutativität von Schlüsseln – ein Schlüssel funktioniert in beide Richtungen:

$$\frac{P \models (R \overset{K}{\leftrightarrow} R')}{P \models (R' \overset{K}{\leftrightarrow} R)} \qquad \frac{P \models (Q \models (R \overset{K}{\leftrightarrow} R'))}{P \models (Q \models R' \overset{K}{\leftrightarrow} R))} \tag{A.11}$$

$$\frac{P \models (R \overset{X}{\rightleftharpoons} R')}{P \models (R' \overset{X}{\rightleftharpoons} R)} \qquad \frac{P \models (Q \models (R \overset{X}{\rightleftharpoons} R'))}{P \models (Q \models (R' \overset{X}{\rightleftharpoons} R))} \tag{A.12}$$

A.3 Definition Verband (lattice)

Ein algebraischer Verband $(V, \cap, \cup)$ über eine nichtleere Menge V besitzt zwei Operationen $\cap$ (meet, Infimum) und $\cup$ (join, Supremum). Beide Operationen sind kommutativ ($u \cap v = v \cap u$; analog für $\cup$) und assoziativ ($u \cap (v \cap w) = (u \cap v) \cap w$; analog für $\cup$). Zusätzlich gelten die Absorptionsregeln $u \cap (u \cup v) = u$ und $u \cup (u \cap v) = u$. Die Bedingung $u \cap v = u \;\Rightarrow\; u \leq v$ definiert eine partielle Ordnung auf V.

Für eine Menge M bildet die Potenzmenge $\mathcal{P}(M)$ mit dem Schnittoperator $\cap$ und der Vereinigung $\cup$ den sogenannten *Teilmengenverband*. Die Halbordnung wird durch die Mengeninklusion erzeugt: $A \subseteq B \;\Rightarrow\; A \leq B$

Ein Teilmengenverband über eine Menge mit drei Elementen sieht beispielsweise wie in Abbildung A.2 aus.

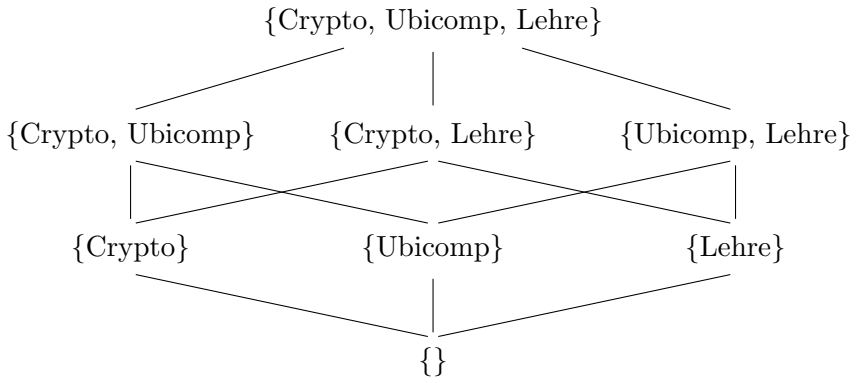

Abbildung A.2: Beispiel für einen Teilmengenverband

B Literaturverzeichnis

[1] Discreet service provision in smart environments. `http://www.ist-discreet.org/`.

[2] JIF: Java + Information Flow. `http://www.cs.cornell.edu/jif/`.

[3] NS-2: The network simulator. `http://nsnam.isi.edu/nsnam/`.

[4] PRIME - privacy and identity management for Europe. `https://www.prime-project.eu/`.

[5] Universal declaration of human rights. `http://www.un.org/Overview/rights.html`, 1948. UN-Resolution 217 A (III) vom 10.12.1948.

[6] Gesetz über die statistik für bundeszwecke (bstatg), Januar 1987.

[7] Jalal Al-Muhtadi, Roy Campbell, Apu Kapadia, M. Dennis Mickunas, und Seung Yi. Routing through the mist: Privacy preserving communication in ubiquitous computing environments. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, Seite 74. IEEE Computer Society, 2002. ISBN 0-7695-1585-1.

[8] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, 2001.

[9] S. Armenia und G. Morabito. *State of the art*. Technical Report D2103, DISCREET project, 2006.

[10] Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, und Matthias Schunter. Enterprise privacy authorization language (EPAL 1.2). W3C Member Submission, November 2003.

[11] Giuseppe Ateniese und Gene Tsudik. Some open issues and new directions in group signatures. In *FC '99: Proceedings of the Third International Conference on Financial Cryptography*, Seite 196–211, London, UK, 1999. Springer-Verlag. ISBN 3-540-66362-2.

[12] Giusseppe Ateniese, Amir Herzberg, Hugo Krawczyk, und Gene Tsudik. *Untraceable mobility or how to travel incognito. Comput. Networks*, 31(9):871–884, 1999. ISSN 1389-1286.

[13] Dirk Balfanz, Glenn Durfee, Rebecca E. Grinter, und D. K. Smetters. *In Search of Usable Security: Five Lessons from the Field. IEEE Journal on Security and Privacy*, 2(5):19–24, 2004.

[14] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana Smetters, Jessica Staddon, und Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Seite 180, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1940-7.

[15] Dirk Balfanz, Diana K. Smetters, Paul Stewart, und H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS*. The Internet Society, 2002. ISBN 1-891562-14-2, 1-891562-13-4.

[16] Rimon Barr und Zygmunt J. Haas. JiST / SWANS: Java in simulation time / scalable wireless ad hoc network simulator. `http://jist.ece.cornell.edu/`, 2006.

[17] Michael Beigl und Hans Gellersen. Smart-its: An embedded platform for smart objects. In *Smart Objects Conference (SOC 2003), Grenoble, France*, Mai 2003.

[18] Michael Beigl, Tobias Zimmer, Albert Krohn, Christian Decker, und Philip Robinson. *Smart-Its – Communication and Sensing Technology for UbiComp Environments.* ISSN 1432-7864 2003/2, University of Karlsruhe, 2003.

[19] D. Elliott Bell und Leonard J. LaPadula. *Secure Computer Systems: Mathematical Foundations.* Technical report MTR-2547, Volume 1, MITRE corporation, März 1973.

[20] D. Elliott Bell und Leonard J. LaPadula. *Secure Computer Systems: Unified Exposition and MULTICS Interpretation.* Technical report MTR-2997 Rev. 1, MITRE corporation, März 1976.

[21] Steven M. Bellovin. A technique for counting natted hosts. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, Seite 267–272, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-603-X.

[22] Alastair Beresford und Frank Stajano. Mix zones: User privacy in location-aware services. In *PerCom Workshops*, Seite 127–131. IEEE Computer Society, 2004.

[23] Alastair R. Beresford und Frank Stajano. *Location Privacy in Pervasive Computing. IEEE Pervasive Computing*, 2(1):46–55, 2003. ISSN 1536-1268.

[24] Oliver Berthold, Hannes Federrath, und Stefan Köpsell. Web MIXes: A system for anonymous and unobservable internet access. In Federrath [85], Seite 115–129. ISBN 3-540-41724-9.

[25] Elisa Bertino, Elena Ferrari, und Anna Cinzia Squicciarini. Privacy-preserving trust negotiations. In Martin und Serjantov [156], Seite 283–301. ISBN 3-540-26203-2.

[26] Christian Bettstetter, Hannes Hartenstein, und Xavier Pérez-Costa. Stochastic properties of the random waypoint mobility model: epoch length, direction distribution, and cell change rate. In *MSWiM '02: Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, Seite 7–14, New York, NY, USA, 2002. ACM Press. ISBN 1-58113-610-2.

[27] Christian Bettstetter und Christian Wagner. The spatial node distribution of the random waypoint mobility model. In *German Workshop on Mobile Ad-Hoc Networks (WMAN)*, 2002.

[28] K. J. Biba. *Integrity Considerations for Secure Computer Systems.* Technical Report TR-3153, MITRE corporation, Bedford, MA, 1977.

[29] Matt Bishop. *Computer Security: Art and Science.* Addison Wesley, 2003.

[30] Matt Bishop. *Introduction to Computer Security.* Addison Wesley, 2005.

[31] Johann Bizer, Kai Dingel, Benjamin Fabian, Oliver Günther, Markus Hansen, Michael Klafft, Jan Möller, und Sarah Spiekermann. *TAUCIS - Technikfolgenabschätzung Ubiquitäres Computing und Informationelle Selbstbestimmung.* Technical report, Bundesministerium für Bildung und Forschung, Juli 2006.

[32] Nikita Borisov, Ian Goldberg, und Eric Brewer. Off-the-record communication, or, why not to use pgp. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, Seite 77–84, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-968-3.

[33] Stefan Brands, Liesje Demuynck, und Bart De Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *12th Australasian Conference on Information Security and Privacy*, 2007.

[34] Volker Braun. *Vertrauensbeziehungen in ubiquitären Systemen.* Diplomarbeit, Universität Ulm, 2006.

[35] D. F. C. Brewer und M. J. Nash. The chinese wall security policy. In *Proceedings / 1988 IEEE Symposium on Security and Privacy : April 18 - 21, 1988, Oakland, California*, Seite 206–214, 1988.

[36] Tobias Breyer. Modellierung der bewegung und des verhaltens von dienstnutzern in mobilen ad-hoc-netzen. Master's thesis, Friedrich-Schiller-Universität Jena, 2003.

[37] David Brogan und Nicholas Johnson. Realistic human walking paths. In *Computer Animation and Social Agents*, Seite 94 – 101, 2003.

[38] Klaus Brunnstein. *Über Möglichkeiten der Re-Identifikation von Personen aus Volkszählungsdaten.* FB-Mitteilung M-144, Universität Hamburg, Department Informatik, 1986.

[39] Klaus Brunnstein. *Über die Möglichkeit der Re-Identifikation von Personen aus Volkszählungsdaten*, Seite 62–81. Kölner Volksblatt Verlag, 1987. ISBN 3-92324331-6.

[40] Michael Burrows, Martín Abadi, und Roger M. Needham. *A Logic of Authentication.* SRC Research Report 39, DEC System Research Center, Februar 1989.

[41] Michael Burrows, Martín Abadi, und Roger M. Needham. *A Logic of Authentication. ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[42] Laurent Bussard, Refik Molva, und Yves Roudier. History-based signature or how to trust anonymous documents. In *iTrust 2004, 2nd International Conference on Trust Management, 29th March- 1st April 2004, St. Anne's College, Oxford, UK / Also published in Lecture Notes in Computer Science (LCNS), Volume 2995, Jensen, Christian; Poslad, Stefan; Dimitrakos, Theo (Eds.), Springer 2004, ISBN 3-540-21312-0*, März 2004.

[43] Jan Camenisch und Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, 2001.

[44] Jan Camenisch und Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO*, 2002.

[45] Jan Camenisch, Abhi Shelat, Dieter Sommer, Simone Fischer-Hübner, Marit Hansen, Henry Krasemann, Gérard Lacoste, Ronald Leenes, und Jimmy Tseng. Privacy and identity management for everyone. In *DIM '05: Proceedings of the 2005 workshop on Digital identity management*, Seite 20–27, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-232-1. Entstanden innerhalb des PRIME-Projects [4].

[46] Jan Camenisch und Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, number 1294 in Lecture Notes in Computer Science, Seite 410–424, London, UK, 1997. Springer-Verlag. ISBN 3-540-63384-7.

[47] Kim Cameron. *The Laws of Identity.* Technical report, Microsoft Corporation, 2005.

[48] Kim Cameron. *Microsoft's Vision for an Identity Metasystem*. Technical report, Microsoft Corporation, 2005.

[49] Roy H. Campbell, Jalal Al-Muhtadi, Prasad Naldurg, Geetanjali Sampemane, und M. Dennis Mickunas. Towards security and privacy for pervasive computing. In Mitsuhiro Okada, Benjamin C. Pierce, Andre Scedrov, Hideyuki Tokuda, und Akinori Yonezawa, editors, *ISSS*, volume 2609 of *Lecture Notes in Computer Science*, Seite 1–15. Springer, 2002. ISBN 3-540-00708-3.

[50] Thibault Candebat, Cameron Ross Dunne, und David T. Gray. Pseudonym management using mediated identity-based cryptography. In *DIM '05: Proceedings of the 2005 workshop on Digital identity management*, Seite 1–10, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-232-1.

[51] EPIC (Electronic Privacy Information Center). Pretty poor privacy: An assessment of p3p and internet privacy. `http://www.epic.org/reports/prettypoorprivacy.html`, 2000.

[52] David Chaum. *Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM*, 24(2):84–88, Februar 1981. ISSN 0001-0782.

[53] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, 1982.

[54] David Chaum. Blind signature system. In *CRYPTO*, 1983.

[55] David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In *EUROCRYPT*, Seite 241–244, 1985.

[56] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, und Sanjiva Weerawarana. Web services description language (wsdl) version 2.0, 2006.

[57] David R. Choffnes und Fabián E. Bustamante. An integrated mobility and traffic model for vehicular wireless networks. In *VANET '05: Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*, Seite 69–78, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-141-4.

[58] Cisco Press, editor. *Internetworking Technologies Handbook*. Macmillan Technical Publishing, 4th edition, 2003.

[59] D. D. Clark und D. R. Wilson. A comparison of commercial and military computer security policies. In *IEEE Symposium on Security and Privacy*, Seite 184–195, 1987.

[60] Roger Clarke. Introduction to dataveillance and information privacy, and definitions of terms. `http://www.anu.edu.au/people/Roger.Clarke/DV/Intro.html`, September 1999.

[61] Sebastian Clauß, Dogan Kesdogan, und Tobias Kölsch. Privacy enhancing identity management: protection against re-identification and profiling. In *DIM '05: Proceedings of the 2005 workshop on Digital identity management*, Seite 84–93, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-232-1.

[62] D. A. Cooper und K. P. Birman. Preserving privacy in a network of mobile computers. In *SP '95: Proceedings of the 1995 IEEE Symposium on Security and Privacy*, Seite 26. IEEE Computer Society, 1995.

[63] George Coulouris, Jean Dollimore, und Tim Kindberg. *Distributed Systems - Concepts and Design.* Addison Wesley, 1995.

[64] Karen Coyle. P3p: Pretty poor privacy? `http://www.kcoyle.net/p3p.html`, Juni 1999.

[65] Lorrie Cranor, Brooks Dobbs, Serge Egelman, Giles Hogben, Jack Humphrey, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, Joseph Reagle, Matthias Schunter, David A. Stampley, und Rigo Wenning. The platform for privacy preferences 1.1 (P3P1.1) specification. W3C Working Draft, Juli 2005.

[66] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, und Joseph Reagle. The platform for privacy preferences 1.0 (P3P) specification. W3C Recommendation, April 2002.

[67] Lorrie Faith Cranor, Joseph Reagle, und Mark S. Ackerman. *Beyond Concern: Understanding Net Users' Attitudes About Online Privacy.* Technical Report 99.4.3, AT&T Labs-Research, 1999.

[68] Joan Daemen und Vincent Rijmen. Rijndael for AES. In *The Third Advanced Encryption Standard Candidate Conference*, Seite 343–347, 2000.

[69] Thomas Daniels, Mani Mina, und Steve F. Russell. A signal fingerprinting paradigm for general physical layer and sensor network security and assurance. In *IEEE/Create-Net SecureComm 2005*, 2005.

[70] J. D. Day und H. Zimmermann. The osi reference model. In *Proceedings of the IEEE*, volume 71, Seite 1334 – 1340, Dezember 1983.

[71] Marco de Vivo, Gabriela O. de Vivo, und Germinal Isern. *Internet security attacks at the basic levels. SIGOPS Oper. Syst. Rev.*, 32(2):4–15, 1998. ISSN 0163-5980.

[72] Dorothy E. Denning. *A lattice model of secure information flow. Commun. ACM*, 19(5):236–243, 1976. ISSN 0001-0782.

[73] Dorothy E. Denning und Peter J. Denning. *Certification of programs for secure information flow. Commun. ACM*, 20(7):504–513, 1977. ISSN 0001-0782.

[74] Dorothy E. Denning und Giovanni Maria Sacco. *Timestamps in key distribution protocols. Commun. ACM*, 24(8):533–536, 1981. ISSN 0001-0782.

[75] T. Dierks und E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.1*, April 2006. RFC 4346.

[76] Whitfield Diffie und Martin E. Hellman. *New Directions in Cryptography. IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[77] Whitfield Diffie, Paul C. Van Oorschot, und Michael J. Wiener. *Authentication and authenticated key exchanges. Des. Codes Cryptography*, 2(2):107–125, 1992. ISSN 0925-1022.

[78] Roger Dingledine, Nick Mathewson, und Paul Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, Seite 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[79] Danny Dolev und Andrew Chi-Chih Yao. *On the security of public key protocols. IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[80] Stephan Dreiseitl, Staal Vinterbo, und Lucila Ohno-Machado. *Disambiguation Data: Extracting Information from Anonymized Sources. Journal of the American Medical Informatics Association*, 9:110 – 114, 2002.

[81] Yitao Duan und John F. Canny. Protecting user data in ubiquitous computing: Towards trustworthy environments. In Martin und Serjantov [156], Seite 167–185. ISBN 3-540-26203-2.

[82] Erin English und Scott Hamilton. *Network Security Under Siege: The Timing Attack. Computer*, 29(3):95–97, März 1996. ISSN 0018-9162.

[83] EPIC und Privacy International, editors. *Privacy & Human Rights 2003: An International Survey of Privacy Rights and Developments*. Electronic Privacy Information Center, 2003.

[84] EU project. The disappearing computer. `http://www.disappearing-computer.net/`, 1998.

[85] Hannes Federrath, editor. *Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 25-26, 2000, Proceedings*, volume 2009 of *Lecture Notes in Computer Science*, 2001. Springer. ISBN 3-540-41724-9.

[86] Hannes Federrath, Anja Jerichow, und Andreas Pfitzmann. MIXes in mobile communication systems: Location management with privacy. In *Information Hiding*, Seite 121–135, 1996.

[87] R. J. Feiertag. *A technique for proving specifications are multilevel secure.* Technical report CSL109, Computer Science Laboratory, SRI International, Menlo Park, California, Januar 1980.

[88] Niels Ferguson und Bruce Schneier. *Practical Cryptography.* John Wiley & Sons, Inc., New York, NY, USA, 2003. ISBN 0471223573.

[89] Michel Foucault. *Überwachen und Strafen. Die Geburt des Gefängnisses.* Suhrkamp-Verlag, 10. Auflage, 1994.

[90] Dan Frankowski, Dan Cosley, Shilad Sen, Loren Terveen, und John Riedl. You are what you say: privacy risks of public mentions. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, Seite 565–572, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-369-7.

[91] Edward L. Glaeser, David I. Laibson, José A. Scheinkman, und Christine L. Soutter. *Measuring Trust. The Quarterly Journal of Economics*, 115(3):811–846, August 2000.

[92] J. A. Goguen und J. Meseguer. Security policies and security models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, Seite 11–20, Los Alamitos, CA, USA, 1982. IEEE Computer Society.

[93] Dieter Gollmann. *Computer Security.* John Wiley & Sons, second edition, 2005.

[94] G. S. Graham und P. J. Denning. Protection: Principles and practices. In *Proceedings of the AFIPS Spring Joint Computer Conference*, volume 40, Seite 417–429, 1972.

[95] Marco Gruteser und Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*. USENIX, 2003.

[96] Marco Gruteser und Dirk Grunwald. *Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. Mobile Networks and Applications*, 10(3):315–325, Juni 2005. ISSN 1383-469X.

[97] Marco Gruteser und Baik Hoh. On the anonymity of periodic location samples. In Hutter und Ullmann [123], Seite 179–192. ISBN 3-540-25521-4.

[98] Carl A. Gunter, Michael J. May, und Stuart G. Stubblebine. A formal privacy system and its application to location based services. In Martin und Serjantov [156], Seite 256–282. ISBN 3-540-26203-2.

[99] Andreas Görlach, Andreas Heinemann, und Wesley W. Terpstra. *Privacy, Security and Trust within the Context of Pervasive Computing*, volume 780 of *The Kluwer International Series in Engineering and Computer Science*, Kapitel Survey on Location Privacy in Pervasive Computing. Springer, 2004.

[100] Andreas Görlach, Andreas Heinemann, und Wesley W. Terpstra. Survey on location privacy in pervasive computing. In Philip Robinson, Harald Vogt, und Waleed Wagealla, editors, *Privacy, Security and Trust within the Context of Pervasive Computing*, The Kluwer International Series in Engineering and Computer Science, Seite 23–34. Kluwer Academic Publishers, 2005.

[101] Monika Göttle. *Location Privacy in ubiquitären Systemen.* Diplomarbeit, Universität Ulm, 2006.

[102] Jeyanthi Hall, Michel Barbeau, und Evangelos Kranakis. Detection of transient in radio frequency fingerprinting using signal phase. In *IASTED International conference on Wireless and Optical Communications (WOC 2003)*, 2003.

[103] Christian Hammer, Jens Krinke, und Gregor Snelting. Information flow control for java based on path conditions in dependence graphs. In *IEEE International Symposium on Secure Software Engineering*, 2006.

[104] Marit Hansen, Henry Krasemann, Christian Krause, Martin Rost, und Dr. Riccardo Genghini. *Identity Management Systems (IMS): Identification and Comparison Study.* Technical report, Independent Centre for Privacy Protection (ICPP), 2004.

[105] D. Harkis und D. Carrel. *The Internet Key Exchange (IKE).* IETF - Networking Group, November 1998. RFC 2049.

[106] Harris Interactive. IBM multi-national consumer privacy survey, Oktober 1999.

[107] Michael A. Harrison, Walter L. Ruzzo, und Jeffrey D. Ullman. *Protection in Operating Systems. Commun. ACM*, 19(8):461–471, 1976.

[108] Katia Hayati und Martín Abadi. Language-based enforcement of privacy policies. In Martin und Serjantov [156], Seite 302–313. ISBN 3-540-26203-2.

[109] Mike Hazas, John Krumm, und Thomas Strang, editors. *Location- and Context-Awareness, Second International Workshop, LoCA 2006, Dublin, Ireland, May 10-11, 2006, Proceedings*, volume 3987 of *Lecture Notes in Computer Science*, 2006. Springer. ISBN 3-540-34150-1.

[110] Dirk Helbing. Models for pedestrian behavior. In *Natural Structures. Principles, Strategies, and Models in Architecture and Nature*, volume Part II, Seite 93–98. Mitteilungen des SFB 230, Stuttgart, 1992.

[111] Dirk Helbing und Peter Molnar. *Social Force Model for Pedestrian Dynamics. Physical Review E*, 51:4282, 1995.

[112] Dirk Helbing, Peter Molnar, und Frank Schweitzer. Computer simulations of pedestrian dynamics and trail formation. In *Evolution of Natural Structures*, Seite 229–234. Proceedings of the 3rd International Symposium of the SFB 230, (Mitteilungen des SFB 230, Heft 9), Stuttgart, 1994.

[113] L. F. Henderson. *The statistics of crowd fluids. Nature*, 229:381 – 383, 1971.

[114] Jaap-Henk Hoepman. The ephemeral pairing problem. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, Seite 212–226. Springer, 2004. ISBN 3-540-22420-3.

[115] Jaap-Henk Hoepman. Ephemeral pairing on anonymous networks. In Hutter und Ullmann [123], Seite 101–116. ISBN 3-540-25521-4.

[116] Baik Hoh und Marco Gruteser. Protecting location privacy through path confusion. In *IEEE/CreateNet Intl. Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm)*, 2005.

[117] Jason Hong, Gaetano Borriello, James Landay, David McDonald, Bill Schilit, und Doug Tygar. Privacy and security in the location-enhanced world wide web. In *Fifth International Conference on Ubiquitous Computing*, 2003.

[118] Jason I. Hong und James A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, Seite 177–189, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-793-1.

[119] Jason I. Hong, Jennifer D. Ng, Scott Lederer, und James A. Landay. Privacy risk models for designing privacy-sensitive ubiquitous computing systems. In David Benyon, Paul Moody, Dan Gruen, und Irene McAra-McWilliam, editors, *Conference on Designing Interactive Systems*, Seite 91–100. ACM, 2004. ISBN 1-58113-787-7.

[120] Andy Hopper, Andy Harter, und Tom Blackie. The active badge system (abstract). In *CHI '93: Proceedings of the SIGCHI conference on Human factors in computing systems*, Seite 533–534, New York, NY, USA, 1993. ACM Press. ISBN 0-89791-575-5.

[121] Eric Hughes. A cypherpunk's manifesto. `http://www.activism.net/cypherpunk/manifesto.html`, März 1993.

[122] Sebastian Hunt und David Sands. On flow-sensitive security types. In *POPL '06: Conference record of the 33rd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, Seite 79–90, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-027-2.

[123] Dieter Hutter und Markus Ullmann, editors. *Security in Pervasive Computing, Second International Conference, SPC 2005, Boppard, Germany, April*

6-8, 2005, Proceedings, volume 3450 of *Lecture Notes in Computer Science*, 2005. Springer. ISBN 3-540-25521-4.

[124] ISO/IEC JTC1 SC17 WG3/TF1. *Machine Readable Travel Document specification 9303*. Technical report, ICAO, 1997. Übernommen als ISO/IEC 7501.

[125] Xiaodong Jiang, Jason I. Hong, und James A. Landay. Approximate information flows: Socially-based modeling of privacy in ubiquitous computing. In Gaetano Borriello und Lars Erik Holmquist, editors, *Ubicomp*, volume 2498 of *Lecture Notes in Computer Science*, Seite 176–193. Springer, 2002. ISBN 3-540-44267-7.

[126] Ari Juels. RFID security and privacy: A research survey. Manuscript, September 2005.

[127] Ari Juels und John Brainard. Soft blocking: flexible blocker tags on the cheap. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, Seite 1–7, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-968-3.

[128] Ari Juels, David Molnar, und David Wagner. Security and privacy issues in e-passports. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm*, Athens, Greece, September 2005. IEEE.

[129] Ari Juels, Ronald L. Rivest, und Michael Szydlo. The blocker tag: selective blocking of RFID tags for consumer privacy. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, Seite 103–111, New York, NY, USA, 2003. ACM Press. ISBN 1-58113-738-9.

[130] Lalana Kagal, Massimo Paoucci, Naveen Srinivasan, Grit Denker, Tim Finin, und Katia Sycara. *Authorization and Privacy for Semantic Web Services. IEEE Intelligent Systems (Special Issue on Semantic Web Services)*, 19(4): 50–56, Juli 2004.

[131] Rudolf E. Kalman. A new approach to linear filtering and prediction problems. In *Transactions of the ASME - Journal of Basic Engineering*, number 82, Seite 35 – 45, 1960.

[132] Frank Kargl, Torsten Illmann, Stefan Schlott, Matthias Zeile, und Alfred Geiß. Smartreminder - personal assistance in a mobile computing environment. Poster session and demonstration, Pervasive 2002, Switzerland, 2002.

[133] Sindhu Karthikeyan und Mikhail Nesterenko. RFID security without extensive cryptography. In *SASN '05: Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, Seite 63–67, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-227-5.

[134] Richard Kemmerer, Catherine Meadows, und Jonathan Millen. *Three systems for cryptographic protocol analysis. Journal of Cryptology*, 7(2):79–130, 1994.

[135] B. S. Kerner und P. Konhäuser. *Cluster effect in initially homogeneous traffic flow. Physical Review*, E 48(4):2335 ff, Oktober 1993.

[136] Dogan Kesdogan, Peter Reichl, und Klaus Junghärtchen. Distributed temporary pseudonyms: A new approach for protecting location information in mobile communication networks. In Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows, und Dieter Gollmann, editors, *ESORICS*, volume 1485 of *Lecture Notes in Computer Science*, Seite 295–312. Springer, 1998. ISBN 3-540-65004-0.

[137] Ziv Kfir und Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard systems. In *Conference on Security and Privacy for Emerging Areas in Communication Networks – SecureComm 2005*, Athens, Greece, September 2005. IEEE.

[138] Anya Kim, Lance J. Hoffman, und C. Dianne Martin. Building privacy into the semantic web: An ontology needed now. In *Semantic Web Workshop 2002*, 2002.

[139] Anya Kim, Jim Luo, und Myong Kang. Security ontology for annotating resources. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Özalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, und Stefano Spaccapietra, editors, *OTM Conferences (2)*, volume 3761 of *Lecture Notes in Computer Science*, Seite 1483–1499. Springer, 2005. ISBN 3-540-29738-3.

[140] Dominik Alexander Klein. People tracking. In *Lernen und Planen für Autonome mobile Roboter*. Rheinische Friedrich-Wilhelms-Universität Bonn, 2007.

[141] Mirko Knoll, Torben Weis, Andreas Ulbrich, und Alexander Brändle. Scripting your home. In Hazas et al. [109], Seite 274–288. ISBN 3-540-34150-1.

[142] Donald E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison Wesley, 2nd edition, 1998.

[143] Paul C. Kocher, Joshua Jaffe, und Benjamin Jun. Differential power analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, volume 1666, Seite 388–397, London, UK, 1999. Springer-Verlag.

[144] Tobias Kölsch, Lothar Fritsch, Markulf Kohlweiss, und Dogan Kesdogan. Privacy for profitable location based services. In Hutter und Ullmann [123], Seite 164–178. ISBN 3-540-25521-4.

[145] Dennis Kügler. Security concept of the EU-passport. In Hutter und Ullmann [123], Seite 85. ISBN 3-540-25521-4.

[146] Dennis Kügler. *Risiko Reisepass. c't*, 3:84 – 89, Februar 2005.

[147] Marc Langheinrich. Privacy by design - principles of privacy-aware ubiquitous systems. In Gregory D. Abowd, Barry Brumitt, und Steven A. Shafer, editors, *Ubicomp*, volume 2201 of *Lecture Notes in Computer Science*, Seite 273–291. Springer, 2001. ISBN 3-540-42614-0.

[148] Marc Langheinrich. When trust does not compute – the role of trust in ubiquitous computing. In *Workshop on Privacy at Ubicomp 2003*, Seattle, Washington, Oktober 2003.

[149] Marc Langheinrich. *Personal Privacy in Ubiquitous Computing – Tools and System Support*. PhD thesis, ETH Zurich, Zurich, Switzerland, Mai 2005.

[150] Kenneth C. Laudon. *Markets and privacy. Commun. ACM*, 39(9):92–104, 1996. ISSN 0001-0782.

[151] Cedric Laurant. Privacy and human rights 2003. `http://www.privacyinternational.org/survey/phr2003/`, 2003.

[152] Scott Lederer, Jason I. Hong, Anind K. Dey, und James A. Landay. *Personal privacy through understanding and action: five pitfalls for designers. Personal and Ubiquitous Computing*, 8(6):440–454, 2004.

[153] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, und L. Jones. SOCKS protocol version 5. Internet Engineering Task Force: RFC 1928, März 1996.

[154] Lawrence Lessig. *The Architecture of Privacy. Vanderbilt Entertainment Law and Practice*, 1, 1999.

[155] Steven B. Lipner. Non-discretionary controls for commercial applications. In *IEEE Symposium on Security and Privacy*, Seite 2–10, 1982.

[156] David Martin und Andrei Serjantov, editors. *Privacy Enhancing Technologies, 4th International Workshop, PET 2004, Toronto, Canada, May 26-28, 2004, Revised Selected Papers*, volume 3424 of *Lecture Notes in Computer Science*, 2005. Springer. ISBN 3-540-26203-2.

[157] John McHugh und Donald L. Good. An information flow tool for gypsy. In *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, Seite 46, Los Alamitos, CA, USA, 1985. IEEE Computer Society.

[158] James Michael. *Privacy and Human Rights: An International and Comparative Study, with Special Reference to Developments in Information Technology*. Dartmouth, 1994.

[159] David Molnar und David Wagner. Privacy and security in library RFID: Issues, practices, and architectures. In Birgit Pfitzmann und Peng Liu, editors, *Conference on Computer and Communications Security – ACM CCS*, Seite 210–219, Washington, DC, USA, Oktober 2004. ACM, ACM Press.

[160] Andrew C. Myers. Jflow: practical mostly-static information flow control. In *POPL '99: Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, Seite 228–241, New York, NY, USA, 1999. ACM Press. ISBN 1-58113-095-3.

[161] Andrew C. Myers und Barbara Liskov. *Protecting privacy using the decentralized label model. ACM Trans. Softw. Eng. Methodol.*, 9(4):410–442, 2000. ISSN 1049-331X.

[162] Roger M. Needham und Michael D. Schroeder. *Using encryption for authentication in large networks of computers. Commun. ACM*, 21(12):993–999, 1978. ISSN 0001-0782.

[163] Roger M. Needham und Michael D. Schroeder. *Authentication revisited. SIGOPS Oper. Syst. Rev.*, 21(1):7–7, 1987. ISSN 0163-5980.

[164] Dan M. Nessett. *A Critique of the Burrows, Abadi and Needham Logic. ACM Operating Systems Review*, 24(2):35–38, 1990.

[165] C. Neuman, T. Yu, S. Hartman, und K. Raeburn. The kerberos network authentication service. Internet Engineering Task Force: RFC 4120, Juli 2005.

[166] Flemming Nielson, Hanne Riis Nielson, und Chris Hankin. *Principles of Program Analysis*. Springer, corrected 2nd printing, 2005.

[167] Donald A. Norman. *The Invisible Computer*. The MIT Press, Cambridge, Massachusetts; London, England, 1998.

[168] William Pitt (Earl of Chatham). Speech on the excise bill. In John Bartlett, editor, *Familiar Quotations, 10th ed.*, 1919.

[169] Miyako Ohkubo, Koutarou Suzuki, und Shingo Kinoshita. *RFID privacy issues and technical challenges. Commun. ACM*, 48(9):66–71, 2005. ISSN 0001-0782.

[170] Shigeyuki Okazaki. *A study of pedestrian movement in architectural space, part 1: Pedestrian movement by the application of magnetic models. Transactions of the Architectural Institute of Japan*, (283):111 – 119, 1979.

[171] Kevin R. O'Neill, Michael R. Clarkson, und Stephen Chong. Information-flow security for interactive programs. In *Computer Security Foundations Workshop*, Seite 190–201, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[172] Nuria Pelechano, Kevin O'Brien, Barry Silverman, und Norman Badler. Crowd simulation incorporating agent psychological models, roles and communication. In *First International Workshop on Crowd Simulation*, 2005.

[173] C. Perkins. IP mobility support for IPv4. Internet Engineering Task Force: RFC 3220, Januar 2002.

[174] Andreas Pfitzmann und Marit Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In Federrath [85], Seite 1–9. ISBN 3-540-41724-9.

[175] Andreas Pfitzmann und Marit Köhntopp. Anonymity, unlinkability, unobservability, pseudonymity, and identity management - a consolidated proposal for terminology. `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`, August 2005.

[176] Charles P. Pfleeger und Shari Lawrence Pfleeger. *Security in Computing.* Prentice Hall, 3rd edition, 2003.

[177] Matthew Pirretti, Sencun Zhu, N. Vijaykrishnan, P. McDaniel, M. Kandemir, und R. Brooks. The sleep deprivation attack in sensor networks: Analysis and methods of defense. In *Conference on Innovations and Commercial Applications of Distributed Sensor Networks*, October 2005.

[178] Jon Postel. Internet protocol - DARPA internet program protocol specification. Internet Engineering Task Force: RFC 791, September 1981.

[179] Ilya Prigogine und Robert C. Herman. *Kinetic Theory of Vehicular Traffic.* American Elsevier, 1971.

[180] The Pew Internet & American Life Project. Trust and privacy online: Why americans want to rewrite the rules. `http://www.pewinternet.org/PPF/r/19/report_display.asp`, August 2000.

[181] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues, and open problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, Seite 10–29. Springer-Verlag, LNCS 2009, July 2000.

[182] Donald B. Reid. *An Algorithm for Tracking Multiple Targets. IEEE Tran. on Automatic Control*, 24(6):843–854, Dezember 1979.

[183] Craig Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference 1999*, 1999.

[184] Melanie R. Rieback, Bruno Crispo, und Andrew S. Tanenbaum. RFID guardian: A battery-powered mobile device for RFID privacy management. In *Proc. 10th Australasian Conf. on Information Security and Privacy (ACISP 2005)*, volume 3574 of *LNCS*, Seite 184–194. Springer-Verlag, Juli 2005.

[185] R. L. Rivest, A. Shamir, und L. Adleman. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the Association for Computing Machinery*, 21(2):120 – 126, Februar 1978. ISSN 0001-0782.

[186] Ronald L. Rivest. Whither information security? `http://wean1.ulib.org/Lectures/DistinguishedLectures/2001/03.0RonaldLRivest/`, März 2001. Invited talk at the SCS Distinguished Lecture Series.

[187] Marc Rotenberg und Cedric Laurant. Privacy and human rights 2004. `http://www.privacyinternational.org/survey/phr2004/`, 2004.

[188] Beate Rössler. *Der Wert des Privaten*. Suhrkamp Verlag, 2001. ISBN 978-3-518-29130-6.

[189] Beate Rössler. *Privat!*, Kapitel 1: Der Wert des Privaten, Seite 15 – 32. Heise Verlag, 2003. ISBN 3-936931-01-1.

[190] Francois Jegou Saadi Lahlou. *European Disappearing Computer Privacy Design Guidelines*. Technical report, Ambient Agoras Project, November 2003.

[191] A. Sabelfeld und A. C. Myers. *Language-Based Information-Flow Security. IEEE J. Selected Areas in Communications*, 21(1):5–19, Januar 2003.

[192] Stefan Schlott und Frank Kargl. Wiedererkennung anonymer knoten. In *3. Krypto-Tag - Workshop über Kryptographie*, volume Technical Report No. TI-1/05, 2005.

[193] Stefan Schlott, Frank Kargl, und Michael Weber. Random IDs for preserving location privacy. In *SecureComm*, Seite 415 – 417, 2005.

[194] Stefan Schlott, Frank Kargl, und Michael Weber. Re-identifying anonymous nodes. In Hazas et al. [109], Seite 103–115. ISBN 3-540-34150-1.

[195] Steve Schneider. *Modelling security properties with CSP*. Technical Report CSD-TR-96-04, Royal Holloway, University of London, Department of Computer Science, Egham, Surrey TW20 0EX, England, 1996.

[196] Steve Schneider. Security properties and csp. In *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*, Seite 174, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7417-2.

[197] Bruce Schneier. Crypto-gram. `http://www.schneier.com/crypto-gram-0109.html`, September 2001.

[198] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, und Niels Ferguson. Comments on Twofish as an AES candidate. In *The Third Advanced Encryption Standard Candidate Conference*, Seite 355–356, 2000.

[199] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, und Niels Ferguson. *Twofish: A 128-Bit Block Cipher*. Technical report, Counterpane Systems, Juni 1998.

[200] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, und Niels Ferguson. Performance comparison of the AES submissions. In *Second AES Candidate Conference (AES2)*, 2000.

[201] Bruce W. Schneier. *Applied Cryptography*. Wiley, 2nd edition, 1996.

[202] Matthias Schunter, Paul Ashley, Satoshi Hada, Günter Karjoth, und Calvin Powers. *Enterprise Privacy Authorization Language (EPAL 1.0)*. Research Report RZ 3485 (#93951), IBM, März 2003.

[203] Uwe Schöning. *Theoretische Informatik kurz gefaßt*. BI Wissenschaftsverlag, 1993.

[204] Jean-Marc Seigneur, Stephen Farrell, und Christian Damsgaard Jensen. Secure ubiquitous computing based on entity recognition. In *UBICOMP2002 - Workshop on Security in Ubiquitous Computing*, 2002.

[205] Jean-Marc Seigneur, Stephen Farrell, Christian Damsgaard Jensen, Elizabeth Gray, und Yong Chen. End-to-end trust starts with recognition. In *First International Conference on Security in Pervasive Computing*, März 2003.

[206] Jean-Marc Seigneur und Christian Damsgaard Jensen. Trading privacy for trust. In Christian D. Jensen, Stefan Poslad, und Theodosis Dimitrakos, editors, *iTrust*, volume 2995 of *Lecture Notes in Computer Science*, Seite 93–107. Springer, 2004. ISBN 3-540-21312-0.

[207] Jean-Marc Seigneur und Christian Damsgaard Jensen. Trust enhanced ubiquitous payment without too much privacy loss. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, Seite 1593–1599, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-812-1.

[208] Claude Shannon. *Communication Theory of Secrecy Systems. The Bell System Technical Journal*, 28(4):656–715, 1949.

[209] Gustavus J. Simmons. How to (selectively) broadcast a secret. In *Proceedings of the 1985 IEEE Symposium on Security and Privacy*, Seite 108–113, 1985.

[210] Polly Sprenger. Sun on privacy: 'Get Over It'. `http://www.wired.com/news/politics/0,1283,17538,00.html`, Januar 1999.

[211] W. Stadje. *The exact probability distribution of a two-dimensional random walk. Journal of Statistical Physics*, 46, 1987.

[212] Frank Stajano. The resurrecting duckling - what next? In B. Christianson, B. Crispo, und M. Roe, editors, *Security Protocols, 8th International Workshop Proceedings*, Seite 204ff., 2000.

[213] Frank Stajano. *Security for ubiquitous computing.* Wiley, 2002.

[214] Frank Stajano und Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In B. Christianson, B. Crispo, und M. Roe, editors, *Security Protocols, 7th International Workshop Proceedings*, Seite 172–194, 1999.

[215] Richard Stevens. *TCP/IP illustrated*, volume 1. Addison Wesley, 1994.

[216] Douglas R. Stinson. *Cryptography: Theory and Practice.* CRC Press, Inc., Boca Raton, FL, USA, 1995. ISBN 0849385210.

[217] Clifford Stoll. *Cuckoo's Egg.* Doubleday, 1989. ISBN 0385249462.

[218] William H. Stufflebeam, Annie I. Antón, Qingfeng He, und Neha Jain. Specifying privacy policies with P3P and EPAL: lessons learned. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, Seite 35–35, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-968-3.

[219] Dirk Stüker. *Heterogene Sensordatenfusion zur robusten Objektverfolgung im automobilen Straßenverkehr.* PhD thesis, Carl von Ossietzky-Universität Oldenburg, 2004.

[220] Girish Suryanarayana und Richard Taylor. *A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications.* Technical Report UCI-ISR-03-9, Institute for Software Research, University of California, Irvine, 2004.

[221] Latanya Sweeney. *Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10 (5):571–588, 2002. ISSN 0218-4885.

[222] Latanya Sweeney. *k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002. ISSN 0218-4885.

[223] Andrew S. Tanenbaum. *Distributed Operating Systems.* Prentice Hall, 1995.

[224] Humphrey Taylor. The harris poll. `http://www.harrisinteractive.com/harris_poll/index.asp?PID=365`, März 2003. Technical Report 17, HarrisInteractive.

[225] Kardi Teknomo, Yasushi Takeyama, und Hajime Inamura. Review on microscopic pedestrian simulation model. In *Proceedings Japan Society of Civil Engineering Conference*, 2000.

[226] Sotirios Terzis, Waleed Wagealla, Colin English, und Paddy Nixon. Trust lifecycle management in a global computing environment. In Corrado Priami und Paola Quaglia, editors, *Global Computing*, volume 3267 of *Lecture Notes in Computer Science*, Seite 291–313. Springer, 2004. ISBN 3-540-24101-9.

[227] TNS EMNID. TNS EMNID untersucht die Akzeptanz von Kundenkarten unter deutschen Verbrauchern. `http://www.tns-emnid.com/pdf/presse-presseinformationen/2002/2002_04_26_TNS_Emnid_Kundenkarten.pdf`, März 2002.

[228] Krishna Venkatasubramanian und Sandeep Gupta. Security for pervasive healthcare. In *Security in Distributed, Grid, Mobile, and Pervasive Computing*, 2007.

[229] Dennis Volpano, Cynthia Irvine, und Geoffrey Smith. *A sound type system for secure flow analysis. J. Comput. Secur.*, 4(2-3):167–187, 1996. ISSN 0926-227X.

[230] Bernhard Walke. *Mobilfunknetze und ihre Protokolle.* Teubner, 2001.

[231] Samuel Warren und Louis Brandeis. *The Right to Privacy.* Technical report, Harvard Law Review, Dezember 1890.

[232] David Watson, Matthew Smart, G. Robert Malan, und Farnam Jahanian. *Protocol scrubbing: network security through transparent flow modification. IEEE/ACM Trans. Netw.*, 12(2):261–273, 2004. ISSN 1063-6692.

[233] Ulrich Weidmann. *Transporttechnik der Fußgänger.* Schriftenreihe des Instituts für Verkehrsplanung, Transporttechnik, Straßen- und Eisenbahnbau 90, ETH Zürich, 1993.

[234] Mark Weiser. *The Computer for the Twenty-First Century. Scientific American*, 9:94–110, September 1991.

[235] Mark Weiser. *Some Computer Science Issues in Ubiquitous Computing. Commun. ACM*, 36(7):74–84, 1993.

[236] Uwe Wesel. *Geschichte des Rechts. Von den Frühformen bis zur Gegenwart.* Beck Verlag, 2001. ISBN 3-406-47543-4.

[237] Jan Wessels. Applications of BAN-logic. In *IPA Spring Days on Security*, 2001.

[238] Alan Westin. *Privacy and Freedom.* Bodley Head, 1967.

[239] Heike Wirth. *Die faktische Anonymität von Mikrodaten.* Number 19 in Schriftenreihe Forum der Bundesstatistik. Metzler-Poeschel, 1991.

[240] Ryan Wishart, Karen Henricksen, und Jadwiga Indulska. Context obfuscation for privacy via ontological descriptions. In Thomas Strang und Claudia Linnhoff-Popien, editors, *LoCA*, volume 3479 of *Lecture Notes in Computer Science*, Seite 276–288. Springer, 2005. ISBN 3-540-25896-5.

[241] Hao Wu, Richard Fujimoto, Randall Guensler, und Michael Hunter. Mddv: a mobility-centric data dissemination algorithm for vehicular networks. In *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, Seite 47–56, New York, NY, USA, 2004. ACM Press. ISBN 1581139225.

[242] Ye Zhu, Xinwen Fu, Riccardo Bettati, und Wei Zhao. Anonymity analysis of mix networks against flow-correlation attacks. In *Global Telecommunications Conference (GLOBECOM) 2005*, volume 3, Seite 5–9, 2005.

C Danksagung

> Keine Aufgabe, die es wirklich Wert war getan zu werden, wurde jemals pünktlich oder innerhalb ihres Budgets fertiggestellt.
>
> *Khufu, ägyptischer Pharao (2590 – 2568 v. Chr.), über den Bau der Cheops-Pyramide*

Kreativität funktioniert nicht auf Knopfdruck, und so entstehen auch Arbeiten wie diese nicht am Schreibtisch im Rahmen eines Bürojobs. Es bedarf der richtigen Mischung aus Freiraum und Routine, Zerstreuung und konzentrierter Arbeit, aus kreativem Gespräch und kritischem Feedback. Die Chance, in einer solchen Umgebung arbeiten zu dürfen, ist nicht selbstverständlich – dafür möchte ich mich beim Betreuer meiner Arbeit, Prof. Michael Weber, herzlich bedanken. Die Arbeit in seiner Abteilung, sowohl in Projekten, in der Lehre als auch an meiner Promotion, möchte ich nicht missen. Ebenso gilt mein Dank Volker Birk von der Firma logix-tt, welcher mir die Fertigstellung dieser Arbeit nach meiner Anstellung an der Universität Ulm ermöglichte.

Zum Umfeld einer Abteilung zählen selbstverständlich die Kollegen. Auch bei ihnen will ich mich bedanken: Für kreativen Austausch; den einen oder anderen Gedankenanstoß, welcher bei den Gesprächen entstand; die gegenseitige Ermutigung; und die Möglichkeit, in einer verfahrenen Situation sich seinen Frust von der Seele reden zu können. Mein besonderer Dank gilt hierbei Thorsten Mahler und Frank Kargl, die immer etwas Zeit für eine kurze Diskussion erübrigen konnten.

Weiterhin gilt mein Dank den Studenten, welche direkt oder indirekt zu dieser Arbeit beigetragen haben. Die Diskussionen mit Monika Göttle über ihre Diplomarbeit haben auch meine Tätigkeit vorangebracht, und Björn Wiedersheim stand mir mit seiner Erfahrung bei der Programmierung von JiST/SWANS ebenfalls mehrfach zur Seite.

Dankeschön sagen möchte ich auch meiner Mutter und Josef Kovacs, die sich diese Ausarbeitung durchgelesen und akribisch nach Schreibfehlern durchsucht haben. Vielen Dank für die Ausdauer, die insbesondere ein Fachfremder hierfür aufbringen muß.

Zuletzt gilt mein herzlicher Dank meinen Eltern und meiner Schwester. Ohne ihre fortwährende Unterstützung wäre ich sicherlich nicht da, wo ich heute stehe.