Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Masterarbeit

Entwicklung einer datenschutz-freundlichen Ausführungsumgebung für Smart-Home-Dienste

Marcel Mayer

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Bernhard Mitschang
Betreuer/in:	Dr. Christoph Stach
Beginn am:	17. Oktober 2018
Beendet am:	17. April 2019

Kurzfassung

In einem Smart Home werden Alltagsgegenstände miteinander vernetzt und mit Sensoren und Aktuatoren ausgestattet, um so die Lebensqualität der Bewohner zu erhöhen. Auf diese Weise werden jedoch auch viele private Daten erfasst, die, wenn sie miteinander kombiniert werden, ein Risiko für die Privacy des Nutzers darstellen und zum Nachteil des Nutzers eingesetzt werden können.

Mit PATRON und der PMP gibt es bereits Systeme, die den Zugriff auf bestimmte Muster beschränken, die Weitergabe der Sensordaten verhindern und die Verwendung der Aktuatoren begrenzen können. Es existiert jedoch kein System für das Smart-Home-Umfeld, das gleichzeitig Sensordaten verfremden oder blockieren, Muster in Sensordaten verschleiern und die Verwendung der Aktuatoren begrenzen kann.

Deshalb stellt diese Arbeit das Privacy-System SPYaware vor, das PATRON mit der PMP kombiniert und in das Regelverarbeitungssystem MIALinx integriert. Dabei werden das Berechtigungsrichtlinienmodell der PMP und dessen Erweiterung ACCESSORS dazu genutzt, die Sensoren und Aktuatoren zu reglementieren. Mithilfe der Zugriffssteuerung von PATRON können private Muster verschleiert werden, bevor sie das Regelverarbeitungssystem erreichen. Die Verteilung der Privacy-Einstellungen auf die einzelnen Geräte erfolgt analog zu dem Verteilungsmechanismus von AVARE.

Um die Realisierung des Konzepts anhand eines realitätsnahen Anwendungsfalls aus dem Smart-Home-Umfeld zu demonstrieren, wird SPYaware prototypisch implementiert. Die anschließende Evaluation zeigt, dass SPYaware durch die Kombination von PATRON und der PMP die Privacy des Nutzers in diesem Anwendungsfall gewährleisten kann. Die Evaluation zeigt allerdings auch, dass es nicht immer möglich ist, die Privacy des Nutzers vollständig sicherzustellen, ohne die Servicequalität der Anwendung zu beeinträchtigen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACCESSORS	**A** Data-**C**entri**c** P**e**rmi**ss**ion M**o**del fo**r** the Internet of Thing**s**
AD-RMS	**A**ctive **D**irectory **R**ights **M**anagement **S**ervices
AVARE	**A**nwendung zur **V**erteilung und **A**uswahl **r**echtskonformer Datenschutz-**e**instellungen
CHARIOT	**C**loud-Assisted **A**ccess Cont**r**ol for the **I**nternet **o**f **T**hings
CSV	**C**omma-**s**eparated **v**alues
DB	**D**aten**b**ank
DSGVO	**D**aten**s**chutz-**G**rund**ver**o**r**dnung
ERP	**E**nterprise **R**esource **P**lanning
GPS	**G**lobal **P**ositioning **S**ystem
IoT	**I**nternet **o**f **T**hings
IP	**I**nternet **P**rotokoll
MAC	**M**edia **A**ccess **C**ontrol
MIALinx	**M**anufacturing **I**ntegration **A**ssistant
PATRON	**P**riv**a**cy in S**tr**eam Pr**o**cessi**n**g
PMP	**P**rivacy **M**anagement **P**latform
PPM	**P**rivacy **P**olicy **M**odel
RFID	**R**adio-**F**requency **Id**entification
SMS	**S**hort **M**essage **S**ervice
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol
SPS	**S**peicher**p**rogrammierbare **S**teuerung
SPYaware	**S**mart Home **P**rivac**y**-**awa**re **R**untime **E**nvironment
SQL	**S**tructured **Q**uery **L**anguage

THOR	Da**t**ensc**h**utzk**o**nzept für die Indust**r**ie 4.0
UI	**U**ser **I**nterface
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
XML	E**x**tensible **M**arkup **L**anguage

1 Einleitung

In einem Smart Home sind Alltagsgegenstände wie Heizung, Lampen, Jalousien oder Kühlschrank miteinander vernetzt und mit Sensoren und Aktuatoren ausgestattet [VRM+17]. Diese Gegenstände werden dadurch zu *intelligenten Geräten* (engl. Smart Devices). Das Ziel ist es, die Lebensqualität und die Sicherheit der Bewohner zu erhöhen und Energie effizienter zu nutzen [DLY+06]. Die Sensoren ermöglichen die kontinuierliche Erfassung von Kontextinformationen. Diese Kontextinformationen werden an ein Datenstromsystem gesendet, das die Daten in Echtzeit analysiert [VRM+17] und die Aktuatoren entsprechen zuvor definierter Regeln steuert [CFS+14].

Die Sensoren und Aktuatoren können beispielsweise genutzt werden, um zu erkennen, wenn der Nutzer nach Hause kommt. Dazu wird der Standort des Nutzers mithilfe des GPS-Sensors in seinem Smartphone kontinuierlich erfasst. Durch das *Internet der Dinge* (engl. Internet of Things, IoT) können die Daten aus unterschiedlichen Datenquellen miteinander kombiniert werden, wodurch eine große Menge an Daten erfasst wird [CBM+15]. In diesen Daten kann das Datenstromsystem Muster erkennen [KLC02] und eine zuvor definierte Aktion auslösen. Muster sind bestimmte Abfolgen von Sensordaten, aus denen weiteres Wissen abgeleitet werden kann [SDM+17]. Aus den Standortinformationen kann beispielsweise das Bewegungsmuster abgeleitet werden. Bewegt sich der Nutzer auf das Haus zu, kann rechtzeitig bevor der Nutzer nach Hause kommt die Heizung eingeschaltet werden.

Die Sensoren in einem Smart Home erfassen auch viele private Daten, die, wenn sie miteinander kombiniert werden, ein Risiko für die Datensicherheit und die Privacy des Nutzers darstellen können [ZCW+14]. Als Privacy bezeichnet man die Fähigkeit einer Person, selbst zu bestimmen, welche persönlichen Informationen sie preisgibt und gegenüber wem sie das tut [Moo08]. Das im vorherigen Beispiel erfasste Bewegungsmuster kann über die Aktuatoren verbreitet und zum Nachteil des Nutzers eingesetzt werden. Beispielsweise könnte die Versicherungsgesellschaft des Nutzers anhand des Bewegungsmusters erkennen, welche Straßen der Nutzer befahren hat und ob er dabei die zulässige Höchstgeschwindigkeit überschritten hat. Das Überschreiten der zulässigen Höchstgeschwindigkeit könnte zur Erhöhung der Versicherungsbeiträge führen.

1.1 Aufgabenstellung

Mit PATRON [SDM+17] und der PMP [SM13] gibt es bereits Systeme, die den Zugriff auf bestimmte Muster beschränken, die Weitergabe der Sensordaten verhindern und die Verwendung der Aktuatoren begrenzen können. Jedoch existiert kein System für das Smart-

Home-Umfeld, das gleichzeitig Sensordaten verfremden oder blockieren, Muster in Sensordaten aus mehreren Quellen verschleiern und die Verwendung der Aktuatoren begrenzen kann.

PATRON [SDM+17] ist ein Privacy-System für das Internet der Dinge. Es ergänzt die Standardarchitektur für Anwendungen im Internet der Dinge um eine Verifikations- und Konfigurationsschicht und um eine Zugriffssteuerungsschicht. Dadurch können Muster im Datenstrom, die der Nutzer zuvor in natürlicher Sprache formuliert hat, verschleiert werden. Dabei wird sichergestellt, dass der Nutzen der Anwendung so wenig wie möglich beeinträchtigt wird [PDT+18].

Die PMP (Privacy Management Platform) [SM13] ist ein Berechtigungssystem für Mobilplattformen, das den Zugriff der Anwendungen auf die privaten Daten des Nutzers steuert. Dazu legt der Nutzer beim ersten Start einer Anwendung fest, welche Funktionen und Daten die Anwendung nutzen darf. Dabei ist es auch möglich, der Anwendung statt den korrekten Daten randomisierte Daten unterzuschieben. Der Nutzer wird dabei über die Konsequenzen seiner Entscheidungen informiert. Mit der PMP ist es jedoch nicht möglich, Datenströme, die aus mehreren Quellen kommen, zu reglementieren.

In MIALinx [WHS+16] können Regeln definiert werden, die beschreiben, wie ein Aktuator auf ein bestimmtes Muster in den Sensordaten reagieren soll. Ziel dieser Arbeit ist es, ein Privacy-System zu entwickeln, das PATRON mit der PMP kombiniert und in MIALinx integriert. Dieses System trägt den Namen SPYaware (Smart Home Privacy-aware Runtime Environment). Dazu wird ein Konzept erstellt und anschließend prototypisch implementiert, das PATRON als sichere Ausführungsumgebung für MIALinx nutzt und mithilfe des Berechtigungsrichtlinienmodells der PMP bzw. dessen Erweiterung ACCESSORS die Sensoren und Aktuatoren reglementiert. Die zentrale Konfiguration von PATRON und der PMP erfolgt analog zu dem Verteilungsmechanismus von AVARE [AOP+17].

1.2 Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Anwendungsszenarien und Anforderungen beschreibt Anwendungsszenarien aus dem Smart-Home-Umfeld und leitet daraus Anforderungen an das zu entwickelnde System ab.

Kapitel 3 – Verwandte Arbeiten stellt verwandte Privacy-Mechanismen für Smart Devices und Datenstromsysteme vor. Außerdem werden verwandte Systeme beschrieben, mit denen Einstellungen für verschiedene Geräte zentral verwaltet werden können.

Kapitel 4 – Konzept beschreibt den Aufbau und die Funktionsweise des Privacy-Systems SPYaware. Dazu wird zunächst der schematische Aufbau von SPYaware erklärt und diskutiert, in welchen Komponenten die Privacy-Kontrolle stattfinden soll. Anschließend werden die einzelnen Komponenten von SPYaware erläutert und es wird er-

klärt, wie die Privacy-Einstellungen auf die einzelnen Komponenten verteilt werden können.

Kapitel 5 – Grundlagen stellt existierende Systeme vor, die zur Implementierung von SPYaware verwendet werden.

Kapitel 6 – Implementierung zeigt, wie die in Kapitel 5 vorgestellten Systeme kombiniert werden können, um SPYaware zu realisieren.

Kapitel 7 – Prototyp beschreibt, wie der Prototyp realisiert wird. Dazu wird zunächst ein Anwendungsbeispiel vorgestellt, das in dem Prototyp umgesetzt werden soll. Danach wird auf die Umsetzung des Prototyps eingegangen.

Kapitel 8 – Evaluation bewertet die Ergebnisse des Prototyps. Dazu wird zunächst überprüft, inwieweit SPYaware die Anforderungen aus Kapitel 2 erfüllt. Danach geht es darum, wie gut die privaten Muster verschleiert werden können und inwieweit dabei die öffentlichen Muster erhalten bleiben.

Kapitel 9 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und bietet einen Ausblick auf zukünftige Arbeiten.

2 Anwendungsszenarien und Anforderungen

Zunächst werden in Abschnitt 2.1 Anwendungsszenarien im Bereich Smart Home beschrieben und die daraus resultierenden Bedrohungen für die Sicherheit und die Privacy des Nutzers aufgezeigt. Anhand dieser Szenarien werden in Abschnitt 2.2 die Anforderungen an das Datensicherheits- und Datenschutzkonzept abgeleitet. In Abschnitt 2.3 werden Schutzziele vorgestellt und überprüft, ob die Anforderungen sicherstellen, dass die Schutzziele eingehalten werden.

2.1 Anwendungsszenarien

Im Smart-Home-Umfeld gibt es eine Vielzahl an Anwendungsszenarien, bei denen die Gewährleistung der Privacy der Nutzer wichtig ist [AAL11]. In diesem Abschnitt werden zwei Anwendungsszenarien exemplarisch beschreiben.

Anwendungsszenario 1

Rechtzeitig bevor der Nutzer nach Hause kommt, soll die Heizung eingeschaltet werden, damit in der Wohnung eine angenehme Raumtemperatur herrscht. Kommt der Nutzer nach 17 Uhr nach Hause, soll der Backofen vorgewärmt werden, damit der Nutzer sich eine Pizza machen kann.

Dieses Szenario ist in Abbildung 2.1 dargestellt. Zunächst wird eine Möglichkeit benötigt, den Standort des Nutzers zu ermitteln, um daraus das Bewegungsmuster des Nutzers abzuleiten. Wie in Abbildung 2.1 zu sehen, kann dazu der GPS-Sensor des Autos oder der GPS-Sensor des Smartphones des Nutzers verwendet werden. Außerdem muss es möglich sein, die Heizung und den Backofen zu steuern. Dazu wird eine zentrale Steuerung benötigt, die auf die Positionsveränderung des Autos reagiert und die Heizung und den Backofen einschalten kann. Da der Backofen erst nach 17 Uhr eingeschaltet werden soll, muss außerdem die Uhrzeit bekannt sein.

Anwendungsszenario 2

Der Nutzer soll per SMS informiert werden, wenn ein Paket in der Paketbox vor dem Haus abgelegt wird. Die Benachrichtigung soll auch an das Multimediasystem im Auto des Nut-

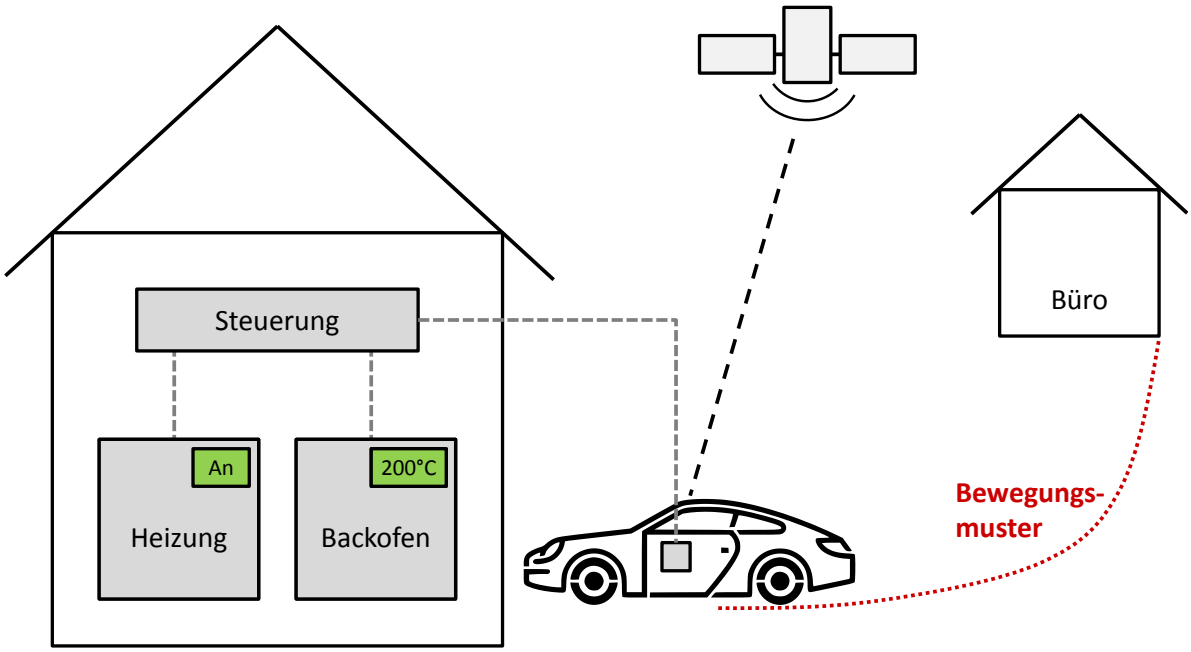

Abbildung 2.1: Anwendungsszenario 1

zers geschickt werden. Auf diese Weise kann der Nutzer auch während der Fahrt benachrichtigt werden, wenn er das Smartphone nicht bedienen darf.

Dieses Szenario ist in Abbildung 2.2 dargestellt. Die Paketbox vor dem Haus muss erkennen können, dass ein Paket abgelegt wurde. Außerdem muss sie über eine Internetanbindung verfügen, um das Ereignis den anderen Systemen mitteilen zu können. Die zentrale Steuerung, die alle ankommenden Sensordaten verarbeitet und die Aktuatoren steuert, muss über eine Funktion verfügen, die das Versenden von SMS-Nachrichten erlaubt. Der Empfang der SMS-Nachricht muss sowohl mit dem Smartphone des Nutzers als auch mit dem Multimediasystem im Auto möglich sein.

Aus diesen Szenarien ergeben sich verschiedene Bedrohungen für die Sicherheit und die Privacy des Nutzers:

1. Um zu erkennen, dass der Nutzer sich auf das Haus zubewegt, ist es notwendig, die aktuelle Position des Nutzers mithilfe des GPS-Sensors in seinem Smartphone oder mithilfe des GPS-Sensors des Autos zu ermitteln. Diese Informationen können missbraucht werden, um das Bewegungsprofil des Nutzers auszuspionieren. Das Bewegungsmuster kann beispielsweise für einen Supermarkt interessant sein, der mithilfe des Bewegungsprofils weiß, wann der Nutzer üblicherweise einkauft und so seine Werbung zielgerichteter personalisieren kann.

2. Das Bewegungsprofil kann außerdem in die Hände eines Einbrechers gelangen, der dadurch weiß, wann der Nutzer nicht zu Hause ist und abschätzen kann, wann der Nutzer wieder zu Hause sein wird. Erkennt der Einbrecher beispielsweise, dass der

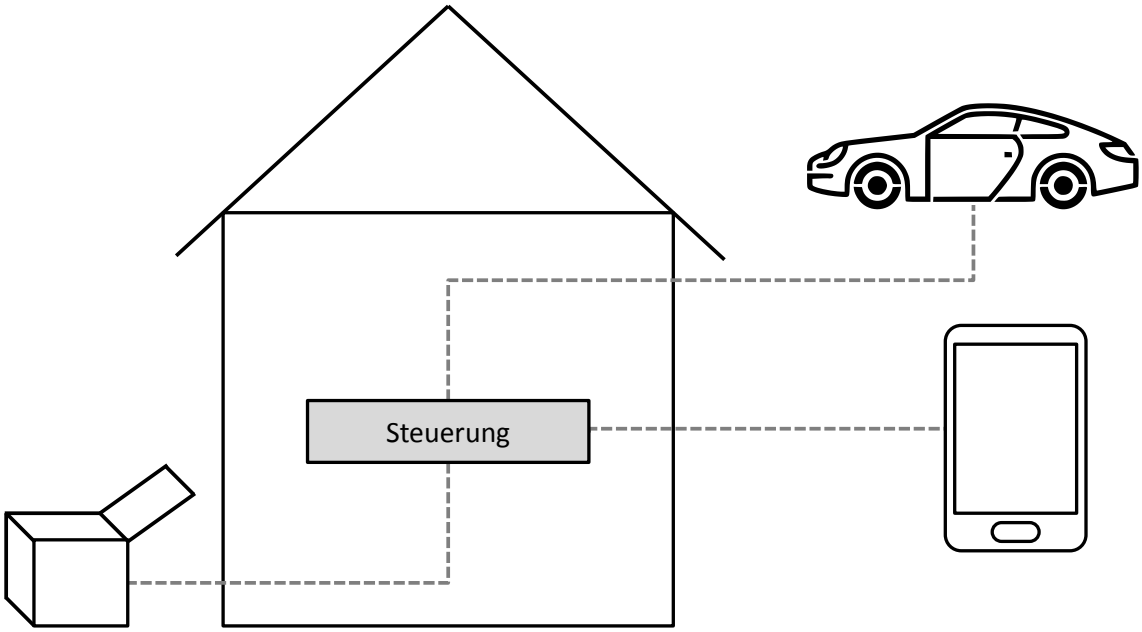

Abbildung 2.2: Anwendungsszenario 2

Nutzer sich auf dem Weg ins Kino befindet, weiß er, dass mit einer Rückkehr des Nutzers frühestens in zwei Stunden zu rechnen ist. Auf diese Weise kann sich der Einbrecher sicher sein, bei seinem Einbruch nicht erwischt zu werden.

3. Das Bewegungsprofil kann auch für die Versicherungsgesellschaft interessant sein. Anhand des Bewegungsprofils kann ermittelt werden, welche Straßen der Nutzer befahren hat und ob er dabei die zulässige Höchstgeschwindigkeit überschritten hat. Ein häufiges Überschreiten der zulässigen Höchstgeschwindigkeit könnte zur Erhöhung der Versicherungsbeiträge führen.

4. Ein weiterer Angriffspunkt ist das Datenstromsystem, das Sensordaten empfängt und entsprechend der empfangenen Daten die Aktuatoren steuert. Die Steuerung der Aktuatoren erfolgt über Regeln. Eine mögliche Regel wäre beispielsweise *„Wenn ein Paket in die Paketbox gelegt wird, informiere den Nutzer per SMS-Nachricht“*. Hat der Nutzer keine SMS-Flatrate gebucht, können durch das Versenden vieler SMS-Nachrichten hohe Kosten entstehen. Wird der Empfängerkreis der SMS-Nachricht nicht eingeschränkt, könnte die SMS-Nachricht außerdem an einen Dieb geschickt werden, der das Paket aus der Paketbox entwenden will.

2.2 Anforderungen

Aus den im vorherigen Abschnitt beschriebenen Anwendungsszenarien und den beschriebenen Bedrohungen lassen sich nachfolgende Anforderungen ableiten. Die Anforderungen

werden in die Gruppen *Regelverarbeitung*, *Musterverschleierung*, *Reglementierung der Sensoren und Aktuatoren* und *Konfiguration* eingeteilt.

Regelverarbeitung

A1 Wenn-Dann-Regeln

Im Smart Home befinden sich verschiedene Sensoren und Aktuatoren. Die Steuerung der Aktuatoren erfolgt in Abhängigkeit von den Daten, die die Sensoren liefern. Es muss also möglich sein, Wenn-Dann-Regeln zu definieren, die beschreiben, wie auf ein bestimmtes Muster in den Sensordaten reagiert werden soll. Eine Regel könnte beispielsweise lauten: *„Wenn sich der Nutzer auf das Haus zubewegt, dann schalte die Heizung ein“*. Das Muster *„Nutzer bewegt sich auf das Haus zu“* gehört in diesem Beispiel zum Wenn-Bestandteil der Regel und wird mithilfe der Daten des GPS-Sensors erkannt. Die Aktion *„Heizung einschalten“* gehört zum Dann-Bestandteil der Regel und steuert die Heizung.

A2 Echtzeitverarbeitung

Die Sensordaten kommen als Datenströme aus mehreren Quellen wie beispielsweise dem GPS-Sensor des Autos oder dem GPS-Sensor des Smartphones an. Da sich die Position des Nutzers jederzeit ändern kann, ist es notwendig, innerhalb einer bestimmten Zeitspanne darauf zu reagieren. Kommt der Nutzer beispielsweise aufgrund einer Umleitung erst später nach Hause, darf die Heizung erst später eingeschaltet werden. Um auf solche Ereignisse reagieren zu können, ist es notwendig, die Sensordaten in Echtzeit zu verarbeiten.

Musterverschleierung

A3 Öffentliche Muster

Die Muster, die im Wenn-Bestandteil der Wenn-Dann-Regeln definiert werden, werden als öffentliche Muster bezeichnet. Um zu wissen, wann der Dann-Bestandteil der Wenn-Dann-Regeln ausgeführt werden soll, muss es möglich sein, die öffentlichen Muster im Datenstrom zu erkennen. In Anwendungsszenario 1 soll beispielsweise das Muster *„Nutzer bewegt sich auf das Haus zu“* in den Daten des GPS-Sensors erkannt werden.

A4 Private Muster

Private Muster sind Muster im Datenstrom, die nicht preisgegeben werden dürfen, da sonst die Privacy des Nutzers verletzt wird. Dazu zählt in Anwendungsszenario 1 beispielsweise das Muster *„Nutzer fährt zum Supermarkt“*. Diese privaten Muster müssen aus dem Datenstrom entfernt bzw. verschleiert werden. Die Verschleierung der Muster muss außerdem domänenspezifisch sein. So soll die Information, dass der Nutzer nach Hause kommt, beispielsweise nur für die Heizung und den Backofen verfügbar sein. Alle anderen Geräte dürfen keinen Zugriff auf diese Information erhalten.

A5 Muster erhalten

Die öffentlichen Muster werden benötigt, um festzustellen, wann die Bedingung im Wenn-Bestandteil einer Regel zutrifft. Ohne die öffentlichen Muster ist die Steuerung der Aktuatoren nicht möglich. Deshalb dürfen durch das Verschleiern der privaten Muster die öffentlichen Muster nicht verloren gehen. Beispielsweise darf durch das Verschleiern des Musters *„Nutzer fährt zum Supermarkt“* das Muster *„Nutzer bewegt sich auf das Haus zu“* nicht zerstört werden.

A6 Keine False Positives

Treten durch das Verschleiern der privaten Muster neue öffentliche Muster im Datenstrom auf, die zuvor nicht enthalten waren, spricht man von False Positives. Treten False Positives auf, hat das Einfluss auf die Auswertung der Bedingungen im Wenn-Bestandteil der Wenn-Dann-Regeln. Das kann dazu führen, dass die Aktion, die im Dann-Bestandteil der Regel festgelegt ist, fälschlicherweise ausgeführt wird. Deshalb dürfen durch die Verschleierung der privaten Muster keine False Positives entstehen. Beispielsweise darf durch das Verschleiern des privaten Musters *„Nutzer fährt zum Supermarkt“* nicht das öffentliche Muster *„Nutzer bewegt sich auf das Haus zu“* im Datenstrom auftreten, sofern es zuvor nicht schon enthalten war.

Reglementierung der Sensoren und Aktuatoren

A7 Sensoren reglementieren

Die genaue Fahrtroute und die genaue Geschwindigkeit des Nutzers darf nicht erkannt werden, da ansonsten das genaue Bewegungsmuster des Nutzers ausspioniert werden kann. Wie in Abschnitt 2.1 beschrieben, kann das Bewegungsprofil für die Versicherung oder für einen Einbrecher interessant sein. Daher darf es beispielsweise nicht erkennbar sein, dass der Nutzer Anliegerstraßen nutzt oder die Geschwindigkeitsbegrenzung einer bestimmten Straße überschreitet. Um das zu erreichen, muss es möglich sein, die Weitergabe der Sensordaten zu reglementieren.

A8 Aktuatoren reglementieren

Neben der Reglementierung der Sensoren muss es auch möglich sein, Aktuatoren zu reglementieren. Um in Anwendungsszenario 2 hohe Kosten durch das Versenden der SMS-Nachrichten zu vermeiden, muss es möglich sein, das Senden von SMS-Nachrichten zu beschränken. Beispielsweise könnte festgelegt werden, dass maximal zehn SMS-Nachrichten pro Tag verschickt werden dürfen.

A9 Feingranularität

Die vollständige Blockierung der Weitergabe der Sensordaten ist oft nicht sinnvoll, da die Sensordaten für die Auswertung der Wenn-Dann-Regeln benötigt werden. Beispielsweise werden die Daten des GPS-Sensors benötigt, um das Muster *„Nutzer bewegt sich auf das Haus zu“* zu erkennen. Anstatt die Sensordaten vollständig zu blockieren, wird eine feingranulare Reglementierung der Sensorda-

ten benötigt, die es beispielsweise erlaubt, die Präzision der Sensordaten zu reduzieren. In Anwendungsszenario 1 reicht es aus, die Position auf 100 Meter genau zu kennen, um festzustellen, dass der Nutzer sich auf das Haus zubewegt. Durch die Reduzierung der Präzision der Sensordaten kann das genaue Bewegungsprofil verschleiert werden, ohne die Erkennung der Muster zu beeinträchtigen.

A10 Kontextsensitivität

Welche Daten weitergegeben werden, kann sich von Situation zu Situation unterscheiden. Im Normalfall soll die Versicherung nicht erfahren, dass der Nutzer zu schnell gefahren ist. Kommt es jedoch zu einem Unfall, darf die zu hohe Geschwindigkeit nicht abgestritten werden können. Die Reglementierung der Sensoren und Aktuatoren muss daher kontextsensitiv sein.

A11 Erweiterbarkeit

Ein Smart Home besteht aus vielen verschiedenen Sensoren und Aktuatoren. Durch den Kauf neuer Produkte kann das Smart Home um neue Sensoren oder Aktuatoren erweitert werden. Beispielsweise könnte sich der Nutzer einen intelligenten Kühlschrank kaufen und in das bestehende System einbinden wollen. Dafür ist es notwendig, dass sich neue Sensoren bzw. Aktuatoren einfach in das bestehende System integrieren lassen.

A12 Datenzentriertheit

Da die meisten Nutzer keine Erfahrung mit der Definition von Privacy-Regeln haben, ist es wichtig, dass die Konfiguration von SPYaware möglichst einfach ist. Die Nutzer wissen zwar, welche Daten sie geheim halten möchten, jedoch nicht, welche Sensoren diese Daten erfassen. Beispielsweise weiß der Nutzer nicht, welcher Sensor genutzt wird, um das Muster *„Nutzer bewegt sich auf das Haus zu“* zu erkennen. Anstatt einzelne Sensoren zu reglementieren, sollte der Nutzer den Zugriff auf bestimmte Daten, wie beispielsweise die Standortdaten, einschränken können. Die Reglementierung muss daher datenzentriert erfolgen.

A13 Übertragungsfrequenz

Werden in Anwendungsszenario 1 die Standortdaten in einem kurzen Zeitraum sehr häufig abgefragt, kann daraus der exakte Standort berechnet werden. Dazu wird angenommen, dass die unpräzisen Standortdaten einem zufälligen Standort innerhalb eines 100-Meter-Radius um den exakten Standort entsprechen. Berechnet man nun den Mittelwert der unpräzisen Standortdaten, erhält man eine sehr gute Abschätzung des exakten Standorts. Um dies zu verhindern, dürfen die Standortdaten beispielsweise nur alle 30 Sekunden abgefragt werden. Es muss also möglich sein, die Übertragungsfrequenz der Sensordaten einzuschränken.

Konfiguration

A14 Zentrale Konfiguration

Die Sensoren und Aktuatoren in den vorgestellten Anwendungsszenarien befinden sich in einem heterogenen Umfeld. Die Sensoren und Aktuatoren können sich so-

wohl im Smart Home, als auch in einem Auto befinden. Um die Privacy des Nutzers sicherzustellen, muss die Konfiguration aller Sensoren und Aktuatoren aufeinander abgestimmt sein. Daher sollte die Konfiguration der Privacy-Einstellungen an einer zentralen Stelle erfolgen und von dort auf die verschiedenen Smart Devices ausgeliefert werden.

2.3 Einhaltung der Schutzziele

Um die Datensicherheit des Konzepts zu gewährleisten, ist es notwendig, dass das Konzept bestimmte Schutzziele einhält. In diesem Abschnitt werden Schutzziele vorgestellt und dargelegt, inwiefern die Anforderungen sicherstellen, dass die Schutzziele eingehalten werden.

Die ISO 27001 [ISO13] definiert drei Schutzziele:

- **Vertraulichkeit**
  Es muss sichergestellt werden, dass nicht autorisierte Personen oder Systeme keinen Zugriff auf die Daten erhalten.

- **Integrität**
  Es muss sichergestellt werden, dass die Daten von Dritten nicht verändert werden.

- **Verfügbarkeit**
  Es muss sichergestellt werden, dass der Zugriff auf die Daten autorisierten Personen oder Systemen innerhalb eines bestimmten Zeitrahmens möglich ist.

Die drei Schutzziele der ISO 27001 [ISO13] werden in der Literatur um weitere Ziele ergänzt. Eckert [Eck13] nennt folgendes weiteres Ziel:

- **Nichtabstreitbarkeit**
  Es muss sichergestellt werden, dass ein System, das eine bestimmte Aktion durchgeführt hat, dies nicht abstreiten kann.

Pohl [Poh04] sieht die folgenden Ziele als Unterziele der Integrität:

- **Konsistenz**
  Es muss sichergestellt werden, dass die verarbeiteten Werte mit den tatsächlichen Werten übereinstimmen.

- **Genauigkeit**
  Es muss sichergestellt werden, dass die Daten in der benötigten Genauigkeit vorliegen.

- **Korrektheit**
  Es muss sichergestellt werden, dass die Daten fehlerfrei sind.

- **Vollständigkeit**
  Es muss sichergestellt werden, dass keine Daten fehlen.

Um die Vertraulichkeit zu gewährleisten, muss sichergestellt werden, dass nur die öffentlichen Muster preisgegeben werden. Um das zu erreichen, stellt Anforderung A4 (Private Muster) sicher, dass alle privaten Muster verschleiert werden. Außerdem wird durch Anforderung A7 (Sensoren reglementieren) und A9 (Feingranularität) die Präzision der Sensordaten reduziert, wodurch private Informationen wie beispielsweise die exakte Position des Nutzers verschleiert werden.

Durch Anforderung A2 (Echtzeitverarbeitung) wird sichergestellt, dass ankommende Sensordaten innerhalb einer bestimmten Zeitspanne verarbeitet werden. Dadurch wird die Verfügbarkeit gewährleistet.

Die Nichtabstreitbarkeit ist dann gewährleistet, wenn sämtliche öffentlichen Muster erkannt werden, alle privaten Muster verschleiert werden und keine False Positives im Datenstrom auftreten. Dazu müssen die Anforderungen A3 (Öffentliche Muster), A4 (Private Muster), A5 (Muster erhalten) und A6 (Keine False Positives) erfüllt sein.

Anforderung A6 (Keine False Positives) verhindert, dass durch das Verschleiern neue Muster im Datenstrom auftreten, die zuvor nicht in dem Datenstrom enthalten waren, und gewährleistet so die Konsistenz. Die Genauigkeit wird durch Anforderung A9 (Feingranularität) sichergestellt, da durch Anforderung A9 gewährleistet ist, dass die Präzision der Sensordaten nur so weit reduziert wird, dass die Erkennung der öffentlichen Muster noch möglich ist. Die Korrektheit ist dann gegeben, wenn nach der Verschleierung keine privaten Muster mehr im Datenstrom vorhanden sind und keine False Positives auftreten. Das wird durch Anforderung A4 (Private Muster) und Anforderung A6 (Keine False Positives) gewährleistet. Die Vollständigkeit ist erfüllt, wenn nach dem Verschleiern der privaten Muster noch alle öffentlichen Muster im Datenstrom enthalten sind. Dies wird durch Anforderung A5 (Muster erhalten) sichergestellt.

3 Verwandte Arbeiten

In diesem Abschnitt werden verwandte Ansätze vorgestellt und bewertet. THOR [SSM18] ist ein Konzept für die Sicherstellung der Privacy der Nutzer für die Industrie 4.0. In der Industrie 4.0 werden die am Produktionsprozess beteiligte Systeme mit Sensoren ausgestattet und miteinander vernetzt, um mit den erfassten Daten den Produktionsprozess zu optimieren und die Produktion weitestgehend selbstständig zu organisieren. Ein möglicher Anwendungsfall wäre es, in einer Smart Factory anhand der erhobenen Sensordaten eine Abweichung selbstständig festzustellen und im Fehlerfall das notwendige Ersatzteil zu bestellen und einen Servicetechniker zu informieren, der das Ersatzteil einbauen kann. Sind mehrere Servicetechniker verfügbar, soll der Servicetechniker gerufen werden, der sich am nächsten an der Maschine befindet und gerade nicht mit anderen Arbeiten beschäftigt ist. Dazu ist es jedoch notwendig, den Standort der Servicetechniker ermitteln zu können und Zugriff auf den Kalender der Servicetechniker zu erhalten. Dadurch wird der Mensch zu einem relevanten Bestandteil des Produktionsprozesses, wodurch es notwendig wird, die persönlichen Daten vor unberechtigtem Zugriff zu schützen [SSM18].

Um die persönlichen Daten des Nutzers schützen und dabei die Datenanalyse so wenig wie möglich zu beeinträchtigen, kombiniert THOR die PMP und PATRON miteinander. Die PMP wird genutzt, um die Sensoren zu reglementieren. PATRON ermöglicht es, private Muster in den Daten zu verschleiern. Beide Systeme werden in Abschnitt 5.2 bzw. Abschnitt 5.3 ausführlicher erklärt. Bevor THOR eingesetzt werden kann, müssen zunächst die Privacy-Einstellungen festgelegt werden. Dazu beschreibt der Datenschutzbeauftragte bestimmte Schutzziele. Aus diesen Schutzzielen leitet ein Domänenexperte die Privacy-Regeln für die PMP und die privaten Muster für PATRON ab. Ein Datenanalyst formuliert Analyseziele, aus denen ein Domänenexperte die öffentlichen Muster folgert, die für PATRON und MIALinx benötigt werden [SSM18].

Zur Laufzeit reglementiert die PMP die Sensoren und stellt so sicher, dass nur die erwünschten Daten zum Datenstromsystem gelangen. PATRON untersucht die ankommenden Datenströme aller Sensoren auf private Muster. Tritt ein privates Muster auf, wird es in ein neutrales Muster umgewandelt. Ein neutrales Muster ist ein Muster, das keine Auswirkung auf die spätere Verarbeitung hat. Die von PATRON bereinigten Datenströme dienen anschließend als Eingabe für MIALinx. MIALinx erkennt die öffentlichen Muster und löst entsprechend der Wenn-Dann-Regeln die passenden Aktionen aus. Da PATRON alle Ausgaben von MIALinx mitlesen kann, kann nun verifiziert werden, ob alle relevanten Ereignisse erkannt wurden und ob alle privaten Muster erfolgreich verschleiert wurden [SSM18].

THOR ist ein Konzept, das für den Einsatz in einer Smart Factory konzipiert wurde. Für den Einsatz in einem Smart Home gibt es jedoch einige Anforderungen, die von THOR nicht

abgedeckt werden. THOR ermöglicht zwar die Reglementierung der Sensoren, die Reglementierung der Aktuatoren ist jedoch nicht vorgesehen. Im Kontext eines Smart Homes ist dies jedoch notwendig, um beispielsweise die Anzahl der SMS, die pro Tag versendet werden dürfen, einzuschränken, um so die Kosten für den Nutzer zu begrenzen.

Im Gegensatz zu einer Smart Factory sind die Nutzer eines Smart Homes keine Fachleute. Insbesondere sind in diesem Umfeld keine Datenanalysten und Datenschutzbeauftragte vorhanden, die sich mit der Definition von Analysezielen und Datenschutzzielen auskennen. Es gibt auch keine Domänenexperten, die die definierten Ziele in Muster überführen können. Daher muss die Definition der Wenn-Dann-Regeln und der Privacy-Einstellungen besonders einfach sein und soweit wie möglich softwareseitig unterstützt werden. Da die Nutzer eines Smart Homes über unterschiedliche Endgeräte verfügen können, muss die Konfiguration unabhängig von Gerätetyp und Betriebssystem möglich sein. Die Konfiguration muss auch von unterwegs aus möglich sein, da die Nutzer sich auch außerhalb des Smart Homes befinden können.

Da der Nutzer nicht den ganzen Tag zuhause ist, müssen zur Steuerung der Aktuatoren des Smart Homes neben den Sensoren im Smart Home auch Sensoren außerhalb des Smart Homes verwendet werden. Um beispielsweise zu erkennen, dass der Nutzer nach Hause kommt, können die Sensoren des Smartphones oder des Autos verwendet werden. Um die persönlichen Informationen des Nutzers zu schützen, wird in diesem Fall jedoch ein heterogenes und verteiltes Privacy-System benötigt, das an einer zentralen Stelle konfiguriert werden kann und Geräte aus verschiedenen Bereichen mit einbezieht.

Im folgenden Abschnitt 3.1 werden Privacy-Mechanismen vorgestellt, mit denen Smart Devices reglementiert werden können. Dies ist notwendig, um Sensordaten schon vor der Weitergabe an das Datenstromsystem zu verschleiern und die Verwendung der Aktuatoren einzuschränken. Danach geht es in Abschnitt 3.2 um Privacy-Mechanismen, die zum Schutz der Daten in Datenstromsystemen eingesetzt werden. Abschnitt 3.3 beschreibt schließlich, wie Einstellungen, die über verschiedene Geräte verteilt sind, zentral verwaltet werden können.

3.1 Privacy-Mechanismen für Smart Devices

Die hier vorgestellten Privacy-Mechanismen für Smart Devices lassen sich in zwei Kategorien einteilen. Der erste Ansatz besteht darin, das Berechtigungssystem von Android zu erweitern. Vertreter des ersten Ansatzes sind Apex [NKZ10], AppFence [HHJ+11], Constroid [SPH11], CRêPE [CCF+12], MockDroid [BRS+11], SEAF [BAK+11], Sorbet [FBJ+12] und YAASE [RCF+11]. Der zweite Ansatz erweitert die Apps selbst um eine zusätzliche Überwachungskomponente, die die Einhaltung der Privacy-Regeln sicherstellt. AFP [SMA+17], AppGuard [BGH+14], Aurasium [XSA12], Dr. Android and Mr. Hide [JMV+12] und RetroSkeleton [DC13] sind Vertreter des zweiten Ansatzes. Im Folgenden werden für jeden Ansatz zwei Privacy-Mechanismen exemplarisch erklärt. Alle vier Ansätze

haben gemeinsam, dass sie auf Android basieren. Außerdem wird mit der PMP [Sta13] ein alternatives Berechtigungssystem für Android vorgestellt, das sowohl die Anpassung der Apps als auch die Veränderung des Betriebssystems erfordert. ACCESSORS [SM18] ist eine Erweiterung des Berechtigungsrichtlinienmodells der PMP und erlaubt es, den Zugriff auf bestimmte Daten unabhängig davon, von welchen Sensoren die Daten stammen, zu reglementieren. Die vorgestellten Privacy-Mechanismen werden in Tabelle 3.1 miteinander verglichen.

Ein Vertreter des ersten Ansatzes ist **Apex** [NKZ10]. Mit Apex ist es möglich, restriktionsbasierte Regeln zu erstellen. Eine solche Regel könnte beispielsweise die Anzahl der SMS, die pro Tag verschickt werden dürfen, einschränken. Neben der Definition solcher Bedingungen ist es auch möglich, einer App lediglich eine Teilmenge der angeforderten Berechtigungen zu genehmigen. Das Hinzufügen und Entziehen der Berechtigungen ist nicht nur bei der Installation der App, sondern auch zur Laufzeit möglich. Nutzt eine App jedoch eine Funktion, für die Berechtigungen benötigt werden, die zuvor verweigert wurden, stürzt die App mit einer Fehlermeldung ab. Außerdem werden für Apex Root-Rechte benötigt, da Apex das Android-Betriebssystem verändert [Sta13].

Apex erlaubt es, einer App einzelne Berechtigungen zu verweigern. Es ist jedoch nicht möglich, feingranulare Bedingungen zu formulieren. Beispielsweise kann der Zugriff auf die Standortdaten erlaubt oder verweigert werden. Es ist jedoch nicht möglich, die Genauigkeit der Standortdaten einzuschränken. Zudem können die Privacy-Regeln nicht an bestimmte Situationen wie beispielsweise die Tageszeit geknüpft werden. Da Apex auf den in Android definierten Berechtigungen basiert, kann es zudem nicht erweitert werden [SM18].

Der Ansatz von **MockDroid** [BRS+11] besteht wie bei Apex darin, das Berechtigungssystem von Android zu erweitern. Anstatt jedoch einer App den Zugriff auf die Daten zu verweigern, erhält die App stattdessen falsche Daten. Fordert eine App bei der Installation bestimmte Berechtigungen an, werden diese zunächst erteilt. Der Nutzer kann dann zur Laufzeit festlegen, ob die App die korrekten Daten oder falsche Daten erhält. Fordert eine App beispielsweise Zugriff auf den Kalender oder die Kontakte an, wird vorgetäuscht, dass sich im Kalender bzw. in der Kontaktliste keine Einträge befinden. Möchte die App einen neuen Kalendereintrag oder einen neuen Kontakt anlegen, simuliert MockDroid ein volles Dateisystem, das keine weiteren Daten aufnehmen kann. Auch der Zugriff auf die Standortdaten und das Internet lässt sich mit MockDroid reglementieren. Anstatt die tatsächlichen Standortdaten bereitzustellen, erhält die App den Hinweis, dass keine Standortinformationen verfügbar sind. Versuche auf das Internet zuzugreifen führen zu einer Zeitüberschreitung.

MockDroid macht sich dabei zunutze, dass die meisten Apps so entwickelt werden, dass sie mit der Nichtverfügbarkeit von Ressourcen umgehen können. Auf diese Weise ist es möglich, eine App zu installieren und ohne Absturz zu nutzen, ohne der App Zugriff auf sensible Daten geben zu müssen. Diese Vorgehensweise ist insbesondere dann sinnvoll, wenn eine App optionale Funktionen besitzt, die der Nutzer nicht verwenden möchte. Möchte der Nutzer beispielsweise nur eine bestimmte Adresse suchen und auf der Karte anzeigen, muss der App dafür der Standort des Nutzers nicht bekannt sein. Erst für die Navigation zu dieser

Adresse werden die Standortinformationen benötigt. MockDroid unterstützt jedoch nur eine Teilmenge der in Android verfügbaren Berechtigungen [BRS+11]. Der Zugriff auf die Kamera oder das Mikrofon lassen sich mit MockDroid nicht reglementieren. MockDroid ist daher nicht erweiterbar. Zudem können die Bedingungen nicht an bestimmte Situationen wie beispielsweise die Tageszeit geknüpft werden. Da es nicht möglich ist, feingranulare Bedingungen zu formulieren, verlieren einige Apps ihren Sinn. Eine App, die beispielsweise alle Tankstellen in der näheren Umgebung anzeigen soll, kann ohne Standortdaten nicht arbeiten. Wäre es möglich, die Berechtigungen feingranularer einzustellen, also der App den Standort zumindest auf 100 Meter genau mitzuteilen, würde das für das Anzeigen der Tankstellen in der näheren Umgebung ausreichen.

Ein Vertreter des zweiten Ansatzes ist **AppGuard** [BGH+14]. AppGuard verändert den Bytecode der Apps und erweitert sie so um die vorgegebenen Privacy-Regeln und eine Überwachungskomponente, die die Einhaltung der Privacy-Regeln sicherstellt. Dabei werden nur die Apps selbst verändert. Änderungen am Android-Betriebssystem sind nicht notwendig. Daher werden auch keine Root-Rechte benötigt. Durch die Änderung der Apps werden alle Methodenaufrufe an die eingebaute Überwachungskomponente umgeleitet. Anschließend prüft die Überwachungskomponente, ob der Aufruf zulässig ist. Der Nutzer kann bei der Installation oder zur Laufzeit entscheiden, auf welche Daten die App zugreifen darf und welche Funktionen die App nutzen darf.

Anstatt den Zugriff generell zu erlauben oder zu verweigern, ist es auch möglich, feingranulare Regeln zu definieren. Beispielsweise kann der Internetzugriff auf eine bestimmte Adresse beschränkt werden. Die korrekten Daten können verfälscht oder durch zufällig generierte Daten ersetzt werden. Es ist auch möglich, Regeln an bestimmte Kontexte wie den Standort zu knüpfen. AppGuard kann um weitere Datenquellen erweitert werden, da grundsätzlich jeder beliebige Befehl im Bytecode reglementiert werden kann [BGH+14]. Problematisch bei diesem Ansatz ist, dass die Veränderung des Bytecodes der App gegen das Urheberrecht des Entwicklers verstößt und daher unzulässig ist [APW17].

Ein weiterer Vertreter des zweiten Ansatzes ist **RetroSkeleton** [DC13]. RetroSkeleton ist ein System, mit dem das Verhalten einer beliebigen App verändert werden kann, ohne Zugriff auf den Quellcode der App zu haben. Dazu werden bestimmte Methodenaufrufe in der App abgefangen und durch den Aufruf einer anderen Methode ersetzt. Wie in Abbildung 3.1 zu sehen, definiert ein Richtlinienersteller zunächst eine Transformationsrichtlinie. Eine Transformationsrichtlinie enthält eine Beschreibung der alten Methoden, die ersetzt werden sollen und den Quellcode der neuen Methoden, die stattdessen aufgerufen werden sollen. RetroSkeleton analysiert die ursprüngliche App und identifiziert die Methodenaufrufe, die gemäß der Transformationsrichtlinie ersetzt werden müssen. Diese Methodenaufrufe werden durch Aufrufe an die in der Transformationsrichtlinie festgelegten Methoden ersetzt. Der Rest der App bleibt unverändert. Zusätzlich wird der Quellcode der in der Transformationsrichtlinie enthaltenen Methoden kompiliert. Am Ende werden die kompilierten Methoden, die Aufrufe dieser Methoden und die Bestandteile der ursprünglichen App zu einer neuen App zusammengefügt.

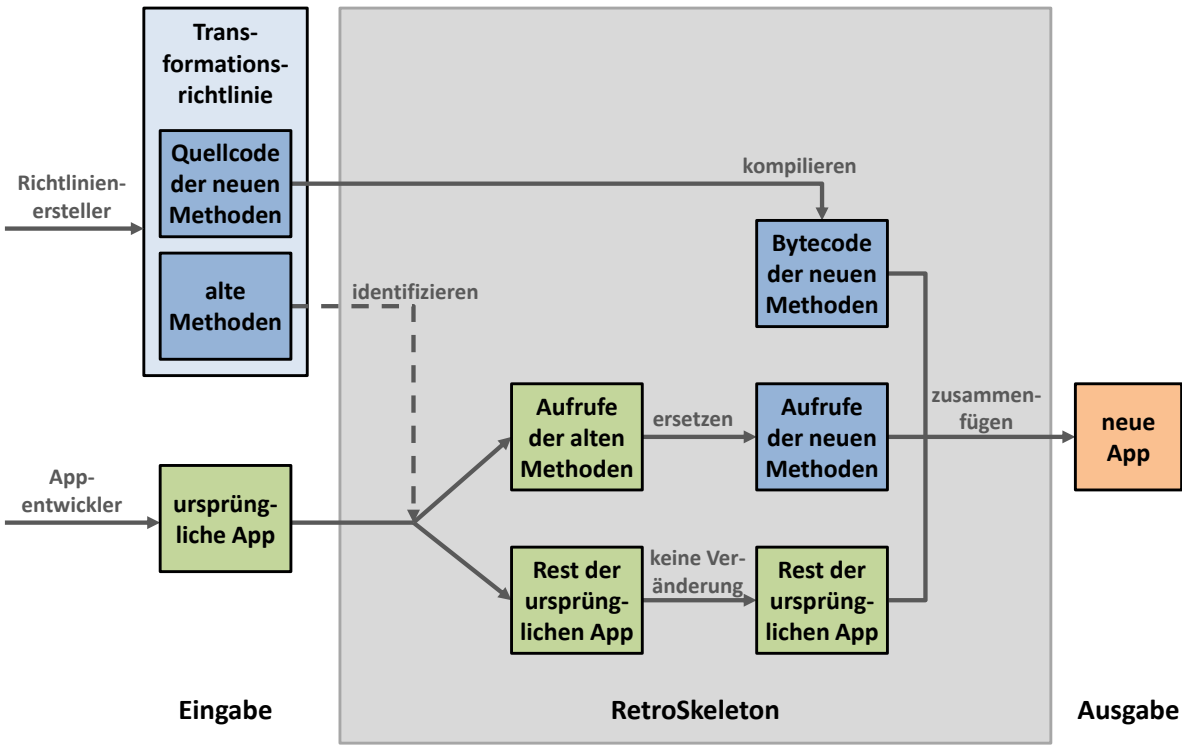

Abbildung 3.1: Systemdiagramm von RetroSkeleton, in Anlehnung an [DC13]

Die so erzeugte App kann auf jedem Android-Gerät ausgeführt werden. Eine Veränderung des Android-Betriebssystems oder Root-Rechte sind nicht notwendig. Im Gegensatz zu AppGuard wird die App nicht um eine Überwachungskomponente ergänzt, die die Einhaltung der Privacy-Regeln sicherstellt. Stattdessen werden beliebige Methodenaufrufe durch Aufrufe anderer Methoden ersetzt. Auch auf diese Weise kann die Einhaltung der Privacy-Regeln sichergestellt werden. Beispielsweise kann der Aufruf der Standortdaten angefangen und die Rückgabe verändert werden. Anstatt den genauen Standort zurückzugeben, kann der Standort auch nur auf 100 Meter genau zurückgegeben werden. Auf diese Weise können feingranulare und kontextsensitive Privacy-Regeln definiert werden. Da die so definierten Privacy-Regeln nicht von den bestehenden Berechtigungen in Android abhängen, sind die Privacy-Regeln beliebig erweiterbar. Allerdings erfordern das Identifizieren der zu ersetzenden Methodenaufrufe und das Programmieren der Methoden, die stattdessen aufgerufen werden sollen, Programmierkenntnisse. Zudem verstößt die Veränderung der ursprünglichen App ohne das Einverständnis des Entwicklers wie bei AppGuard gegen das Urheberrecht und ist daher nicht zulässig [APW17].

Die **PMP** (Privacy Management Platform) [Sta13] ist ein alternatives Berechtigungssystem für Android. Im Berechtigungsrichtlinienmodell der PMP besteht eine App aus Service Features. Ein Service Feature kapselt eine bestimmte Funktion der App. Ein Beispiel für ein Service Feature ist die Navigation zu einer bestimmten Adresse in einer Navigations-App. Eine Ressource ist einer Schnittstelle zu geschützten Daten oder Systemfunktionen. Ressourcen sind beispielsweise der GPS-Sensor oder die Kamera des Smartphones. Mithilfe

Anforderung	Apex	Mock Droid	App Guard	Retro Skeleton	PMP	ACCES-SORS
Sensoren reglementieren	✓	✓	✓	✓	✓	✓
Aktuatoren reglementieren	✓	✗	✓	✓	✓	✓
Feingranularität	✗	✗	✓	✓	✓	✓
Kontextsensitivität	✗	✗	✓	✓	✓	✓
Erweiterbarkeit	✗	✗	✓	✓	✓	✓
Datenzentriertheit	✗	✗	✗	✗	✗	✓

Tabelle 3.1: Evaluation der Privacy-Mechanismen für Smart Devices

von Privacy-Regeln kann der Nutzer festlegen, auf welche Ressourcen ein Service Feature zugreifen darf. Beispielsweise kann festgelegt werden, dass das Service Feature, das den Nutzer zu einer bestimmten Adresse navigiert, auf die Daten des GPS-Sensors zugreifen darf. Wird der Zugriff für das Service Feature verweigert, wird das betroffene Service Feature deaktiviert. Die anderen Service Features der App sind davon nicht betroffen. So ist es weiterhin möglich, die Position einer bestimmten Adresse auf einer Karte anzuzeigen. Mithilfe feingranularer Privacy-Regeln kann zudem festlegt werden, dass der Standort des GPS-Sensors nur auf 100 Meter genau weitergegeben wird. Es kann außerdem festgelegt werden, dass eine Privacy-Regel nur in einem bestimmten Kontext gilt. Beispielsweise kann eine Privacy-Regel nur an einem bestimmten Ort gelten. In Abschnitt 5.3 wird die PMP ausführlicher erklärt.

ACCESSORS [SM18] ist eine Erweiterung des Berechtigungsrichtlinienmodels der PMP, das den Zugriff auf bestimmte Daten unabhängig davon, von welchen Sensoren die Daten stammen, reglementiert. Dazu muss ACCESSORS wissen, welche Sensoren welche Arten von Daten erzeugen. Dadurch eignet sich ACCESSORS nicht nur für den Einsatz auf einem Smartphone, sondern auch für den Einsatz in einem Smart Home. Genau wie mit dem Berechtigungsrichtlinienmodell der PMP ist es möglich, feingranulare und kontextsensitive Privacy-Regeln zu definieren. Jedoch kann das Objekt, das Zugriff auf die Daten anfordert, sowohl ein Smart Device als auch eine andere App sein. Als Datenquellen kommen andere Apps, Datenbanken oder Sensoren infrage. Aus diesen Datenquellen werden Informationen wie beispielsweise der Standort abgeleitet. Eine Privacy-Regel legt fest, unter welchen Bedingungen ein anforderndes Objekt auf diese Informationen zugreifen darf. ACCESSORS wird in Abschnitt 5.4 ausführlich beschrieben.

Die vorgestellten Privacy-Mechanismen wurden alle entwickelt, um die Berechtigungen auf einem Smartphone zu reglementieren. In einem Smart Home müssen jedoch eine Vielzahl von Sensoren und Aktuatoren reglementiert werden. Da die Sensoren und Aktuatoren zusammen eine vernetzte Infrastruktur bilden, reicht die Reglementierung einzelner Sensoren nicht aus. Zudem macht es oft keinen Sinn, den Zugriff auf einen Sensorwert komplett zu unterbinden, da die Anwendung dann den erwünschten Nutzen nicht mehr erbringen kann. Stattdessen sind feingranulare Regeln notwendig, durch die beispielsweise die Standortdaten nur auf 100 Meter genau weitergegeben werden. Die Regeln sollten zudem kontextsensitiv sein, also beispielsweise nur zu einer bestimmten Tageszeit gelten. Da in einem Smart Home jederzeit neue Sensoren oder Datenquellen hinzukommen können, muss ein Privacy-Mechanismus, der die Datenquellen reglementiert, dynamisch erweiterbar sein.

Wie in Tabelle 3.1 zu sehen, erfüllt nur ACCESSORS alle Anforderungen. Mit Apex und MockDroid ist es nicht möglich, feingranulare und kontextsensitive Privacy-Regeln zu definieren. Außerdem sind diese Systeme nicht erweiterbar. Im Gegensatz dazu ist es mit AppGuard und RetroSkeleton möglich, feingranulare und kontextsensitive Regeln zu definieren. Allerdings beruht das Prinzip beider Ansätze darauf, Methodenaufrufe innerhalb der App abzufangen und an eine Überwachungskomponente oder eine selbst implementierte Methode umzuleiten. Dieses Prinzip ist für Sensoren und Aktuatoren in einem Smart Home jedoch nicht geeignet. Somit kommen nur die PMP und deren Erweiterung ACCESSORS für die Reglementierung der Sensoren und Aktuatoren infrage. Da außer ACCESSORS keiner der Privacy-Mechanismen die Definition datenzentrierter Privacy-Regeln erlaubt, kommt nur ACCESSORS für die Reglementierung der Sensoren und Aktuatoren infrage.

3.2 Privacy-Mechanismen für Datenstromsysteme

In diesem Abschnitt werden Privacy-Mechanismen vorgestellt, mit denen die Verarbeitung von Datenströmen reglementiert werden kann. In Tabelle 3.2 werden die vorgestellten Privacy-Mechanismen verglichen. **Lindner und Meier** [LM06] schlagen eine Architektur für Datenstromsysteme vor, die die rollenbasierte Zugriffskontrolle erweitert und so den Datenstrom vor unberechtigtem Zugriff schützt. In dieser Architektur wird anhand der Rolle, die einer Anwendung zugeordnet ist, entschieden, auf welche Daten die Anwendung zugreifen darf. Die Grundidee besteht darin, einen neuen Operator auf den Datenstrom anzuwenden, der die Datensätze aus dem Datenstrom entfernt, die nicht den Zugriffsregeln entsprechen. Auf diese Weise enthält der Datenstrom nur noch die Datensätze, auf die die Anwendung zugreifen darf. Dieser Ansatz hat jedoch den Nachteil, dass zunächst auch Datensätze verarbeitet werden, die später wieder aus dem Datenstrom entfernt werden müssen. Dadurch entsteht ein unnötiger Berechnungsaufwand. Außerdem kann die vorgestellte Architektur nicht mit Daten aus mehreren Datenströmen umgehen. Ein Smart Home besteht jedoch aus vielen Sensoren, die viele Datenströme erzeugen. Zudem ist das Herausfiltern ganzer Datensätze zu restriktiv, da diese Datensätze dadurch für die spätere Verarbeitung nicht mehr zur Verfügung stehen. Stattdessen sollten die Muster, die aus den Datensätzen abgeleitet werden können, nach bestimmten Bedingungen verschleiert werden.

Anforderung	Lindner und Meier	ACStream	StreamShield	PATRON
Zugriffskontrolle	attributbasiert	attributbasiert	attributbasiert	musterbasiert
Umgang mit unerwünschten Daten	nachträglich entfernen	sofort entfernen	nachträglich entfernen	verschleiern
Keine Übertragung unerwünschter Daten	✗	✓	✗	✓
Daten aus mehreren Datenquellen	✗	✓	✓	✓

Tabelle 3.2: Evaluation der Privacy-Mechanismen für Datenstromsysteme

Carminati et al. schlagen mit **ACStream** [CCF+09] ebenfalls ein Modell vor, das die rollenbasierte Zugriffskontrolle erweitert. Dabei werden die Datensätze, die den Zugriffsregeln nicht entsprechen, jedoch nicht erst im Nachhinein entfernt. Stattdessen wird, sobald das Datenstromsystem eine Anfrage erreicht, gemäß den Zugriffsregeln geprüft, ob der Zugriff vollständig oder nur teilweise erteilt werden darf. Darf der Zugriff nur teilweise erteilt werden, wird die Anfrage so umgeschrieben, dass im Ergebnis nur Datensätze enthalten sind, auf die die anfragende Anwendung zugreifen darf. Mit ACStream ist es möglich, feingranulare Zugriffsregeln für die Datenströme zu definieren. Mithilfe weiterer Bedingungen kann der Zugriff beispielsweise auf aggregierte Daten eingeschränkt werden. Jedoch arbeiten die Zugriffsregeln in ACStream auf Attributebene und erlauben daher nicht die Verschleierung komplexerer Muster [SDM+18].

Nehme et al. stellen mit **StreamShield** [NLB+09] einen Ansatz vor, bei dem sich jeder einzelne Operator selbst überwacht. Dazu wird vor der Übertragung eines Datensatzes zunächst ein Metadatensatz übertragen, der beschreibt, unter welchen Bedingungen auf die Daten im Datensatz zugegriffen werden darf. Nur wenn der Operator die dort beschriebenen Bedingungen erfüllt, darf der Datensatz verarbeitet werden. Ansonsten muss der Datensatz verworfen werden. Da die Überprüfung der Zugriffsbedingungen in den einzelnen Operatoren stattfindet, setzt dieser Ansatz eine vertrauenswürdige Ausführungsumgebung voraus. Genau wie beim Ansatz von Lindner und Meier [LM06] werden jedoch auch hier Datensätze übertragen, die später verworfen werden, da die Prüfung der Zugriffsbedingungen erst in den Operatoren stattfindet und nicht schon vorher. Zudem vergrößert sich das zu übertragende Datenvolumen im Datenstromsystem, da zu jedem Datensatz ein weiterer Metadatensatz übertragen werden muss. Wie ACStream arbeitet StreamShield auf Attributebene, wodurch die Verschleierung komplexerer Muster nicht möglich ist.

PATRON [SDM+17] ist ein System für den Schutz privater Muster in Datenstromsystemen. Dazu werden vorab private und öffentliche Muster definiert. Die privaten Muster müssen

aus dem Datenstrom entfernt werden, da sie die Privacy des Nutzers verletzen. Dabei sollen die öffentlichen Muster so gut wie möglich erhalten bleiben. In Anwendungsszenario 1 in Abschnitt 2.1 soll das private Muster *„Nutzer fährt zum Supermarkt“* verschleiert werden. Das öffentliche Muster *„Nutzer bewegt sich auf das Haus zu“* soll dabei jedoch erhalten bleiben, damit die Auswertung der Wenn-Dann-Regeln weiterhin möglich ist. Dazu erweitert PATRON die Standardarchitektur des Internets der Dinge [WLL+10] um eine Zugriffssteuerungsschicht. Die erste Schicht der Standardarchitektur dient dem Zugriff auf die Datenquellen. Darüber sorgt die neue Zugriffssteuerungsschicht dafür, dass die privaten Muster aus den Datenströmen entfernt werden. In der Datenstromverarbeitungsschicht werden die Daten aus der vorherigen Schicht zusammengeführt und verarbeitet. Zur darüberlegenden Anwendungsschicht gehören die Anwendungen und Dienste, die mit dem Nutzer interagieren. PATRON wird in Abschnitt 5.2 ausführlicher beschrieben.

Li et al. [LOW08], Kim et al. [KSC14], Waye [Way14] und Assam et al. [AHS12] stellen weitere Ansätze vor, bei denen der Datenstrom mithilfe von k-Anonymity, ℓ-Diversity bzw. Differential Privacy anonymisiert wird.

Wie in Tabelle 3.2 zu sehen, arbeitet nur PATRON musterbasiert. Die anderen Systeme arbeiten attributbasiert, was für viele Anwendungsfälle jedoch zu restriktiv ist. Durch die attributbasierte Filterung werden mehr Daten entfernt wie nötig. Die musterbasierte Verschleierung von PATRON sorgt dafür, dass die Daten, die Teile eines privaten Musters sind, so verändert werden, dass das private Muster verschleiert wird. Dabei wird versucht, die öffentlichen Muster so gut wie möglich zu erhalten. Im Gegensatz zum Ansatz von Lindner und Meier und zu StreamShield verhindert PATRON, dass private Muster übertragen werden. Außerdem kann PATRON Muster erkennen, die aus Daten bestehen, die aus mehreren Datenquellen stammen. Das ist notwendig, da es in einem Smart Home viele verschiedene Datenquellen gibt und die vom Nutzer definierten Muster sich über mehrere Datenquellen erstrecken können.

3.3 Zentrale Konfigurationsmechanismen

In diesem Abschnitt werden Systeme beschreiben, mit denen Einstellungen, die sich auf verschiedene Geräte oder Nutzer beziehen, zentral verwaltet werden können. Das **Active Directory** ist ein Verzeichnisdienst, das Informationen über verschiedene Nutzer, Geräte und Ressourcen in einem Unternehmensnetzwerk enthält. Ressourcen sind beispielsweise Speicherplatz, Zugriffsrechte auf Verzeichnisse, Netzwerkdrucker oder Anwendungen. Mithilfe des Active Directory können die Administratoren in einem Unternehmen definieren, auf welche Ressourcen ein bestimmter Nutzer zugreifen darf. Die Administratoren können beispielsweise festlegen, dass nur die Nutzer in der Personalabteilung auf das Verzeichnis zugreifen dürfen, in dem die Personaldaten der Mitarbeiter gespeichert sind. Die Daten des Active Directory sind in einer verteilten relationalen Datenbank gespeichert [Hau17].

Die **Active Directory Rights Management Services** (AD-RMS) sind Teil des Active Directory. Mithilfe der AD-RMS ist es möglich, Dokumente unabhängig vom Speicherort vor unberechtigtem Zugriff zu schützen. Zwar können die Administratoren mithilfe des Active Directory festlegen, dass nur Nutzer in der Personalabteilung auf das Verzeichnis zugreifen dürfen, in dem die Personaldaten der Mitarbeiter gespeichert sind. Jedoch kann ein Nutzer in der Personalabteilung ein Dokument aus diesem Verzeichnis in ein anderes Verzeichnis kopieren, auf das auch andere Nutzer Zugriff haben. In diesem Fall können auch unberechtigte Nutzer auf das Dokument zugreifen. Deshalb wird eine Rechteverwaltung wie die AD-RMS benötigt, die an das Dokument selbst und nicht an einen bestimmten Speicherort gebunden ist. Dazu weisen die Administratoren einer bestimmten Nutzergruppe bestimmte Rechte an einem Dokument zu. Beispielsweise kann allen Nutzern in der Personalabteilung erlaubt werden, die Personalakte eines bestimmten Mitarbeiters zu lesen. Dazu wird das Dokument mithilfe eines Zertifikats verschlüsselt. Möchte ein Nutzer in der Personalabteilung das Dokument öffnen, erwirbt er über den Rechteverwaltungsserver eine Lizenz, die das Öffnen des Dokuments erlaubt. Dabei stellt der Server die Authentizität des Nutzers mithilfe des Active Directory sicher und prüft, ob der Nutzer berechtigt ist, die entsprechende Lizenz zu erwerben [Hau17].

Das **Simple Network Management Protocol** (SNMP) ist ein Protokoll zur Überwachung und Steuerung von Geräten in einem Netzwerk wie beispielsweise Router, Switches, Server oder Drucker durch eine zentrale Administrationskomponente. Das Protokoll regelt die Kommunikation zwischen der Administrationskomponente und den einzelnen Netzwerkgeräten. Auf den Geräten läuft dazu eine Software, die den Zustand des Geräts erkennen und deren Einstellungen verändern kann. Das SNMP definiert neun Nachrichtentypen, mit denen die Administrationskomponente den Zustand der Netzwerkgeräte abfragen und Befehle an die Geräte senden kann. Mithilfe der Befehle GET, GET-NEXT und GET-BULK kann die Administrationskomponente den Zustand eines Geräts auslesen. Beispielsweise kann die Administrationskomponente auf diese Weise den Netzwerknamen eines Routers abfragen. Der SET-Befehl ermöglicht es der Administrationskomponente einen Parameter des Geräts zu ändern. Beispielsweise kann die Administrationskomponente mithilfe des SET-Befehls die IP-Adresse eines Servers ändern. Nachdem das Gerät den GET- oder SET-Befehl verarbeitet hat, sendet es eine RESPONSE-Nachricht an die Administrationskomponente. Die RESPONSE-Nachricht enthält die Informationen, die durch den GET-Befehl angefordert wurden bzw. eine Bestätigung, dass der gewünschte Parameter verändert wurde. Die RESPONSE-Nachricht kann auch eine Fehlermeldung enthalten, falls das Abrufen bzw. das Verändern des Parameters nicht möglich ist [MS05].

Eine TRAP-Nachricht ermöglicht es dem Netzwerkgerät, unaufgefordert Informationen an die Administrationskomponente zu senden. Kommt es bei einem Gerät zu einem Hardwaredefekt wie beispielsweise einem kaputten Lüfter, kann das Gerät dies der Administrationskomponente mithilfe einer TRAP-Nachricht mitteilen. NOTICATION-, INFORM- und REPORT-Nachrichten sind spezielle TRAP-Nachrichten, die genutzt werden können, um Informationen zwischen unterschiedlichen Administrationskomponenten auszutauschen. Im Gegensatz zu einer TRAP-Nachricht muss eine INFORM-Nachricht vom Empfänger bestätigt werden. Im Gegensatz zum Active Directory werden die Einstellungen der Netzwerkge-

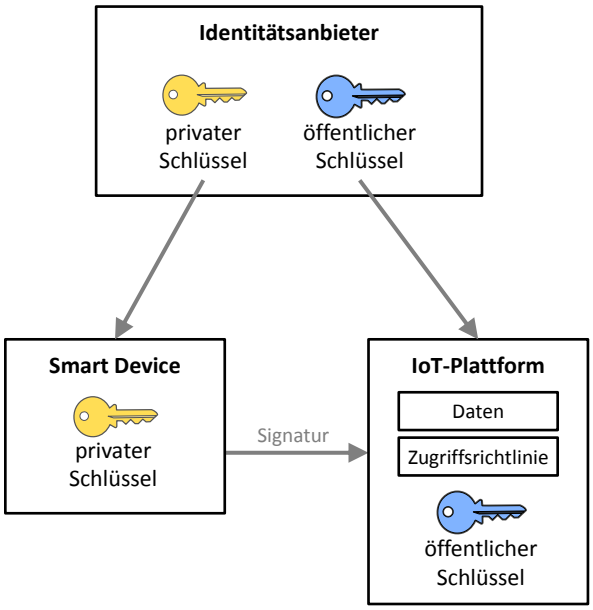

Abbildung 3.2: Sicherstellung der Identität durch ein asymmetrisches Kryptosystem, in Anlehnung an [GÖM18]

räte nicht zentral auf der Administrationskomponente gespeichert und dort verteilt. Stattdessen werden die Einstellungen eines Netzwerkgeräts auf dem jeweiligen Gerät gespeichert und mithilfe des SNMP-Protokolls ausgelesen und verändert [MS05].

CHARIOT [GÖM18] ist ein Ansatz, der Smart Devices mithilfe eines asymmetrischen Kryptosystems eine Identität verleiht. Auf diese Weise kann sichergestellt werden, dass eine Nachricht, die die IoT-Plattform erreicht, tatsächlich von einem bestimmten Smart Device stammt. Die Notwendigkeit dafür kann aus Anwendungsszenario 1 in Abschnitt 2.1 abgeleitet werden. In diesem Szenario darf das Muster *„Nutzer kommt nach Hause“* der Heizung bekannt gegeben werden. Der Kühlschrank darf jedoch nicht erfahren, dass der Nutzer nach Hause kommt. Um die Heizung, den Kühlschrank und andere Smart Devices voneinander unterscheiden zu können, brauchen die Smart Devices eine Identität.

Abbildung 3.2 zeigt, wie ein Smart Device durch ein asymmetrisches Kryptosystem eine Identität erhalten kann. Ein vertrauenswürdiger Identitätsanbieter erzeugt zunächst einen privaten und einen öffentlichen Schlüssel. Das Smart Device erhält den privaten Schlüssel. Der öffentliche Schlüssel wird der IoT-Plattform zur Verfügung gestellt, die die ankommenden Sensordaten aller Smart Devices verarbeitet und die Aktuatoren entsprechend der zuvor festgelegten Regeln steuert. Fordert nun beispielsweise die Heizung bestimmte Daten von der IoT-Plattform an, signiert sie die Nachricht mit dem privaten Schlüssel und schickt sie an die IoT-Plattform. Die IoT-Plattform kann nun mithilfe des öffentlichen Schlüssels sicherstellen, dass die Nachricht tatsächlich von der Heizung stammt. Durch eine Zugriffsrichtlinie wird festgelegt, welche Smart Devices auf welche Daten zugreifen dürfen. Darf die Heizung auf die angeforderten Daten zugreifen, schickt die IoT-Plattform die Daten an die Heizung [GÖM18].

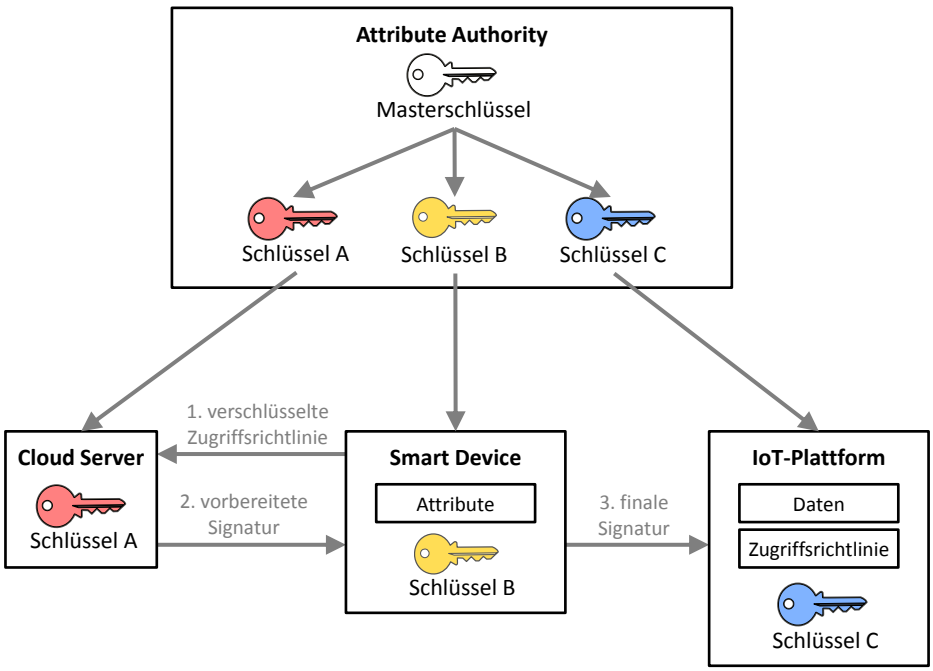

Abbildung 3.3: CHARIOT-Protokoll, in Anlehnung an [GÖM18]

Durch das asymmetrische Verschlüsselungsverfahren ist das Signieren der Nachricht durch das Smart Device jedoch mit einem hohen Berechnungsaufwand verbunden [KMS11]. Da Smart Devices jedoch keine hohe Rechenleistung haben, ist ein asymmetrisches Kryptosystem zur Sicherstellung der Identität der Smart Devices nicht geeignet. CHARIOT ist ein Ansatz, der den Berechnungsaufwand auf dem Smart Device reduziert, indem der Großteil der Berechnung der Signatur auf einen Server in der Cloud ausgelagert wird. CHARIOT stellt außerdem sicher, dass die Identität des Smart Devices gegenüber dem Cloud Server verborgen bleibt. Obwohl der Cloud Server den Großteil der Berechnung der Signatur übernimmt, ist er selbst nicht in der Lage eine Nachricht zu signieren. Das Signieren einer Nachricht ist nur dem Smart Device möglich. Dadurch kann sich der Server auch in einem nicht vertrauenswürdigen Umfeld wie beispielsweise in der Cloud befinden [GÖM18].

Abbildung 3.3 zeigt den Aufbau des CHARIOT-Protokolls. Um das Protokoll zu initialisieren, wird zunächst ein Masterschlüssel für die Attribute Authority erzeugt. Die Attribute Authority dient als vertrauenswürdiger Identitätsanbieter. Die Attribute Authority erstellt mithilfe des Masterschlüssels die drei Schlüssel für die IoT-Plattform, das Smart Device und den Cloud Server [GÖM18]. Der Schlüssel für die IoT-Plattform entspricht dem öffentlichen Schlüssel in Abbildung 3.2. Die beiden Schlüssel für das Smart Device und den Cloud Server entsprechen dem privaten Schlüssel in Abbildung 3.2. Die beiden Schlüssel bestehen aus zwei Teilen eines Geheimnisses, die es erlaubten, die Technik der ausgelagerten Signaturerzeugung [CLH+14] anzuwenden. Zur Erzeugung des Schlüssels für das Smart Device werden die Attribute des Smart Device verwendet. Die Attribute beschreiben Eigenschaften wie beispielsweise den Standort des Smart Devices. Die Attribute des Smart Devices werden in verschlüsselter Form auch in den Schlüssel für den Cloud Server eingebettet [GÖM18].

Möchte das Smart Device auf die IoT-Plattform zugreifen, wird der Großteil der Signaturerzeugung auf den Cloud Server ausgelagert. Dazu schickt das Smart Device zunächst die Zugriffsrichtlinie mit verschlüsselten Attributen an den Cloud Server. Die Zugriffsrichtlinie beschreibt, welche Attribute ein Smart Device besitzen muss, um auf bestimmte Daten auf der IoT-Plattform zugreifen dürfen. Das Smart Device kann die Zugriffsrichtlinie abrufen, darf sie jedoch nicht unverschlüsselt an den Cloud Server weiterleiten, da der Cloud Server nicht als vertrauenswürdig angesehen wird. Der Cloud Server bereitet anschließend mithilfe der verschlüsselten Zugriffsrichtlinie und mit seinem Schlüssel die Signatur vor und sendet sie an das Smart Device zurück. Dabei erfährt der Cloud Server nichts über die Attribute, da die Attribute sowohl in dem Schlüssel für den Cloud Server, als auch in der Zugriffsrichtlinie verschlüsselt sind [GÖM18].

Das Smart Device nutzt die vorbereitete Signatur, um mithilfe seines eigenen Schlüssels die Nachricht an die IoT-Plattform zu signieren. Da die Signatur bereits vom Cloud Server vorbereitet wurde, ist dies nur noch mit geringem Aufwand verbunden. Die Nachricht mit der finalen Signatur wird an die IoT-Plattform geschickt. Die IoT-Plattform kann nun mithilfe der in der Signatur eingebetteten Attribute und der Zugriffsrichtlinie überprüfen, ob das Smart Device auf die angeforderten Daten zugreifen darf [GÖM18].

Die Geräte in einem Smart Home können sich mit Servern außerhalb des Smart Homes verbinden, um Daten an diese Server zu schicken oder Daten abzufragen. Beispielsweise kann sich ein intelligenter Kühlschrank mit dem Server eines Supermarktes verbinden, um Lebensmittel zu bestellen. Um Lebensmittel bestellen zu können, muss sich der Kühlschrank jedoch zuerst bei dem Supermarkt authentifizieren. Dazu kann der Kühlschrank bestimmte Attribute wie Gerätenummer oder die Kundennummer des Nutzers verwenden. Bei der Authentifizierung können jedoch auch sensitive Attribute an den Supermarkt übermittelt werden, die für die Authentifizierung nicht benötigt werden, jedoch die Privacy des Nutzers beeinträchtigen [GÖM19].

Das **Authentifizierungsprotokoll von Gritti et al.** [GÖM19] verlagert die Authentifizierung der Smart Devices an ein Gateway. Das Gateway entfernt die Attribute, die nicht benötigt werden, aus dem Berechtigungsnachweis und stellt so die Privacy des Nutzers sicher. Abbildung 3.4 zeigt einen Anwendungsfall, in dem das Authentifizierungsprotokoll zum Einsatz kommt. In diesem Anwendungsfall möchte sich ein intelligenter Kühlschrank mit dem Server des Supermarktes verbinden, um Lebensmittel zu bestellen. Zur Authentifizierung werden die Gerätenummer des Kühlschranks und die Kundennummer des Nutzers verwendet. Außerdem kann sich der Kühlschrank mit dem Server des Herstellers verbinden, um im Falle eines technischen Problems Unterstützung anzufordern. Zur Authentifizierung bei dem Server des Herstellers werden die Seriennummer, das Modell und das Baujahr des Kühlschranks verwendet.

Das Authentifizierungsprotokoll besteht aus zwei Phasen. In der ersten Phase verbindet sich der Kühlschrank mit dem Gateway und verwendet dazu alle vorhandenen Attribute. Das bedeutet, dass zur Authentifizierung bei dem Gateway auch sensitive Attribute verwendet werden, die nicht an den Supermarkt oder an den Hersteller übermittelt werden sollen. An-

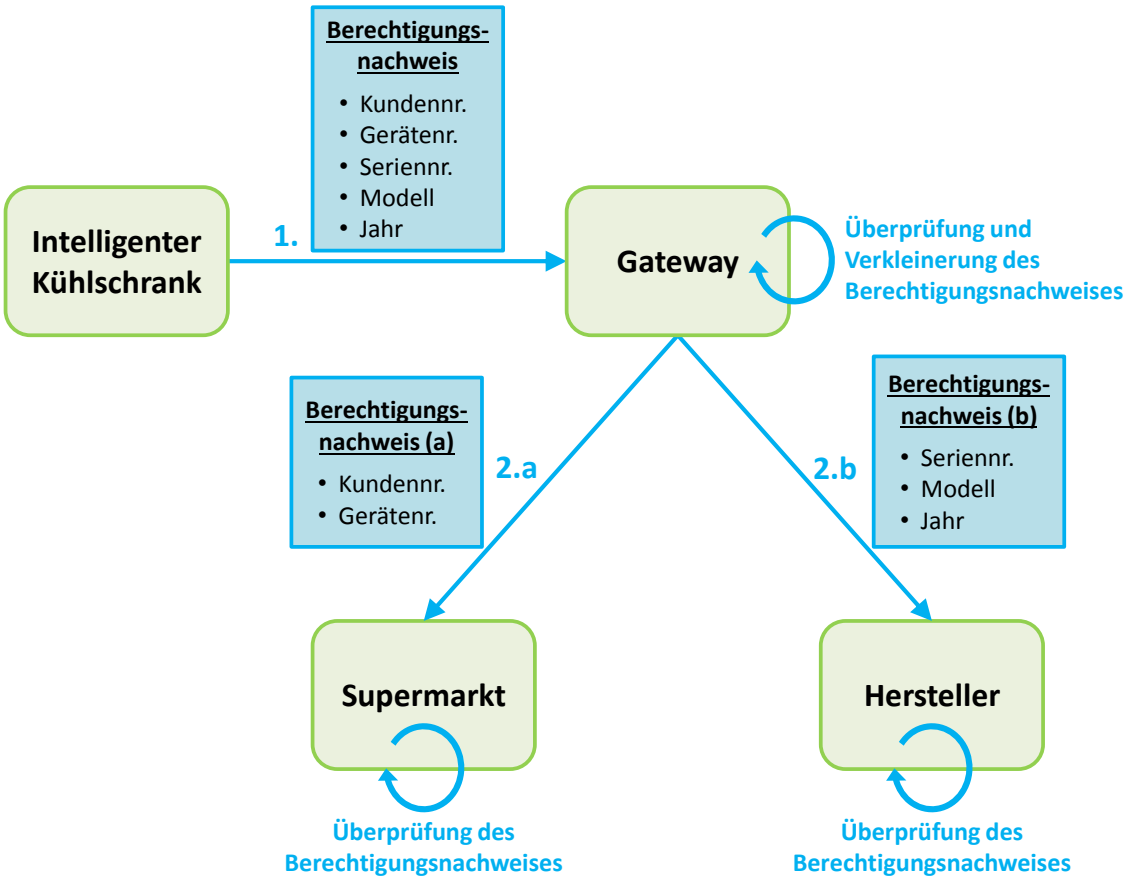

Abbildung 3.4: Authentifizierungsprotoll von Gritti et al. , in Anlehnung an [GÖM19]

schließend überprüft das Gateway anhand der übermittelten Attribute die Authentizität des Kühlschranks. In der zweiten Phase verbindet sich das Gateway mit dem Server des Supermarktes. Wie in Abbildung 3.4 zu sehen, werden dazu nur die Attribute Kundennummer und Gerätenummer verwendet. Da der Supermarkt nur die Kundennummer und die Gerätenummer braucht, um das Gateway bzw. den Kühlschrank zu authentifizieren, wurden die Seriennummer, das Modell und das Baujahr aus dem Berechtigungsnachweis entfernt, um die Privacy des Nutzers zu schützen. Verbindet sich das Gateway mit dem Server des Herstellers, werden nur die Seriennummer, das Modell und das Baujahr des Kühlschranks übermittelt. Die Kundennummer und die Gerätenummer werden von dem Server des Herstellers nicht benötigt und wurden deshalb aus dem Berechtigungsnachweis entfernt [GÖM19].

Durch das Authentifizierungsprotokoll werden gegenüber dem Supermarkt und dem Hersteller immer nur die Attribute preisgegeben, die für die Authentifizierung benötigt werden. Dazu muss jedoch der ursprüngliche Berechtigungsnachweis für jeden einzelnen Dienst modifiziert werden. Da dies einen hohen Berechnungsaufwand erfordert, der von den Smart Devices aufgrund ihrer beschränkten Ressourcen nicht geleistet werden kann, wird die Berechnung der modifizierten Berechtigungsnachweise an das Gateway ausgelagert [GÖM19].

4 Konzept

In diesem Abschnitt werden der Aufbau und die Funktionsweise des Privacy-Systems SPYaware beschrieben. Zunächst wird in Abschnitt 4.1 der schematische Aufbau von SPYaware erklärt und diskutiert, in welchen Komponenten des Systems die Kontrolle der Privacy-Regeln integriert werden kann. Anschließend wird das Konzept von SPYaware in Abschnitt 4.2 vorgestellt. In den Abschnitten 4.3 bis 4.9 werden die einzelnen Komponenten von SPYaware im Detail besprochen. Zuletzt geht es in Abschnitt 4.10 darum, wie die Konfiguration auf die Smart Devices in einem heterogenen Umfeld verteilt werden kann.

4.1 Platzierung der Privacy-Kontrolle

Abbildung 4.1 zeigt den schematischen Aufbau von SPYaware. Zu jedem Sensor gehört ein Adapter, der aus den Rohdaten, die der Sensor liefert, die benötigten Daten extrahiert und der IoT-Plattform in aufbereiteter Form zur Verfügung stellt. Neben Sensoren kann es auch andere Datenquellen wie beispielsweise Datenbanken geben. Die Daten der Datenquellen werden an die IoT-Plattform geschickt und dort verarbeitet. Die IoT-Plattform analysiert die ankommenden Daten und entscheidet anhand zuvor definierter Regeln, welche Aktionen ausgelöst werden. Anschließend schickt die IoT-Plattform die entsprechenden Aktionen an die Aktuatoren. Neben den Aktuatoren kann es auch andere Datensenken wie beispielsweise Datenbanken geben. Zu jeder Datensenke existiert ein Adapter, der die Aktionen der IoT-Plattform in Befehle für die Aktuatoren umwandelt.

Um die in Abschnitt 2.2 beschriebenen Privacy-Anforderungen umsetzen zu können, muss das System über Komponenten verfügen, die die Daten aus dem Datenstrom entfernen, welche die zuvor definierten Privacy-Regeln verletzen. Abbildung 4.2 zeigt drei Möglichkeiten, wo diese Privacy-Kontrolle platziert werden kann. Die erste Möglichkeit (links in Abbildung 4.2) besteht darin, die Privacy-Kontrolle in die Adapter der Datenquellen und Datensenken zu integrieren. Auf diese Weise können Daten, die die Privacy-Regeln verletzen, sofort aus dem Datenstrom entfernt werden. Dies hat den Vorteil, dass nur die Daten übertragen werden, die die Privacy-Regeln einhalten. Dieses Vorgehen entspricht dem Grundsatz der Datenminimierung in Artikel 5, Absatz 1c der DSGVO [EU16], nach dem nur die Daten erfasst werden dürfen, die für die weitere Verarbeitung erforderlich sind. Außerdem wird so die Menge der zu übertragenden Daten reduziert. Dürfen beispielsweise aus einem Adressbuch nur bestimmte Attribute eines bestimmten Kontaktes verarbeitet werden, macht es keinen Sinn, zunächst das gesamte Adressbuch zu übertragen und erst danach die zulässigen Attribute des Kontaktes herauszufiltern.

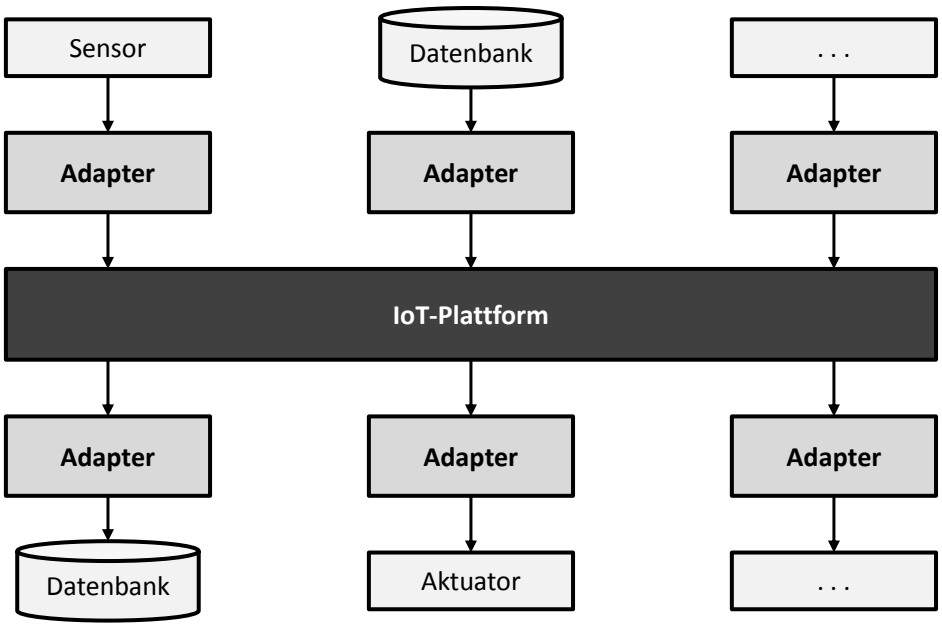

Abbildung 4.1: Schematischer Aufbau von SPYaware

Allerdings können die Daten bei diesem Verfahren nur nach attributbasierten Privacy-Regeln gefiltert werden, wodurch mehr Daten entfernt werden, wie nötig. Um Muster in den Daten einer Datenquelle erkennen und verschleiern zu können, müssen die Daten, die in der Vergangenheit an die IoT-Plattform weitergeleitet wurden, lokal zwischengespeichert werden. Bei der Ankunft neuer Daten kann dann überprüft werden, ob die neuen Daten zusammen mit den bereits übertragenen Daten ein Muster enthalten, das verschleiert werden muss. Da die Smart Devices jedoch nur über sehr wenig Speicher verfügen [GÖM18], ist dies nicht möglich. Außerdem wäre die Erkennung von Mustern, die sich über die Daten mehrerer Datenquellen erstrecken, so nicht möglich.

Eine weitere Möglichkeit besteht darin, die Privacy-Kontrolle auf der IoT-Plattform durchzuführen. Diese Möglichkeit ist in der Mitte von Abbildung 4.2 zu sehen. Im Gegensatz zu dem zuvor beschriebenen Verfahren können so auch Muster erkannt und verschleiert werden, die sich über Daten aus mehreren Datenquellen erstrecken. Da die IoT-Plattform über ausreichend Speicher verfügt, können die bereits weitergegebenen Daten in einer Datenbank zwischengespeichert werden und so bei der Ankunft neuer Daten für die Erkennung von Mustern verwendet werden. Allerdings werden bei dieser Möglichkeit auch Daten an die IoT-Plattform geschickt, die für die weitere Verarbeitung nicht benötigt werden. Diese Daten werden also zunächst von den Datenquellen weitergegeben und müssen anschließend auf der IoT-Plattform gefiltert werden. Die Erfassung und Weitergabe von Daten, die für die eigentliche Verarbeitung nicht benötigt werden, widerspricht dem Grundsatz der Datenminimierung in Artikel 5, Absatz 1c der DSGVO [EU16] und ist daher nicht zulässig.

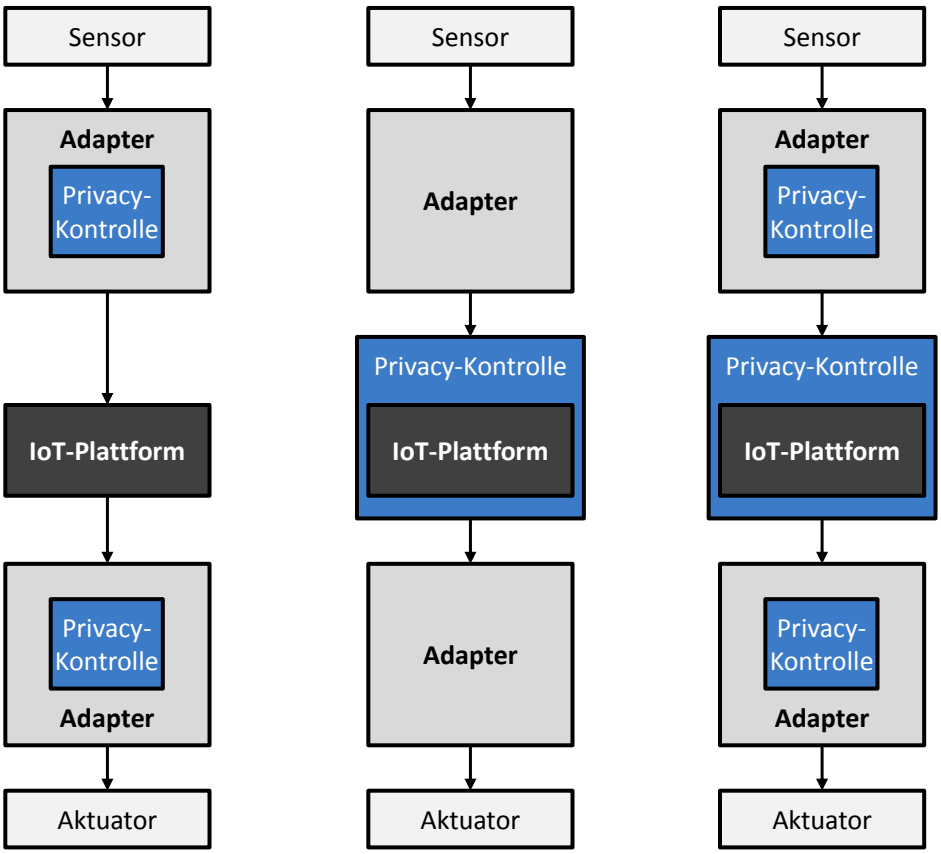

Abbildung 4.2: Platzierung der Privacy-Kontrolle ausschließlich in den Adaptern (links), ausschließlich in der IoT-Plattform (Mitte) oder in den Adaptern und in der IoT-Plattform (rechts)

Die dritte Möglichkeit ist ein hybrides Konzept, das die beiden zuvor beschriebenen Möglichkeiten kombiniert. Dieser Ansatz ist rechts in Abbildung 4.2 zu sehen. Die in die Adapter der Datenquellen integrierte Privacy-Kontrolle filtert die ankommenden Daten nach attributbasierten Privacy-Regeln und leitet nur die Daten weiter, die für die weitere Verarbeitung benötigt werden. Dadurch wird der Grundsatz der Datenminimierung in Artikel 5, Absatz 1c der DSGVO [EU16] eingehalten. Gleichzeitig wird die Menge der zu übertragenden Daten reduziert. Die Privacy-Kontrolle auf der IoT-Plattform ermöglicht die Erkennung und Verschleierung von Mustern, die sich auf Daten aus mehreren Datenquellen beziehen. Die Privacy-Kontrolle in den Adaptern der Datensenken ermöglicht die attributbasierte Reglementierung der Aktuatoren. Dieser Ansatz erfordert eine aufeinander abgestimmte Konfiguration der Privacy-Regeln für die Adapter und die IoT-Plattform. Deshalb sollte die Konfiguration der Privacy-Regeln an einer zentralen Stelle erfolgen. In Abschnitt 4.10 wird beschrieben, wie die Konfiguration der Privacy-Regeln auf die verschiedenen Geräte verteilt werden kann.

4.2 Komponenten von SPYaware

In diesem Abschnitt werden SPYaware und seine Komponenten vorgestellt. Wie in Abbildung 4.3 zu sehen, besteht SPYaware aus fünf Schichten. Die erste Schicht (Nummer 1) besteht aus den Sensoren sowie weiteren Datenquellen. Neben Sensoren kommen auch andere Anwendungen und Systeme wie beispielsweise Datenbanken in Betracht. Die zweite Schicht (Nummer 2) besteht aus den Adaptern. Zu jeder Datenquelle existiert ein zugehöriger Adapter. Ein Adapter besteht aus der Adapterlogik und dem Privacy-Filter. Die Adapterlogik extrahiert aus den empfangenen Daten die gewünschten Werte. Beispielsweise kann aus den von einer Funkuhr empfangenen Daten die korrekte Uhrzeit in einer bestimmten Zeitzone berechnet werden. Der Privacy-Filter filtert die extrahierten Daten gemäß den zuvor definierten Privacy-Regeln. Dabei kann feingranular bestimmt werden, unter welchen Bedingungen Daten weitergegeben werden dürfen. Anstatt die Weitergabe der Daten komplett zu blockieren ist es beispielsweise auch möglich, lediglich die Genauigkeit der Daten zu reduzieren.

Zur dritten Schicht gehört die Zugriffssteuerung (Nummer 3), die Regelausführungskomponente (Nummer 4) und die Konfiguration der Wenn-Dann-Regeln (Nummer 5). Die Zugriffssteuerung (Nummer 3) dient der Verschleierung privater Muster in den ankommenden Datenströmen. Bei der Verschleierung der privaten Muster sollen die öffentlichen Muster so gut wie möglich erhalten bleiben. Nach der Regelausführung wird verifiziert, ob der ausgehende Datenstrom noch private Muster enthält. Die Regelausführungskomponente (Nummer 4) dient der Ausführung der Wenn-Dann-Regeln. Dazu analysiert die Regelausführungskomponente die ankommenden Daten und prüft, ob der Wenn-Bestandteil einer Regel zutrifft. Ist dies der Fall, löst die Regelausführungskomponente die im Dann-Bestandteil festgelegten Aktionen aus. Die Wenn-Dann-Regeln sind im Regelspeicher (Nummer 5) gespeichert und können mithilfe eines Front-Ends bearbeitet werden. Die Regelkonfigurationskomponente stellt die Regeln der Regelausführungskomponente zur Verfügung. Sie kann außerdem Konflikte zwischen Regeln erkennen und bestehende Regeln optimieren, indem beispielsweise Regeln mit identischem Wenn-Bestandteil zu einer einzigen Regel zusammengefasst werden.

Zur vierten Schicht gehören die Adapter für die Datensenken (Nummer 6). Diese Adapter bestehen wie die Adapter für die Datenquellen aus dem Privacy-Filter und der Adapterlogik. Allerdings sind Privacy-Filter und Adapterlogik hier in umgekehrter Reihenfolge angeordnet. Der Privacy-Filter dient der Reglementierung der Aktuatoren. Erfordert eine Wenn-Dann-Regel beispielsweise das Versenden einer SMS-Nachricht, kann durch den Privacy-Filter sichergestellt werden, dass nur eine bestimmte Anzahl an SMS-Nachrichten pro Tag verschickt werden kann. Außerdem könnte der Empfängerkreis der SMS-Nachricht eingeschränkt werden. Die Adapterlogik übersetzt die Aktionen in Befehle für die Aktuatoren. Lautet die Aktion beispielsweise, den Backofen vorzuheizen, so übersetzt die Adapterlogik das in einen Befehl an den Backofen, bis zu einer Temperatur von 200 Grad Celsius zu heizen. Die fünfte Schicht besteht aus den Datensenken (Nummer 7). Datensenken können Aktuatoren oder andere Anwendungen oder Systeme wie beispielsweise Datenbanken sein.

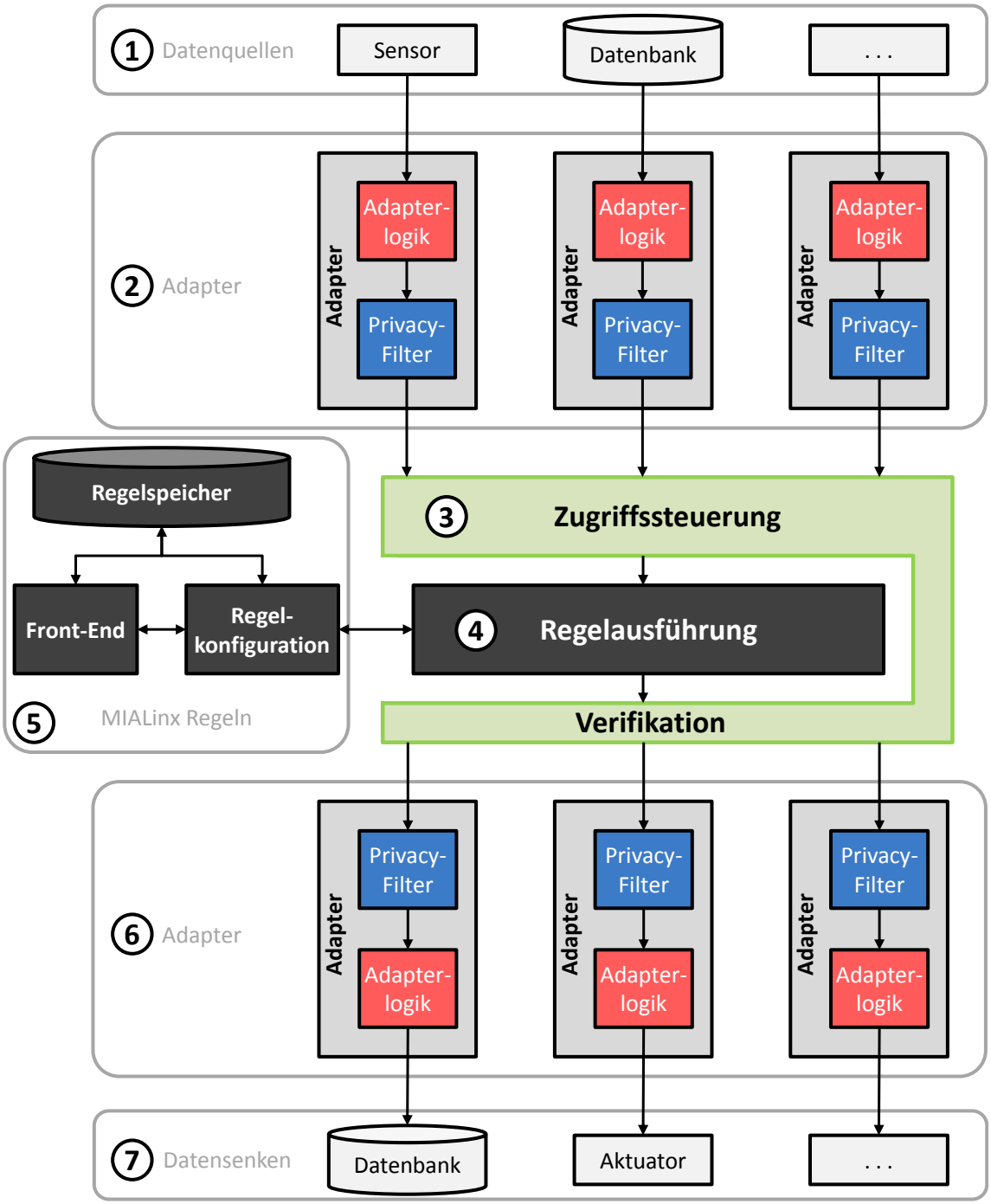

Abbildung 4.3: Komponenten von SPYaware

4.3 Adapterlogik für die Datenquellen

Die Adapterlogik extrahiert die benötigten Werte aus den empfangenen Daten. Handelt es sich um numerische Werte, ist oft eine Umrechnung notwendig. Beispielsweise kann aus der Anzahl der Radumdrehungen die zurückgelegte Strecke berechnet werden oder aus der Temperatur in Grad Fahrenheit die Temperatur in Grad Celsius. Anstatt die Werte direkt ineinander umzurechnen, ist es oft auch möglich, aus mehreren empfangenen Werten den benötigten Wert abzuleiten. Beispielsweise können aus mehreren Positionsangaben mit Längen- und Breitengrad die Geschwindigkeit und die Fahrtrichtung abgeleitet werden. Für die Berechnung der Geschwindigkeit muss jedoch zusätzlich auch die Zeit zwischen den empfangenen Werten bekannt sein.

Handelt es sich hingegen um Zeichenketten, die beispielsweise aus dem Adressbuch eines Smartphones stammen, sind andere Transformationen möglich. Beispielsweise ist es oft notwendig, Zeichenketten miteinander zu verknüpfen oder anhand eines bestimmten Zeichens zu trennen. Dies ist beispielsweise bei der getrennten bzw. gemeinsamen Speicherung von Vor- und Nachname im Adressbuch notwendig. Bei Datumsangaben kann es notwendig sein, diese zunächst in das gewünschte Format zu überführen. Handelt es sich um Daten, die aus einer Datenbank abgefragt werden, kann der SQL-Befehl so angepasst werden, dass die Daten bereits im gewünschten Format geliefert werden.

4.4 Privacy-Filter für die Datenquellen

Der Privacy-Filter dient der Filterung der Daten gemäß den Privacy-Regeln. Je nach Art der Daten kommen unterschiedliche Filtertechniken in Betracht. Die einfachste Filtertechnik ist das vollständige Blockieren der Weitergabe der Daten. Diese Vorgehensweise ist jedoch zu grobgranular, da die blockierten Daten so nicht mehr der späteren Analyse zur Verfügung stehen und dadurch die Servicequalität der Anwendung leidet. Stattdessen werden Filtertechniken benötigt, die nur einen bestimmten Teil der Daten preisgeben oder die Präzision der Daten vor der Weitergabe reduzieren.

Die Präzision numerischer Daten kann durch Rundung des Werts verringert werden. Beispielsweise können Temperaturwerte auf volle Grad Celsius gerundet werden. Alternativ wäre es auch möglich, ein zufälliges Rauschen hinzuzufügen und so den exakten Temperaturwert zu verschleiern. Die Präzision von Datums- und Zeitangaben kann auf die gleiche Weise reduziert werden. Dazu kann beispielsweise das Datum auf volle Monate oder die Uhrzeit auf volle Stunden gerundet werden. Es ist auch möglich, Daten über einen gewissen Zeitraum zu sammeln und anschließend zu aggregieren. Beispielsweise können die Daten eines Temperatursensors über eine Stunde gesammelt und anschließend aggregiert werden, indem man den Mittelwert der Daten bildet. Durch die Aggregation heben sich einzelne Ausreißer in den Daten gegenseitig auf, wodurch sensible Informationen verschleiert wer-

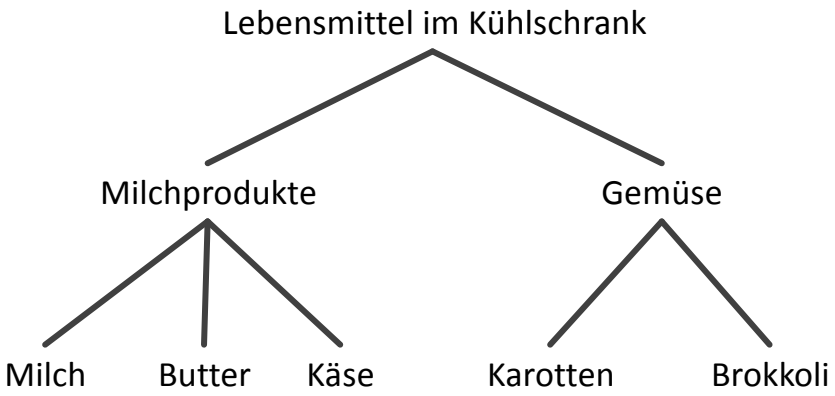

Abbildung 4.4: Hierarchie zur Verallgemeinerung

den. Beispielsweise können Temperaturschwankungen, die durch das kurzzeitige Öffnen und Schließen der Fenster entstehen, so nicht mehr erkannt werden.

Die Präzision von Werten aus einer Aufzählung möglicher Werte zu reduzieren, ist wesentlich schwieriger. Eine Möglichkeit ist es, eine Hierarchie in den Daten zu finden und anschließend auf eine höhere Hierarchieebene zu verallgemeinern. Befinden sich in dem intelligenten Kühlschrank eines Smart Homes beispielsweise die Lebensmittel *Milch*, *Butter*, *Käse*, *Karotten* und *Brokkoli*, können die Lebensmittel entsprechend der Hierarchie in Abbildung 4.4 verallgemeinert werden. Dabei werden *Milch*, *Butter* und *Käse* zu *Milchprodukten* und *Karotten* und *Brokkoli* zu *Gemüse* verallgemeinert. Ob diese Art der Verallgemeinerung möglich ist und wie stark verallgemeinert werden kann, hängt von dem jeweiligen Anwendungsfall ab.

Eine weitere Möglichkeit ist es, nur die Werte zu blockieren, die selten auftreten. Dahinter steckt die Annahme, dass nur die seltenen Werte sensible Daten darstellen. Wenn die Pulswerte des Nutzers den Großteil des Tages im Normalbereich sind, ist ein Wert im Normalbereich kein besonderes Ereignis und kann preisgegeben werden. Ist die Pulsfrequenz jedoch erhöht, stellt dies ein besonderes Ereignis dar, das kritisch für die Privacy des Nutzers ist. Eine Privacy-Regel könnte daher festlegen, dass erhöhte Pulswerte nur an bestimmten Orten oder zu bestimmten Zeiten preisgegeben werden dürfen, Werte im Normalbereich jedoch zu jeder Zeit und an jedem Ort. Abbildung 4.5 zeigt dieses Vorgehen an einem Beispiel. Die schwarzen Punkte liegen im Normalbereich und können daher zu jeder Zeit preisgegeben werden. Punkte außerhalb des Normalbereichs dürfen zwischen 13 Uhr und 16 Uhr nicht preisgegeben werden. Daher werden die beiden roten Punkte blockiert. Die gelben Punkte liegen zwar ebenfalls außerhalb des Normalbereichs, allerdings nicht zwischen 13 Uhr und 16 Uhr. Da die Privacy-Regel nur zwischen 13 Uhr und 16 Uhr angewendet wird, dürfen die gelben Punkte preisgegeben werden.

Am Schwierigsten ist die Filterung boolescher Werte. Kommt eine vollständige Blockierung nicht infrage, ist eine Filterung nur möglich, wenn an anderer Stelle Nachteile in Kauf genommen werden. Ist es akzeptabel, dass ein bestimmter Anteil der Werte falsch weitergegeben wird, kann der boolesche Wert mit einer bestimmten Wahrscheinlichkeit invertiert wer-

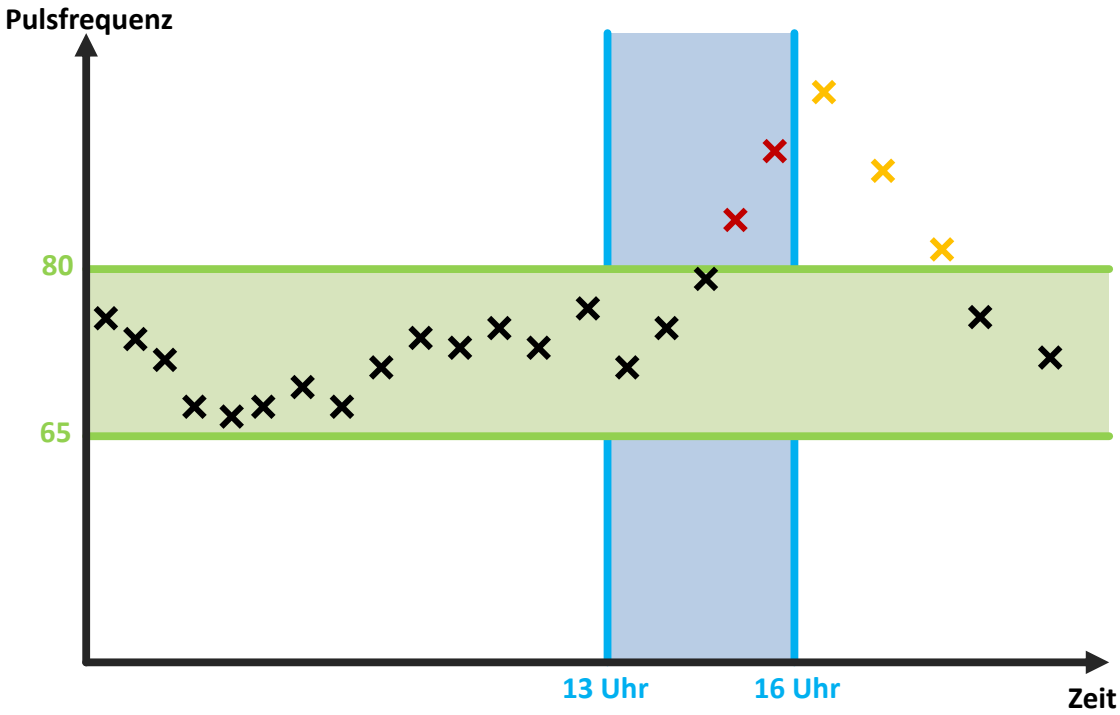

Abbildung 4.5: Blockierung seltener Werte innerhalb eines bestimmten Zeitraums

den, wodurch mit einer bestimmten Wahrscheinlichkeit ein falscher Wert weitergegeben wird. Wird der boolesche Wert beispielsweise mit einer Wahrscheinlichkeit von 20 Prozent invertiert, kann sich der Empfänger nur zu 80 Prozent sicher sein, dass der empfangene Wert auch der tatsächliche Wert ist. Auf diese Weise wird die Datenanalyse zwar erschwert, ist aber ist trotzdem weiterhin möglich. Diese Filtertechnik kann beispielsweise sinnvoll sein, wenn der Sensorwert lediglich zur Absicherung von Wissen benötigt wird, das aus anderen Sensoren gewonnen wurde. Wird beispielsweise anhand der steigenden Raumtemperaturen vermutet, dass die Heizung eingeschaltet ist, kann der Sensor für den Heizungsschalter dieses Wissen mit einer bestimmten Wahrscheinlichkeit bestätigen, ohne den tatsächlichen Zustand des Heizungsschalters sicher preiszugeben. Wenn diese Filtertechnik eingesetzt wird, ist es jedoch notwendig, mehrfache Abfragen des Sensorwerts zu unterbinden. Durch mehrfache Abfragen wäre es ansonsten möglich, invertierte Werte zu erkennen, da diese deutlich seltener auftreten.

Alpers et al. [AEG+18] stellen Filtertechniken vor, mit denen Daten aus dem Adressbuch oder dem Kalender eines Smartphones gefiltert werden können. Diese Filtertechnik wird in Abbildung 4.6 an einem Beispiel veranschaulicht. Verlangt eine Anwendung Zugriff auf die Daten im Adressbuch, ist es oft gar nicht notwendig, den Zugriff auf alle Kontakte und alle Attribute der Kontakte zu gewähren. Durch eine vertikale Filterung kann der Zugriff auf bestimmte Attribute wie beispielsweise Nachname und Stadt beschränkt werden. Diese Attribute sind in Abbildung 4.6 blau hinterlegt. Alle anderen Attribute wie beispielsweise Vorname oder Telefonnummer werden durch leere Zeichenketten ersetzt. Neben der vertikalen Filterung ist auch eine horizontale Filterung möglich. Dabei werden bestimmte Kontakte aus

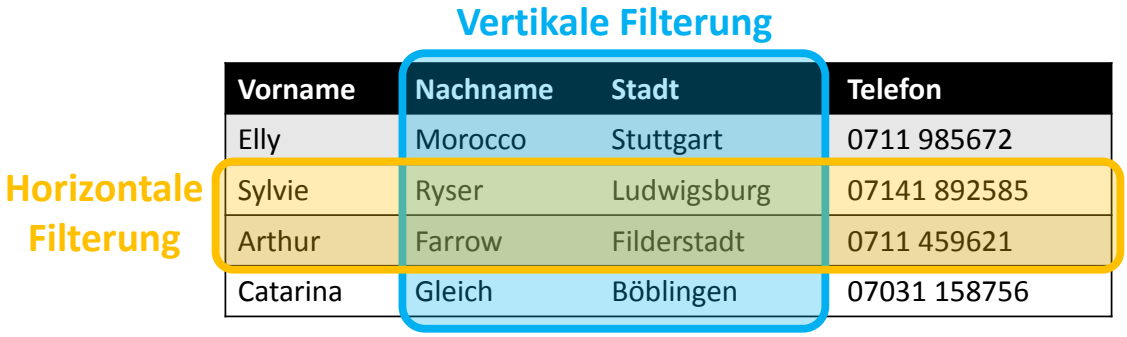

Abbildung 4.6: Verschleierung der Adressbuchdaten, vgl. [AEG+18]

der Kontaktliste, die der Anwendung zurückgegeben wird, entfernt. Die Kontakte, die noch weitergegeben werden, sind in Abbildung 4.6 gelb hinterlegt. Wie in Abbildung 4.6 zu sehen, können die vertikale Filterung und die horizontale Filterung miteinander kombiniert werden.

Auf die Kalenderdaten kann das gleiche Prinzip angewendet werden. Durch eine vertikale Filterung werden nicht alle Attribute eines Kalendereintrags übermittelt. Beispielsweise kann der Zugriff auf den Tag des Kalendereintrags gewährt werden, während die genaue Uhrzeit und der Ort durch leere Zeichenketten überschrieben werden. Eine horizontale Filterung kann durch das Entfernen bestimmter Kalendereinträge aus der Terminliste, die an die Anwendung zurückgegeben wird, erreicht werden. Beispielsweise können alle als privat gekennzeichneten Termine aus dem geschäftlichen Kalender entfernt werden. Neben dem Entfernen von Terminen ist es auch möglich, weitere Termine hinzuzufügen. Dies kann sinnvoll sein, um zu vermeiden, dass die Anwendung die Filterung der Kalendereinträge bemerkt. Beispielsweise können öffentlich verfügbare Ferienwochen hinzugefügt werden. Da diese Informationen öffentlich verfügbar sind, werden keine persönlichen Daten preisgegeben. Gleichzeitig ist es unwahrscheinlich, dass die neu hinzugefügten Informationen das Verhalten der Anwendung ändern, da keine falschen oder nicht plausiblen Daten hinzugefügt werden [AEG+18].

Zur Verschleierung des Standorts schlagen Alpers et al. [AEG+18] eine andere Filtertechnik vor. Diese Filtertechnik wird in Abbildung 4.7 veranschaulicht. Statt der Anwendung die tatsächliche Position mitzuteilen, erhält die Anwendung eine andere Position, die sich in einem bestimmten Umkreis um die tatsächliche Position befindet. Wie groß dieser Umkreis ist, legt der Nutzer fest. Für eine Anwendung, die die Wettervorhersage für den aktuellen Standort liefert, reicht beispielsweise eine Genauigkeit von einigen Kilometern aus. Um die Position zu verschleiern, wird zunächst zufällig ein Winkel zwischen 0 und 360 Grad gewählt. Dieser Winkel ist in Abbildung 4.7 blau dargestellt. Anschließend wird die Position um eine zufällige Entfernung in Richtung des gewählten Winkels verschoben. Die zufällige Entfernung muss dabei kleiner als der vom Nutzer festgelegte Radius sein. Die zufällige Entfernung ist in Abbildung 4.7 grün dargestellt. Sollte die so ermittelte Position in einem anderen Land liegen, wird das Procedere solange wiederholt, bis die tatsächliche Position und die verschleierte Position im gleichen Land liegen. Wurde schon zuvor eine verschleier-

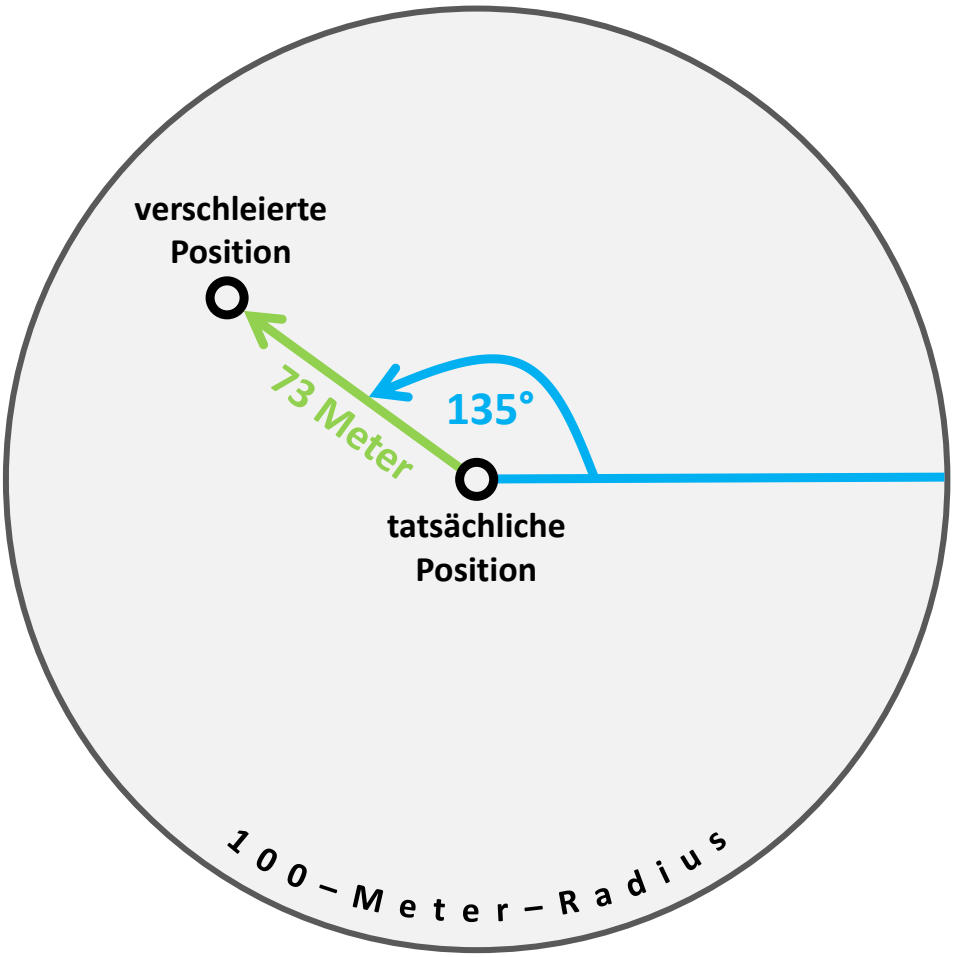

Abbildung 4.7: Verschleierung des Standorts, vgl. [AEG+18]

te Position an die Anwendung geliefert, kommt ein leicht abgewandeltes Vorgehen zum Zuge. Dabei wird zunächst wieder zufällig ein Winkel zwischen 0 und 360 Grad gewählt. Anschließend wird die Distanz zwischen der letzten tatsächlichen Position und der aktuellen tatsächlichen Position berechnet. Dann wird die zuletzt an die Anwendung gelieferte Position um diese Distanz in Richtung des zufällig gewählten Winkels verschoben und an die Anwendung gesendet. Dabei muss darauf geachtet werden, dass sich die so ermittelte Position innerhalb des vom Nutzer definierten Umkreises um die tatsächliche Position und im gleichen Land befindet. Ansonsten muss das Procedere wiederholt werden. Durch dieses Vorgehen bleibt die Fortbewegungsgeschwindigkeit auch bei den verschleierten Positionen erhalten.

Neben den vorgestellten Filtertechniken ist es auch möglich, die Werte zeitlich zu filtern. Anstatt den Standort kontinuierlich zu weiterzugeben, kann die Weitergabe des Standorts auch nur einmal in fünf Minuten erfolgen. Auf diese Weise ist es nicht mehr möglich, die genaue Geschwindigkeit aus den Standortdaten abzuleiten und so beispielsweise Geschwindigkeitsüberschreitungen zu ermitteln. Für andere Anwendungen wie beispielsweise der

Suche nach Tankstellen in der Umgebung wird keine kontinuierliche Übermittlung des Standorts benötigt. Diese Anwendung wird daher durch diesen Filter nicht eingeschränkt. Die zeitliche Filterung kann mit allen anderen Filtertechniken kombiniert werden.

4.5 Zugriffssteuerung

Die Zugriffssteuerung sucht in den ankommenden Datenströmen nach privaten Mustern und verschleiert diese. Dabei sollen die öffentlichen Muster möglichst gut erhalten bleiben, um die Regelausführung so wenig wie möglich zu behindern. Um private Muster zu verschleiern, kommen verschiedene Verschleierungstechniken in Betracht. Die Verschleierungstechnik *blockieren* entfernt die Ereignisse, die Bestandteile eines privaten Musters sind, komplett aus dem Datenstrom. Dabei gehen jedoch viele Informationen verloren. Solange die entfernten Ereignisse nicht zu einem öffentlichen Muster gehören, spielt das für die weitere Verarbeitung keine Rolle. Gehören die entfernten Ereignisse jedoch zu einem öffentlichen Muster, wird durch das Entfernen der Ereignisse auch das öffentliche Muster aus dem Datenstrom entfernt.

Die Verschleierungstechnik *vertauschen* ändert die zeitliche Abfolge der Ereignisse. Wie in Abbildung 5.3 zu sehen, kann durch das Vertauschen der Ereignisse das private Muster entfernt werden, ohne das öffentliche Muster zu zerstören. In dem Beispiel in Abbildung 5.3 werden die Ereignisse *„Nutzer ist bei der Arbeit“* und *„Nutzer ist beim Einkaufen“* vertauscht. Dadurch wird das Muster *„Nutzer geht nach der Arbeit einkaufen“* verschleiert, ohne das Muster *„Nutzer kommt von der Arbeit nach Hause“* zu zerstören.

Die Verschleierungstechnik *verändern* manipuliert einzelne Ereignisse so, dass das private Muster nicht mehr auftritt. Dazu kann beispielsweise der Standort des Supermarktes an eine andere Stelle verschoben werden. In diesem Fall fährt der Nutzer zwar noch den gleichen Weg, da sich der Supermarkt jetzt jedoch an einer anderen Stelle befindet, ist nicht mehr erkennbar, dass der Nutzer am Supermarkt anhält, um dort einzukaufen. Nachdem die privaten Muster verschleiert wurden, wird der veränderte Datenstrom an die Regelausführungskomponente weitergeleitet.

4.6 Verarbeitung der Wenn-Dann-Regeln

Die Verarbeitung der ankommenden Daten erfolgt durch zuvor definierte Regeln. Eine Regel besteht aus einem Wenn-Bestandteil und einem Dann-Bestandteil. Im Wenn-Bestandteil ist beschrieben, welche Bedingungen erfüllt sein müssen. Eine Bedingung bezieht sich hierbei auf Daten in einem oder in mehreren ankommenden Datenströmen. Ist die Bedingung erfüllt, wird der Dann-Bestandteil ausgeführt. Der Dann-Bestandteil beschreibt, welche Aktionen ausgelöst werden sollen. Eine Aktion kann hierbei die Steuerung eines Aktuators, das Einfügen eines Datensatzes in eine Datenbank oder das Senden einer Nachricht an ein anderes System sein.

Die Verarbeitung der ankommenden Daten und die Steuerung der Aktionen könnte auch durch ein anderes System erfolgen. Beispielsweise könnte ein Neuronales Netzwerk anhand von Trainingsdaten lernen, wie es auf bestimmte Ereignisse in den Daten reagieren soll. Die Trainingsdaten könnte das System sammeln, indem es die Aktionen des Nutzers beobachtet. Ein Neuronales Netzwerk hätte jedoch den Nachteil, dass der Nutzer nicht nachvollziehen kann, warum das System eine bestimmte Aktion auslöst. Durch das Lernen unerwünschter Aktionen könnte es zudem zu einem fehlerhaften Verhalten des Systems kommen. Die Nutzung einfacher Wenn-Dann-Regeln bietet demgegenüber den Vorteil, dass der Nutzer das Verhalten des Systems mithilfe der Regeln selbst definieren kann. Entspricht das Verhalten des Systems nicht den Wünschen des Nutzers, kann der die Regeln analysieren und anpassen.

Zur Verarbeitung der Regeln analysiert die Regelausführungskomponente die ankommenden Daten kontinuierlich und prüft dabei, ob der Wenn-Bestandteil einer Regel zutrifft. Ist dies der Fall, wird der Dann-Bestandteil der Regel ausgeführt. Besteht der Dann-Bestandteil einer Regel aus mehreren Aktionen, ist es oft wichtig, dass die Aktionen in einer bestimmen Reihenfolge ausgeführt werden. Um dies sicherzustellen, können Workflow-Technologien genutzt werden. Neben der Ausführung der Regeln untersucht die Regelausführungskomponente die ankommenden Daten auf Ereignisse, die häufig gemeinsam auftreten. Basierend auf diesen Ereignissen und den bereits vorhandenen Regeln können dem Nutzer neue Regeln vorgeschlagen werden.

Die Wenn-Dann-Regeln sind im Regelspeicher persistent gespeichert und können über das Front-End bearbeitet werden. Das Front-End ermöglicht die grafische Modellierung der Regeln. Auf diese Weise ist es auch Nutzern ohne technisches Vorwissen möglich, neue Regeln zu erstellen und bestehende Regeln zu bearbeiten. Um die Konfiguration der Regeln für den Nutzer so einfach wie möglich zu gestalten, können in Abhängigkeit von den verfügbaren Datenquellen und Datensenken bereits einige Regeln vordefiniert sein. Dabei sollte es sich um Regeln handeln, die in einem Smart Home häufig eingesetzt werden. Diese Regeln helfen dem Nutzer dabei, die Funktionsweise des Systems zu verstehen und neue Regeln zu definieren.

Basierend auf den Informationen der Regelausführungskomponente werden im Front-End Vorschläge für neue Regeln angezeigt. Der Nutzer erhält außerdem die Information, welche Regeln in der Vergangenheit wie häufig ausgeführt wurden. Dadurch kann er beispielsweise erkennen, wenn eine Regel selten oder nie ausgeführt wurde. Wurde eine Regel in der Vergangenheit nie ausgeführt, ist diese ein Hinweis darauf, dass die Bedingungen im Wenn-Bestandteil entweder fehlerhaft eingestellt oder zu restriktiv sind. Vor der Speicherung neuer Regeln wird überprüft, ob es bereits gleiche Regeln gibt. Gleiche Regeln sind Regeln, die im Wenn-Bestandteil die gleichen Datenquellen verwenden und im Dann-Bestandteil die gleichen Aktionen auslösen. Ist dies der Fall, wird dem Nutzer vorgeschlagen, die beiden Regeln zu einer einzigen Regel zusammenzufassen.

Das Front-End ermöglicht auch die temporäre Deaktivierung einzelner Regeln. Das kann beispielsweise dann sinnvoll sein, wenn der Nutzer aufgrund einer Dienstreise längere Zeit

nicht zuhause ist. Eine deaktivierte Regel ist zwar noch im Regelspeicher gespeichert und wird auch noch im Front-End angezeigt, wird jedoch nicht mehr der Regelausführungskomponente zur Verfügung gestellt. Da auf diese Weise die ankommenden Daten auf weniger Bedingungen geprüft werden müssen, wird die Ausführung durch die Deaktivierung nicht genutzter Regeln insgesamt beschleunigt.

Die Regelkonfigurationskomponente optimiert die im Regelspeicher gespeicherten Regeln und stellt diese der Regelausführungskomponente zur Verfügung. Dazu wird beispielsweise überprüft, ob es Konflikte zwischen Regeln gibt. Existieren Regeln mit gleichem Wenn-Bestandteil, werden diese zu einer einzigen Regel zusammengefasst, um so die Ausführung zu beschleunigen. Kann der Wenn-Bestandteil einer Regel nie erfüllt sein oder ist der Dann-Bestandteil der Regel leer, wird die Regel aus dem Regelspeicher gelöscht.

4.7 Verifikation

Aufgabe der Verifikation ist es, zu überprüfen, ob die Verschleierung der privaten Muster erfolgreich war. Die Verschleierung war dann erfolgreich, wenn der ausgehende Datenstrom keine privaten Muster mehr enthält, aber alle öffentlichen Muster weiterhin vorhanden sind. Zudem dürfen durch die Verschleierung keine zusätzlichen Muster eingeführt werden, die in dem ursprünglichen Datenstrom nicht vorhanden sind. Um dies zu überprüfen, werden alle Datenströme, die an die Regelausführungskomponente geschickt werden, und alle Datenströme, die die Regelausführungskomponente verlassen, kontrolliert. Auf diese Weise hat die Verifikationskomponente einen detaillierten Überblick darüber, welche Informationen der Regelausführungskomponente zur Verfügung stehen.

Dazu werden alle Daten, die an die Regelausführungskomponente geschickt werden, parallel an die Verifikationskomponente gesendet. Die Verifikationskomponente untersucht den Datenstrom anschließend auf private Muster. Die Überwachung der Datenströme, die die Regelausführungskomponente verlassen, gestaltet sich schwieriger. Das liegt daran, dass die ausgehenden Datenströme nicht mehr die Sensordaten enthalten, sondern Befehle für die Aktuatoren. Für die Verifikationskomponente ist zunächst nicht ersichtlich, aufgrund welcher Sensordaten die Befehle an die Aktuatoren geschickt werden.

Eine Möglichkeit besteht darin, die ausgehenden Befehle mit den im Regelspeicher gespeicherten Regeln abzugleichen. Auf diese Weise kann man herausfinden, welche Regeln und somit welche Sensordaten die Befehle ausgelöst haben. Anschließend kann überprüft werden, ob dabei private Muster in den Sensordaten erkannt wurden. Gibt es jedoch mehrere Regeln, die einen bestimmten Befehl auslösen können, kann durch dieses Vorgehen nicht herausgefunden werden, welche der Regeln den Befehl ausgelöst hat. Außerdem könnte die Regelausführungskomponente noch weitere Regeln berücksichtigen, die im Regelspeicher nicht enthalten sind und private Muster nutzen.

Eine andere Möglichkeit besteht darin, dass die Regelausführungskomponente die ausgehenden Befehle mit den Regeln annotiert, die zur Auslösung des Befehls geführt haben. Auf

diese Weise ist sofort ersichtlich, welche Regel zur Auslösung eines bestimmten Befehls geführt hat. Außerdem ist es so nicht mehr möglich, Befehle aufgrund unbekannter Regeln auszuführen. Ist die Regel bekannt, kann die Verifikationskomponente prüfen, ob in der Bedingung im Wenn-Bestandteil der Regel private Muster enthalten sind.

Sind in den Datenströmen, die an die Regelausführungskomponente geschickt werden oder die Regelausführungskomponente verlassen, noch private Muster vorhanden, muss die Gewichtung der Muster in der Konfiguration angepasst werden. Die Gewichtung der Muster beeinflusst die Qualitätsmetrik, die zur Auswahl der Verschleierungstechnik genutzt wird. Gehen durch die Verschleierung öffentliche Muster verloren, müssen die Gewichte der jeweiligen Muster ebenfalls in der Konfiguration angepasst werden.

4.8 Privacy-Filter für die Datensenken

Der Privacy-Filter in den Adaptern der Datensenken dient der Reglementierung der Aktuatoren gemäß den Privacy-Regeln. Je nach Art der Datensenke können unterschiedliche Filtertechniken zum Einsatz kommen. Handelt es sich bei der Datensenke um ein anderes System wie beispielsweise eine Datenbank, können die gleichen Filtertechniken wie bei den Privacy-Filtern für die Datenquellen zum Einsatz kommen.

Handelt es sich bei den Datensenken um Aktuatoren, kann die Nutzungshäufigkeit des Aktuators eingeschränkt werden. Eine Privacy-Regel kann beispielsweise festlegen, dass nur eine bestimmte Anzahl an SMS-Nachrichten pro Tag verschickt werden darf. Außerdem kann der Empfängerkreis der SMS-Nachrichten eingeschränkt werden. Auf diese Weise kann verhindert werden, dass persönliche Informationen an unerwünschte Empfänger wie beispielsweise die Versicherungsgesellschaft geschickt werden.

Mithilfe des Privacy-Filters ist es zudem möglich, die Verwendung bestimmte Funktionen des Aktuators oder eines bestimmten Betriebsmodus zu verhindern. Handelt es sich bei dem Aktuator beispielsweise um eine Heizung, kann festgelegt werden, dass die Heizung maximal auf Stufe 3 eingestellt werden darf. Der Privacy-Filter kann auch verwendet werden, um die Nutzung der Ressourcen zeitlich oder räumlich einzuschränken. Beispielsweise kann festgelegt werden, dass der Backofen erst nach 17 Uhr eingeschaltet werden darf.

4.9 Adapterlogik für die Datensenken

Aufgabe der Adapterlogik in den Adaptern der Datensenken ist es, die Aktionen der Regelausführungskomponente in Befehle für die Aktuatoren zu übersetzen. Erwartet der Aktuator einen numerischen Wert, muss dieser oft erst berechnet werden. Soll beispielsweise die Zimmertemperatur auf 22 Grad Celsius erhöht werden, kann daraus errechnet werden, dass der Thermostat der Heizung eine halbe Stufe höher eingestellt werden muss. Es kann auch notwendig sein, eine Zeichenkette in einen numerischen Wert zu übersetzen. Soll beispiels-

weise eine SMS-Nachricht an eine bestimmte Person gesendet werden, muss anhand des Namens der Person und mithilfe des Adressbuchs zunächst die zugehörige Mobilfunknummer ermittelt werden.

Erwartet der Aktuator eine Zeichenkette, können andere Transformationen notwendig sein. Oft ist es beispielsweise notwendig, Zeichenketten miteinander zu verknüpfen oder an einer bestimmten Stelle zu trennen. Bei Datumsangaben kann es notwendig sein, diese zunächst in das gewünschte Format zu bringen. Jedes Zielsystem kann ein anderes Datenformat erwarten, in das der Befehl eingebettet ist. Handelt es sich beispielsweise um eine Datenbank, muss der Befehl in einen SQL-Befehl übersetzt werden. Erwartet das Zielsystem ein XML-Dokument, muss der Befehl in ein XML-Dokument eingebettet werden, das der vom Zielsystem erwarteten Struktur entspricht.

4.10 Verteilung der Konfiguration auf die Smart Devices

Wie in Abschnitt 4.1 beschriebenen, muss die Konfiguration der Privacy-Regeln an einer zentralen Stelle erfolgen und von dort auf die Smart Devices verteilt werden. Die Verteilung der öffentlichen und privaten Muster ist unproblematisch, da das Datenstromsystem und die Zugriffssteuerung genau wie die Konfigurationskomponente auf der IoT-Plattform laufen. Die Sensoren und Aktuatoren können sich jedoch an einem beliebigen Ort befinden. Dieser Ort kann das Smart Home, ein Auto oder ein anderer Ort sein. Daher ist es notwendig, die zentral definierten Privacy-Regeln in einem heterogenen Umfeld verteilen zu können.

Dazu kommen zwei Ansätze in Betracht. Der erste Ansatz besteht darin, die Privacy-Regeln der Datenquellen und -senken direkt in den zugehörigen Adaptern zu speichern. Die dort gespeicherten Privacy-Regeln können wie beim in Abschnitt 3.3 vorgestellten Simple Network Management Protocol [MS05] über eine Schnittstelle abgefragt und verändert werden. Der Vorteil dieses Ansatzes besteht darin, dass die Privacy-Regeln dort gespeichert werden, wo sie benötigt werden. Kommt ein neuer Datensatz an, können die Privacy-Regeln sofort angewendet werden und müssen nicht erst von der IoT-Plattform abgerufen werden. Allerdings ist das Abfragen und Verändern der Privacy-Regeln nur möglich, wenn das Smart Device mit der IoT-Plattform kommunizieren kann. Befindet sich das Smart Device in einem Auto, das durch einen Tunnel fährt, ist die Verbindung unterbrochen und das Abfragen und Verändern der Privacy-Regeln dadurch nicht möglich.

Der zweite Ansatz besteht darin, die Privacy-Regeln auf der IoT-Plattform zu speichern. Auf diese Weise können die Privacy-Regeln jederzeit abgefragt und verändert werden. Wird eine Privacy-Regel verändert, wird die Änderung sofort auf das betroffene Smart Device übertragen. Ist das Smart Device nicht verfügbar, werden die Änderungen erst dann übertragen, wenn das Smart Device wieder verfügbar ist. Die Filterung der Daten erfolgt trotzdem entsprechend der neuen Privacy-Regeln, da das Smart Device erst dann Daten an die IoT-Plattform senden kann, wenn es wieder mit der IoT-Plattform kommunizieren kann und dadurch auch die neuen Privacy-Regeln übertragen werden können.

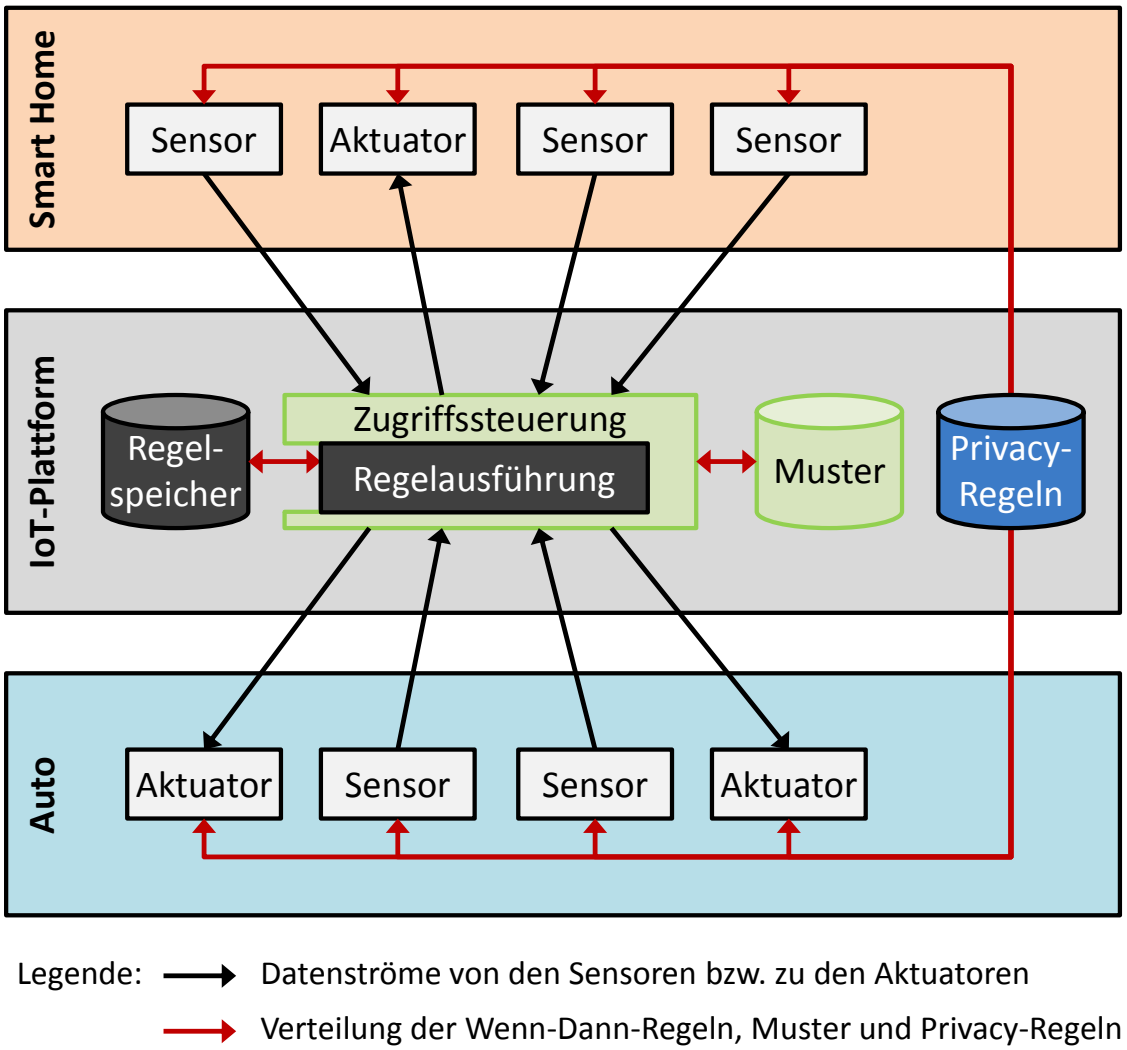

Abbildung 4.8: Verteilung der Konfiguration

Abbildung 4.8 zeigt, wie die Privacy-Konfiguration auf die Datenquellen und -senken verteilt wird. Das Smart Home und das Auto in Abbildung 4.8 sind Beispiele für heterogene Umgebungen, in denen sich die Sensoren und Aktuatoren befinden können. Die Sensoren und Aktuatoren im Smart Home und im Auto senden Daten an die IoT-Plattform bzw. empfangen Daten von der IoT-Plattform. Die Datenströme zwischen den Sensoren bzw. den Aktuatoren und der IoT-Plattform sind schwarz dargestellt. Auf der IoT-Plattform laufen die Zugriffssteuerung und die Regelausführung. Die Wenn-Dann-Regeln im Regelspeicher sind ebenso wie die öffentlichen und privaten Muster für die Zugriffssteuerung auf der IoT-Plattform gespeichert. Auch die Privacy-Regeln, die die Sensoren und Aktuatoren reglementieren, werden zentral auf der IoT-Plattform gespeichert und von dort aus an die Sensoren und Aktuatoren verteilt. Die Anpassung der Wenn-Dann-Regeln, der privaten Muster und der Privacy-Regeln kann mithilfe eines Webinterfaces zu jeder Zeit und von jedem Ort aus erfolgen.

Für den sicheren Austausch der Daten zwischen den Smart Devices und dem Datenstromsystem kann das in Abschnitt 3.3 beschriebene CHARIOT-Protokoll [GÖM18] verwendet werden. Durch CHARIOT erhalten die Smart Devices eine Identität in Form eines privaten Schlüssels. Mithilfe des privaten Schlüssels können die Sensoren ihre Nachrichten an das Datenstromsystem signieren. Das Datenstromsystem kann so sicherstellen, dass die Nachricht tatsächlich von einem bestimmten Sensor stammt. Mithilfe des CHARIOT-Protokolls kann ein Großteil der Berechnung der Signatur ausgelagert werden, sodass das Signieren der Nachricht auch für ein leistungsschwaches Smart Device möglich ist. Auch die Befehle an die Aktuatoren können auf diese Weise verschlüsselt und vom Smart Device wieder entschlüsselt werden. Durch das CHARIOT-Protokoll sind die Nachrichten zwischen Smart Device und Datenstromsystem verschlüsselt. Auf diese Weise ist es einem Außenstehenden nicht möglich, die Nachrichten mitzulesen oder zu manipulieren. Dies schützt die Privacy des Nutzers und trägt zur Datensicherheit des Systems bei.

5 Grundlagen

In diesem Abschnitt werden die Systeme beschreiben, die in Abschnitt 6 für die Umsetzung des Konzepts von SPYaware verwendet werden. Zunächst wird ein System benötigt, das Datenströme in Echtzeit verarbeiten kann und auf bestimmte Muster in den Datenströmen gemäß zuvor definierten Wenn-Dann-Regeln reagieren kann. Das in Abschnitt 5.1 beschriebene MIALinx erfüllt diese Anforderungen. Da durch MIALinx Daten aus vielen verschiedenen Datenquellen verarbeitet werden, kann es sein, dass dabei Informationen sichtbar werden, die der Nutzer nicht preisgeben möchte. Das in Abschnitt 5.2 beschriebene PATRON schützt die Privacy des Nutzers, indem es ermöglicht, Muster zu definieren, die nicht an MIALinx weitergeleitet werden dürfen.

Damit Daten, die eine Gefahr für die Privacy des Nutzers darstellen, aber für die Verarbeitung nicht benötigt werden, erst gar nicht zu dem Datenstromsystem gelangen, können die Sensordaten mithilfe der in Abschnitt 5.3 vorgestellten PMP gefiltert werden. Die PMP ermöglicht außerdem auch die Reglementierung der Aktuatoren. ACCESSORS wird in Abschnitt 5.4 beschrieben und ist eine Erweiterung des Berechtigungsrichtlinienmodells der PMP. Da die beschriebenen Systeme in einem heterogenen Umfeld ausgeführt werden, ist es wichtig, die Systeme an einer zentralen Stelle konfigurieren zu können. Das in Abschnitt 5.5 erläuterte AVARE bietet eine Möglichkeit dafür.

5.1 MIALinx

Die Digitalisierung der industriellen Produktion, die auch als Industrie 4.0 bezeichnet wird, stellt vor allem für kleine und mittlere Unternehmen eine Herausforderung dar [LKS+16]. Der Grund dafür ist, dass diese Unternehmen weder das notwendige Wissen, noch das notwendige Budget haben, um in neue Industrie-4.0-Systeme zu investieren [WHS+16]. MIALinx bietet für diese Unternehmen eine leichtgewichtige und einfach zu bedienende Integrationslösung an, die die Integration der Sensoren und Aktuatoren in die bestehenden Systeme ermöglicht. Dazu können in MIALinx Wenn-Dann-Regeln formuliert werden, die beschreiben, wie ein Aktuator auf ein Muster in den Sensordaten reagieren soll. Eine Regel könnte beispielsweise festlegen, dass im Fall einer defekten Maschine ein Servicetechniker informiert und das passende Ersatzteil bestellt werden soll [WHS+16]. Im Umfeld des Smart Homes könnte eine Regel bestimmen, dass der Backofen vorgewärmt werden soll, wenn der Nutzer nach Hause kommt.

Abbildung 5.1 zeigt die Architektur von MIALinx. Die bereits bestehenden Komponenten sind weiß hinterlegt, die Komponenten, die durch MIALinx erweitert werden, hellgrau und

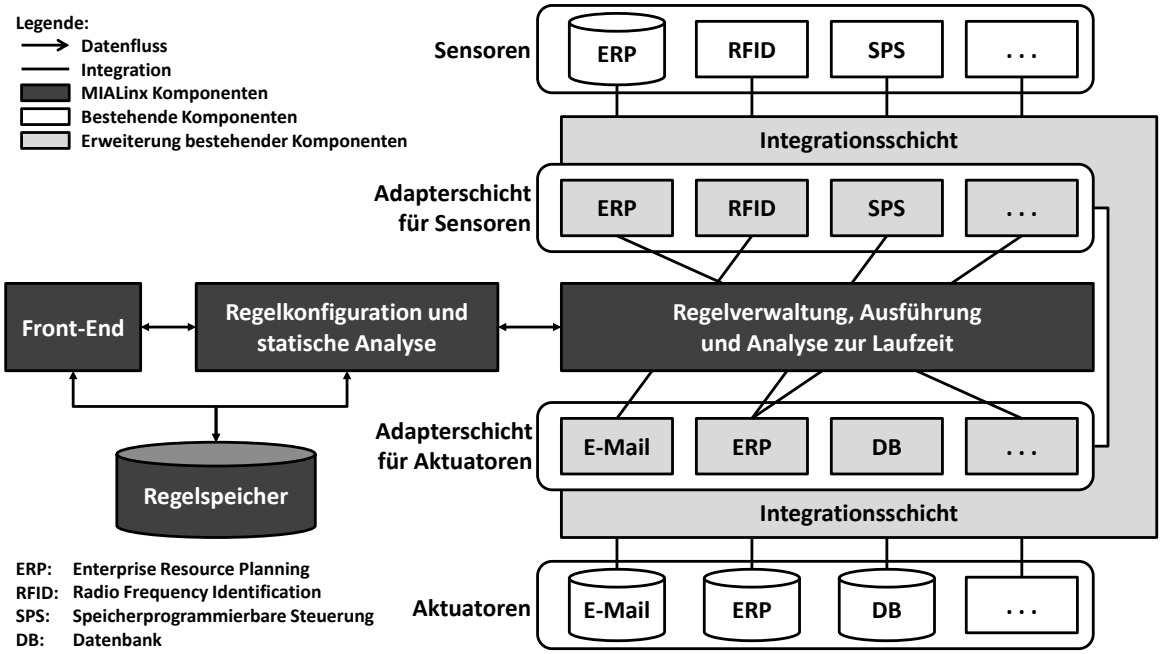

Abbildung 5.1: Architektur von MIALinx, aus dem Englischen übersetzt aus [WHS+16]

die neu hinzugekommenen Komponenten dunkelgrau. Links befindet sich das Front-End, mit dem die Regeln von Domänenexperten grafisch modelliert werden können. Durch die grafische Modellierung können die Regeln von Domänenexperten aus vielen verschiedenen Bereichen modelliert werden, ohne dass dabei spezielle IT-Kenntnisse notwendig sind. Die so modellierten Regeln werden im Regelspeicher gespeichert [WHS+16].

Das Gegenstück zum Front-End ist die Regelkonfigurationskomponente. Die im Front-End erstellten und im Regelspeicher gespeicherten Regeln werden hier analysiert und auf Konflikte untersucht. Außerdem werden Regeln mit gleichem Wenn-Bestandteil zu einer Regel zusammengefasst. Dadurch wird die Ausführung der Regeln beschleunigt und die Darstellung der Regeln im Front-End übersichtlicher. Die Regelkonfigurationskomponente kümmert sich außerdem um die Bereitstellung und Ausführung der im Front-End erstellten oder veränderten Regeln [WHS+16].

Die Adapterschichten für die Sensoren und die Aktuatoren beschreiben, welche Daten die Sensoren liefern und wie die Aktuatoren gesteuert werden können. Dazu werden die Daten der Sensoren konvertiert und so weiterverarbeitet, dass sie in der Regelausführungskomponente genutzt werden können. Auf diese Weise stellen die Adapterschichten eine Schnittstelle zu den Sensoren und Aktuatoren bereit. Auch die Anbindung weiterer Systeme ist durch die Adapterschichten ohne großen Aufwand möglich. Neben Sensoren können auch andere Systeme wie beispielsweise ERP-Systeme (Enterprise-Resource-Planning) als Datenquellen genutzt werden. Statt einen Aktuator zu steuern ist es beispielsweise auch möglich, in einer Datenbank einen neuen Datensatz zu erstellen [WHS+16].

Die Integrationsschicht verbindet die Sensoren und Aktuatoren mit der Regelausführungskomponente. Die Regelausführungskomponente nutzt die Sensordaten, um die Bedingungen im Wenn-Bestandteil der Regeln auszuwerten. Trifft die Bedingung zu, werden die Aktuatoren entsprechend dem Dann-Bestandteil der Regel gesteuert. Sollen mehrere Aktuatoren gesteuert werden, ist die Reihenfolge, in der die Aktuatoren aufgerufen werden, oft wichtig. Wenn beispielsweise ein Ersatzteil bestellt und ein Servicetechniker zum Einbau des Ersatzteils angefordert werden soll, macht es nur Sinn den Servicetechniker anzufordern, wenn die Bestellung des Ersatzteils erfolgreich war. Die Regelausführungskomponente analysiert außerdem zur Laufzeit die ankommenden Sensordaten und schlägt basierend auf den ankommenden Daten und den bestehenden Regeln neue Regeln vor [WHS+16].

5.2 PATRON

PATRON [SDM+17] ist ein System für den Schutz privater Informationen in Datenstromsystemen. Die Grundidee ist, vorab definierte private Muster aus dem Datenstrom zu entfernen und dabei die öffentlichen Muster so gut wie möglich zu erhalten. In Anwendungsszenario 1 in Abschnitt 2.1 soll beispielsweise das Muster *„Nutzer fährt zum Supermarkt“* verschleiert werden. Das Muster *„Nutzer kommt nach Hause“* muss dabei jedoch erhalten bleiben, um die Servicequalität der Anwendung nicht zu beeinträchtigen. Die Servicequalität ist dann hoch, wenn die Anwendung möglichst viele Daten erhält und so den maximalen Nutzen erbringen kann. Dabei dürfen jedoch keine privaten Muster auftreten [SDM+17].

Die Standardarchitektur des Internets der Dinge [WLL+10] besteht aus drei Schichten. Diese Schichten sind in Abbildung 5.2 blau dargestellt. Die unterste Schicht dient dem Zugriff auf die Datenquellen. Datenquellen können andere Anwendungen, Sensoren oder Datenbanken sein. Die Daten aus den Datenquellen werden in der Datenstromverarbeitungsschicht zusammengeführt und verarbeitet. Zur Anwendungsschicht gehören die Anwendungen und Dienste, die mit dem Nutzer interagieren. PATRON erweitert die Standardarchitektur um eine Verifikations- und Konfigurationsschicht und eine Schicht zur Zugriffssteuerung. Diese Schichten sind in Abbildung 5.2 grün dargestellt [SDM+17].

In der Verifikations- und Konfigurationsschicht formuliert der Nutzer die Anforderungen in natürlicher Sprache [SDM+17]. Eine Anforderung könnte lauten: *„Es soll erkannt werden, wenn ich nach Hause komme. Es soll aber niemand wissen, dass ich nach der Arbeit noch einkaufen war“*. Ein Modellierer mit entsprechendem Domänenwissen legt anhand dieser Anforderungen fest, welche Muster verborgen werden sollen und welche Muster erhalten bleiben müssen, um die Servicequalität zu maximieren [SDM+17]. In diesem Beispiel könnte das private Muster lauten: *„Nutzer geht nach der Arbeit einkaufen“*. Ein öffentliches Muster wäre hingegen: *„Nutzer bewegt sich auf das Haus zu“*. Dabei muss der Modellierer darauf achten, dass die Anforderungen an die Privacy nicht dem Nutzen der Anwendung widersprechen [SDM+17]. Ist es beispielsweise der Sinn der Anwendung, zu erkennen, wenn der Nutzer nach Hause kommt, darf das Muster *„Nutzer bewegt sich auf das Haus zu“*

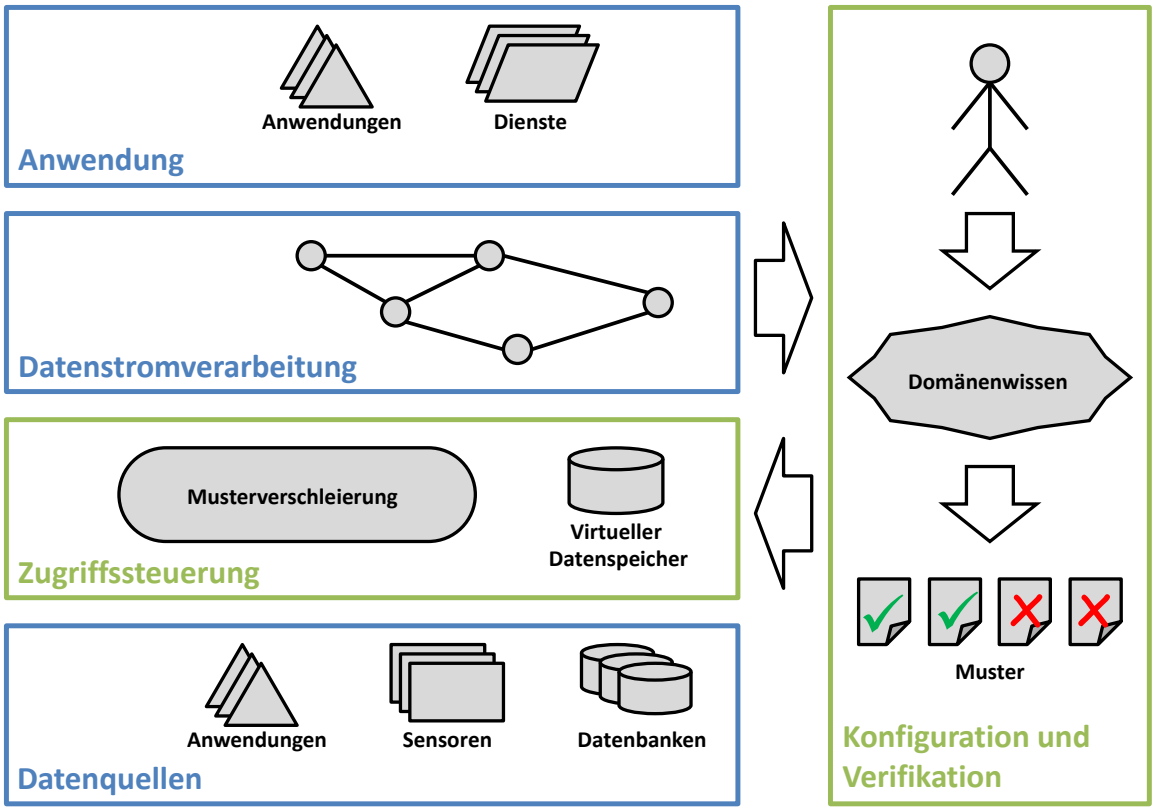

Abbildung 5.2: Architektur von PATRON, in Anlehnung an [SDM+17]

nicht verschleiert werden. Außerdem darf der Modellierer keine Muster definieren, die juristisch unzulässig sind [SDM+17].

Jedem Muster wird ein Gewicht zugeordnet. Das Gewicht gibt an, wie wichtig es ist, dass das Muster verschleiert wird bzw. erhalten bleibt. Außerdem werden Grenzwerte für die Servicequalität festgelegt. Die Grenzwerte beschreiben, wie viele Muster fälschlicherweise erkannt werden dürfen, obwohl sie im Datenstrom nicht aufgetreten sind und wie viele Muster, die tatsächlich aufgetreten sind, übersehen werden dürfen. Die Beschreibung der Muster, der Gewichte und der Servicequalität wird als Modell bezeichnet [SDM+17].

Zu den Anforderungen des Nutzers kann es mehrere Modelle geben. Beispielsweise könnte das private Muster auch *„Nutzer geht einkaufen“* lauten (ohne *„nach der Arbeit“*). Außerdem können sich die Gewichtung der Muster und die Grenzwerte für die Servicequalität unterscheiden. Aus diesen Modellen wird zufällig ein Modell ausgewählt, das für die Konfiguration des Systems verwendet wird. Die anderen Modelle werden später für die Verifikation genutzt [SDM+17].

Die Zugriffssteuerungsschicht ist dafür zuständig, die privaten Muster zu verschleiern. Dazu kommen verschiedene Techniken in Betracht. Zum einen können die Ereignisse, die Teil des Musters sind, komplett aus dem Datenstrom entfernt werden [SDM+17]. Diese ist jedoch nur dann sinnvoll, wenn die Ereignisse nicht auch Teil eines öffentlichen Musters sind, da

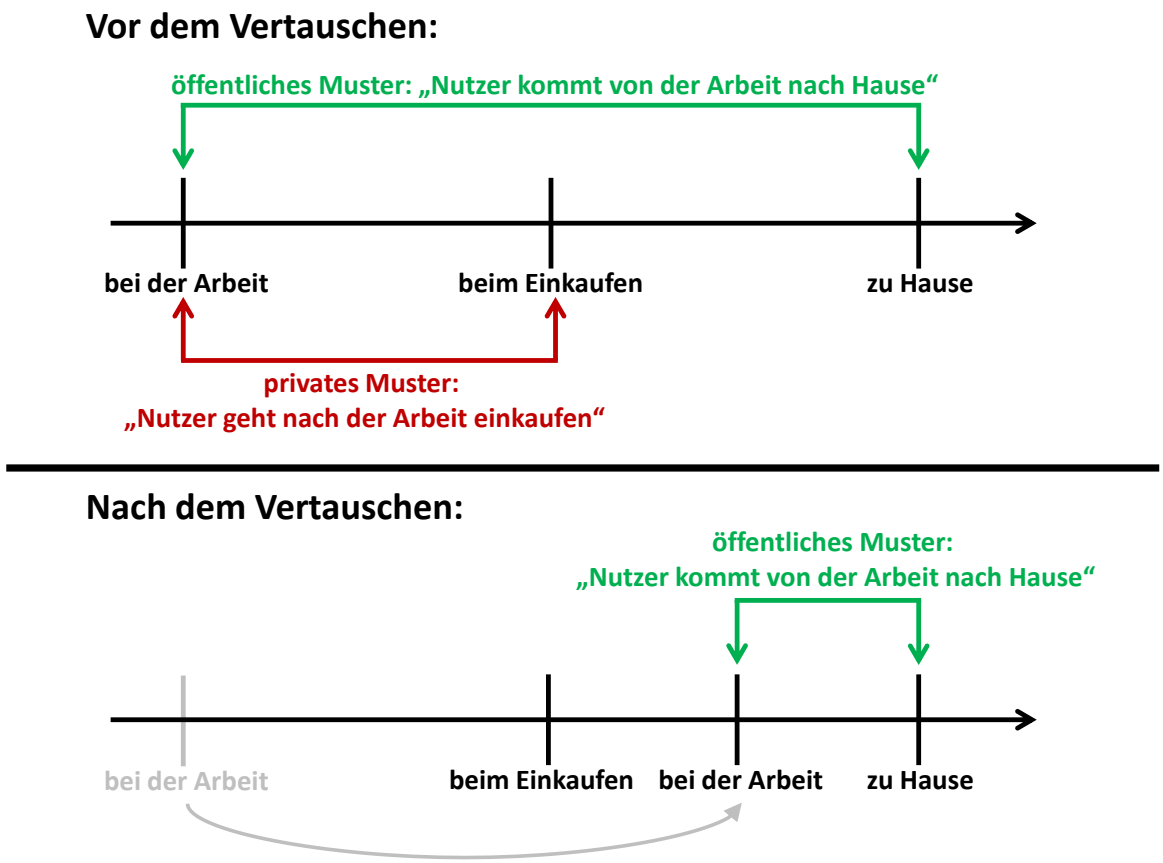

Abbildung 5.3: Vertauschen der Ereignisse zur Verschleierung des privaten Musters

ansonsten die Servicequalität vermindert wird [PDT+18]. Zum anderen kann auch die zeitliche Abfolge der Ereignisse verändert werden [SDM+17]. Soll beispielsweise wie in Abbildung 5.3 das Muster *„Nutzer geht nach der Arbeit einkaufen“* im Datenstrom auf diese Weise verschleiert werden, können die Ereignisse *„Nutzer ist bei der Arbeit“* und *„Nutzer ist beim Einkaufen“* vertauscht werden. Das Muster *„Nutzer kommt von der Arbeit nach Hause“* bleibt dabei erhalten. Eine dritte Möglichkeit ist das Verändern von Ereignissen. Beispielsweise können die GPS-Koordinaten so geändert werden, dass der Nutzer nicht mehr in der Nähe eines Supermarktes, sondern in der Nähe einer Tankstelle anhält [SDM+17].

Welche der drei Techniken zur Verschleierung privater Muster gewählt wird, wird in Abhängigkeit von der Anwendung mithilfe einer Qualitätsmetrik festgelegt. Die Qualitätsmetrik in Formel 5.1 [SDM+18] besteht aus drei Termen. Im ersten Term hat die Gesamtzahl der öffentlichen Muster, die erkannt wurden, eine positive Auswirkung auf die Qualitätsmetrik. Im zweiten Term wirkt sich die Gesamtzahl der Muster, die nicht im Datenstrom vorhanden sind, aber fälschlicherweise erkannt wurden, negativ auf die Qualitätsmetrik aus. Die Gesamtzahl der privaten Muster, die noch im Datenstrom vorhanden sind, wirkt sich im letzten Term ebenfalls negativ auf die Qualitätsmetrik aus. In Formel 5.1 haben die einzelnen Muster ein Gewicht, das angibt, wie wichtig es ist, dass das öffentliche Muster erkannt wird bzw. wie schlimm es ist, wenn das private Muster nicht verschleiert wird. Der zusätzliche

Faktor *p* im zweiten Term gibt an, wie schlimm es ist, wenn ein öffentliches Muster erkannt wird, obwohl es nicht im Datenstrom vorhanden ist [SDM+18].

$$QM = \sum_i \text{öffentliches Muster}_i * w_{Muster\ i} - \sum_j falsches\ Muster_j * w_{Muster\ j} * p - \sum_k privates\ Muster_k * w_{Muster\ k} \quad (5.1)$$

Um zu ermitteln, ob aus dem Datenstrom private Muster abgeleitet werden können, müssen sowohl der aktuelle Datenstrom, als auch historische Daten betrachtet werden [SDM+18]. Der aktuelle Datenstrom enthält beispielsweise die letzte Position des Nutzers. Zu den historischen Daten gehören alle zuvor ermittelten Positionen, also der Fahrweg des Nutzers. Dazu werden die Daten aus den Quellsystemen in virtuellen Datenspeichern gespeichert. Um die Daten in diesen virtuellen Datenspeichern zu schützen, werden die Daten verschlüsselt gespeichert. Alle Anfragen an den virtuellen Datenspeicher werden so umgeschrieben, dass keine privaten Muster preisgegeben werden [SDM+17]. Dazu kann durch das Hinzufügen von Selektionsoperatoren der Umfang der zurückgegebenen Tupel reduziert werden. Das Hinzufügen von Projektionsoperatoren schränkt die Anzahl der zurückgegebenen Attribute ein. Auch durch Filteroperatoren oder durch das Verändern von Zeitstempeln können private Muster verschleiert werden [SDM+18].

Nachdem die privaten Muster verschleiert wurden, kann der Datenstrom von der Anwendungsschicht genutzt werden. Zusätzlich wird der Datenstrom auch an die Verifikations- und Konfigurationsschicht geschickt, um die Konfiguration des Systems zu überprüfen. Dort werden die Daten mit dem Ergebnis eines simulierten Laufs mit den zuvor zurückgestellten Modellen verglichen. Das System bewertet damit, ob die verwendete Konfiguration alle Privacy-Anforderungen erfüllt. Es wird außerdem sichergestellt, dass die gewählte Konfiguration nicht zu restriktiv ist, also die Servicequalität zu stark beeinträchtigt [SAB+18].

5.3 PMP

Heutige Smartphones verfügen über eine Vielzahl von Funktionen und Diensten. Neben dem Nutzen, der sich für den Nutzer aus diesen Funktionen ergibt, ermöglichen diese Funktionen es auch, eine Vielzahl persönlicher Daten über einen Nutzer zu sammeln [SWV14]. Eine besondere Gefahr stellt dabei Software von Drittanbietern, sogenannte *Apps*, dar, die meist über App Stores bezogen wird. Um die Privacy des Nutzers sicherzustellen, ist ein Berechtigungssystem notwendig, das den Zugriff der Apps auf die persönlichen Daten des Nutzers wie beispielsweise Kontaktdaten oder Bilder sowie auf Dienste wie die Positionsbestimmung per GPS einschränkt. Dabei dürfen die Berechtigungen der App jedoch nicht so weit eingeschränkt werden, dass die App die erwünschte Dienstleistung nicht mehr erbrin-

gen kann. Beispielsweise funktioniert eine App zur Navigation nur, wenn sie die Berechtigung hat, die aktuelle Position des Nutzers zu bestimmen [Sta13].

Barrera und van Oorschot [BO11] beschreiben drei Strategien, die von den Herstellern der Mobilplattformen zum Schutz vor Schadsoftware eingesetzt werden. Beim *Walled-Garden-Modell* entscheidet der Plattformanbieter, welche Apps installiert werden dürfen. Somit trägt der Plattformanbieter auch die Verantwortung für den Schutz des Nutzers vor Schadsoftware. Das *Guardian-Modell* sieht vor, dass nicht der Plattformanbieter, sondern eine andere, vertrauenswürdige Instanz entscheidet, welche Apps verfügbar sind. Diese vertrauenswürdige Instanz kann beispielsweise der Administrator in einem Unternehmen sein. Das *User-control-Modell* bietet dem Nutzer am meisten Freiheit. Der Nutzer kann bei diesem Modell selbst entscheiden, welche Apps installiert werden. Um diese Entscheidung treffen zu können, muss der Nutzer über die notwendigen Informationen bezüglich der Datennutzung der App verfügen und die Gefahren, die sich daraus für seine Privacy ergeben, abschätzen können [Sta13]. Viele Nutzer sind dazu jedoch nicht in der Lage [FHE+12].

Die Privacy Management Platform (PMP) [Sta13] ist ein kontextsensitives und absturzsicheres Berechtigungssystem für Smartphones. Das bedeutet, dass bei der Vergabe der Berechtigungen auch festgelegt werden kann, dass die Berechtigungen nur an bestimmten Orten und nur zu bestimmten Zeiten gelten sollen. Wenn bestimmte Berechtigungen nicht erteilt werden, stürzt die App nicht ab. Stattdessen werden nur diejenigen Komponenten der App, die die verweigerte Berechtigung benötigen, deaktiviert. Die anderen Komponenten arbeiten ohne Beeinträchtigung weiter. Wird beispielsweise einer Karten-App die Berechtigung zur Ermittlung der aktuellen Position verwehrt, ist es trotzdem noch möglich, nach einer bestimmten Adresse zu suchen und diese anzuzeigen [SM13]. Außerdem ist es möglich, der App statt den korrekten Daten randomisierte Daten unterzuschieben. Der Nutzer wird dabei stets über die Konsequenzen seiner Entscheidungen informiert. Es ist auch zur Laufzeit möglich Berechtigungen zu ändern [Sta13].

Abbildung 5.4 zeigt das Berechtigungsrichtlinienmodell der PMP. Auf der linken Seite sind die potenziellen Gefahrenquellen, also die App und ihre Service Features zu sehen. Ein *Service Feature* kapselt die Funktionen, die die App anbietet. Ein mögliches Service Feature einer Navigations-App wäre beispielsweise die Funktion *„Navigation per GPS“*. Ein anders Service Feature wäre *„Anzeigen einer Adresse auf der Karte“* [SM13]. Service Features können einzeln aktiviert und deaktiviert werden. Der Nutzer ordnet nicht der App, sondern den einzelnen Service Features Berechtigungen zu. Durch das Aktivieren bzw. das Deaktivieren der Service Features können der App Berechtigungen erteilt und entzogen werden, ohne dass die App abstürzt [Sta13].

Unter einer *Ressource* versteht man Schnittstellen zu geschützten Daten oder Systemfunktionen wie beispielsweise den Zugriff auf die Kontakte des Nutzers oder die Kamera. Der Nutzer kann entscheiden, ob ein Service Feature die korrekten Daten erhält oder ob die Daten nur aggregiert zur Verfügung gestellt werden. Der Nutzer kann auch festlegen, dass das Service Feature lediglich randomisierte Daten erhält. Ressourcen sind eigenständige Diens-

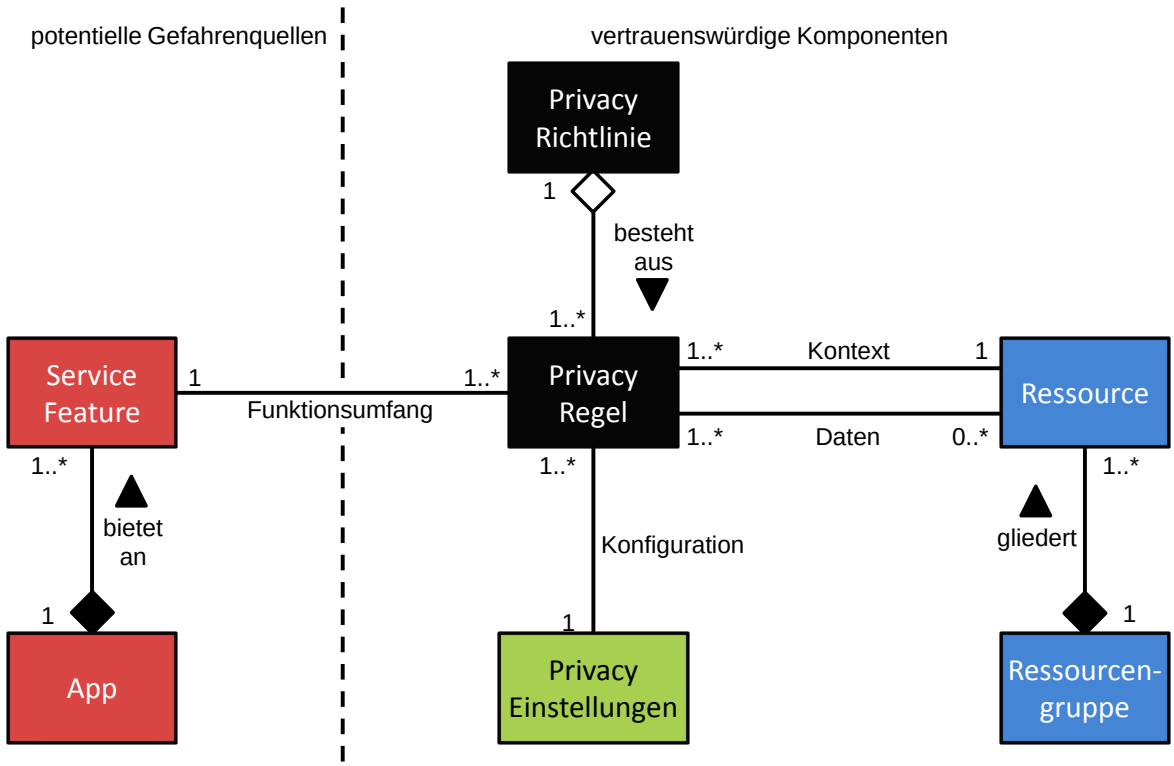

Abbildung 5.4: Berechtigungsrichtlinienmodell der PMP, aus dem Englischen übersetzt aus [Sta13]

te, die von der PMP automatisch installiert werden. Ressourcengruppen sind Gruppen von Ressourcen, die inhaltlich zusammengehören [Sta13].

Welche *Privacy-Einstellungen* es gibt, definiert die jeweilige Ressource. Der Nutzer legt fest, wie die Privacy-Einstellungen konfiguriert werden. Dabei kann der Nutzer wie zuvor beschrieben entscheiden, ob ein Service Feature die korrekten Daten, eine aggregierte Version der Daten oder randomisierte Daten erhält [Sta13]. Welche Einstellung sinnvoll ist, hängt vom jeweiligen Anwendungsfall ab. Um den Nutzer zu einem bestimmten Ort zu navigieren, ist die genaue Position zwingend erforderlich. Möchte der Nutzer sich jedoch nur alle Tankstellen in einem bestimmten Umkreis anzeigen lassen, reicht eine auf 100 Meter genaue Position aus [SM13].

Wie in Abbildung 5.4 zu sehen, verbindet die *Privacy-Regel* die Service Features, die Ressourcen und die Privacy-Einstellungen miteinander. Eine Privacy-Regel besteht aus einem Service Feature, das eine Ressource verwenden will und dabei eine Berechtigung benötigt, die durch die Privacy-Einstellungen definiert wird. Die Regel kann auch festlegten, dass die Berechtigung nur zu einer bestimmten Zeit oder nur an einem bestimmten Ort gilt [Sta13]. Die *Privacy-Richtlinie* ist die Menge aller Privacy-Regeln [SM14].

5.4 ACCESSORS

Die PMP eignet sich vor allem zum Schutz private Daten auf Mobilplattformen. Für ein Smart Home, in dem es eine Vielzahl unterschiedlicher Sensoren gibt, ist jedoch ein anderer Ansatz notwendig. Der Grund dafür ist, dass die Nutzer zwar wissen, welche Daten sie geheim halten wollen, jedoch nicht, welche Sensoren diese Daten erfassen. Statt den Zugriff auf bestimmte Sensoren einzeln zu reglementieren, möchten die Nutzer den Zugriff auf bestimmte Daten reglementieren, unabhängig davon, von welchen Sensoren die Daten stammen [SM18].

ACCESSORS [SM18] ist eine Erweiterung des Berechtigungsrichtlinienmodels der PMP, mit der der Nutzer festlegen kann, welche Arten von Daten in der Anwendung verfügbar sein sollen. Dazu muss das Berechtigungsrichtlinienmodell wissen, welche Sensoren welche Arten von Daten erzeugen. Eine mögliche Datenart wäre das Raumklima, zu dem die Daten der Temperatursensoren und der Luftfeuchtigkeitssensoren gehören. ACCESSORS ermöglicht eine feingranulare Definition von Privacy-Regeln. Beispielsweise ist es möglich, den Zugriff auf den Standort nicht nur generell zu gewähren oder zu verweigern, sondern auch den Zugriff auf den Standort nur auf 100 Meter genau zu erlauben. Außerdem ist es möglich, Regeln kontextsensitiv zu aktivieren oder zu deaktivieren. Beispielsweise kann eine Regel nur zu einer bestimmten Tageszeit oder an einem bestimmten Standort aktiviert werden. ACCESSORS ist so aufgebaut, dass es dynamisch um neue Sensortypen erweitert werden kann [SM18].

Abbildung 5.5 zeigt die Komponenten von ACCESSORS. Der Regelkern ähnelt dem Berechtigungsrichtlinienmodell der PMP in Abbildung 5.4. In der PMP legt eine Privacy-Regel fest, unter welchen Bedingungen ein Service Feature auf eine Ressource zugreifen darf. In ACCESSORS definiert ein anforderndes Objekt ein Service Feature. Eine Privacy-Regel gewährt unter bestimmten Bedingungen Zugriff auf einen Datenproduzenten oder einen Datenkonsumenten. Es können bestimmte Kontexte definiert werden, unter denen die Privacy-Regel aktiviert bzw. deaktiviert wird [SM18].

Ein anforderndes Objekt kann eine Anwendung, ein Smart Device oder der Nutzer sein. Das anfordernde Objekt kann ein oder mehrere Service Features definieren, die Zugriff auf geschützte Daten benötigen. Wie in der PMP kapselt ein Service Feature eine Funktion, die das anfordernde Objekt anbietet. Auf diese Weise ist die Berechtigung nicht an das anfordernde Objekt, sondern an das Service Feature geknüpft. Der Nutzer kann entscheiden, ob er die benötigten Berechtigungen erteilen und das Service Feature aktivieren möchte. Verweigert er die Berechtigungen, wird das Service Feature deaktiviert. Durch die Abstraktion ist es möglich, bei Bedarf weitere Nutzertypen hinzuzufügen [SM18].

Eine Datenquelle kann eine Anwendung, ein Datenspeicher, ein Smart Device oder ein Sensor sein. Aus einer Datenquelle oder aus der Kombination mehrerer Datenquellen können Informationen abgeleitet werden. Beispielsweise kann aus den Daten des GPS-Sensors der Standort des Nutzers abgeleitet werden. Jeder Datenproduzent ist mit einer bestimmten Information verbunden. Durch die Abstraktion sind die Privacy-Regeln im Regelkern unab-

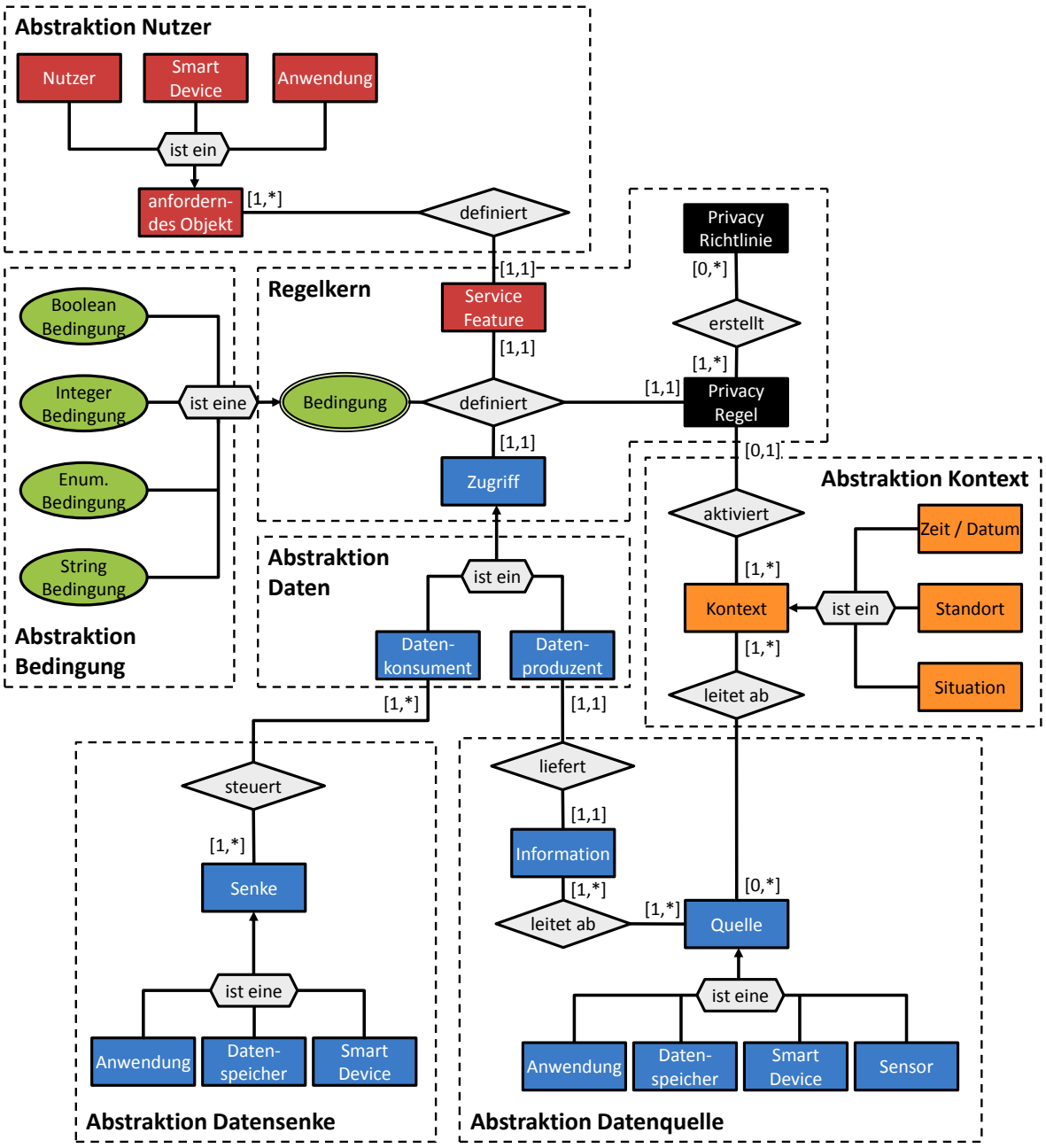

Abbildung 5.5: Berechtigungsmodell in ACCESSORS, aus dem Englischen übersetzt aus [SM18]

hängig von einer konkreten Technologie. Es spielt also keine Rollen, ob der Standort per GPS oder per WLAN ermittelt wird [SM18].

Jeder Datenkonsument ist mit einer oder mehreren Datensenken verbunden. Eine Datensenke kann eine Anwendung, ein Datenspeicher oder ein Smart Device sein. Der Nutzer kann Privacy-Regeln definieren, die festlegen, unter welchen Umständen Daten an einen Daten-

konsumenten geliefert werden. Durch die Abstraktion ist es möglich, weitere Datensenken hinzuzufügen [SM18].

Eine Privacy-Regel kann mit einem Kontext verknüpft werden, der festlegt, unter welchen Bedingungen die Regel aktiviert ist. Eine Privacy-Regel kann beispielsweise nur zu einer bestimmten Tageszeit oder nur an einem bestimmten Ort aktiviert sein. Es ist auch möglich, komplexere Situationen für die Aktivierung der Privacy-Regel zu definieren. Zur Modellierung dieser Situationen werden die Daten aus den Datenquellen verwendet [SM18]. Beispielsweise kann festgelegt werden, dass eine Privacy-Regel nur dann aktiv ist, wenn die Raumtemperatur über 20 Grad Celsius liegt.

Die Bedingungen, unter denen der Zugriff gewährt wird, können auf unterschiedliche Weisen definiert werden. Bei booleschen Bedingungen ist es nur möglich, den Zugriff zuzulassen oder zu verweigern. Abhängig von der Art der Daten kann es noch weitere Bedingungen geben. Mit Integer-Bedingungen ist es möglich, eine obere und untere Grenze festzulegen. Beispielsweise kann festgelegt werden, dass der Standort nur auf 100 Meter genau sein darf. Enumeration-Bedingungen legen mehrere zulässige Möglichkeiten fest. Zur Ermittlung des Standorts des Nutzers könnte beispielsweise der Standort des Smartphones oder der Standort des Fahrzeugs verwendet werden. Mithilfe von String-Bedingungen können textuelle Bedingungen definiert werden. Beispielsweise kann definiert werden, dass nur Daten von Smart Devices verwendet werden dürfen, die sich in einem bestimmten IP-Adressbereich befinden [SM18].

5.5 AVARE

AVARE ist ein Ansatz zur zentralen Verwaltung verteilter Privacy-Einstellungen. Dabei legt der Nutzer die Privacy-Einstellungen einmalig an zentraler Stelle fest. Diese Einstellungen werden anschließend an alle Geräte des Nutzers verteilt und dort angewendet. AVARE ist unabhängig von einem bestimmten Betriebssystem und kann auf unterschiedlichen Geräteklassen eingesetzt werden. So kann AVARE sowohl die Anwendungen auf einem Smartphone, als auch die Sensoren in einem Smart Home reglementieren [AOP+17].

Wie in Abbildung 5.6 zu sehen, basiert AVARE auf einer Client-Server-Architektur. Die Privacy-Einstellungen des Nutzers werden auf dem Server zentral und symmetrisch verschlüsselt gespeichert. Durch die Verschlüsselung wird sichergestellt, dass die Privacy-Einstellungen von Außenstehenden weder gelesen noch verändert werden können. Dadurch kann sich der Server auch in einer nicht vertrauenswürdigen Umgebung wie beispielsweise der Cloud befinden. Aus den auf dem Server gespeicherten Privacy-Einstellungen ergeben sich Regeln, die den Datenfluss zwischen den Anwendungen, dem Betriebssystem, der Hardware und der Außenwelt reglementieren. Eine Regel kann einen Datenfluss filtern, die darin enthaltenen Daten durch andere Daten ersetzen oder aber den Datenfluss vollständig blockieren [ABF+18].

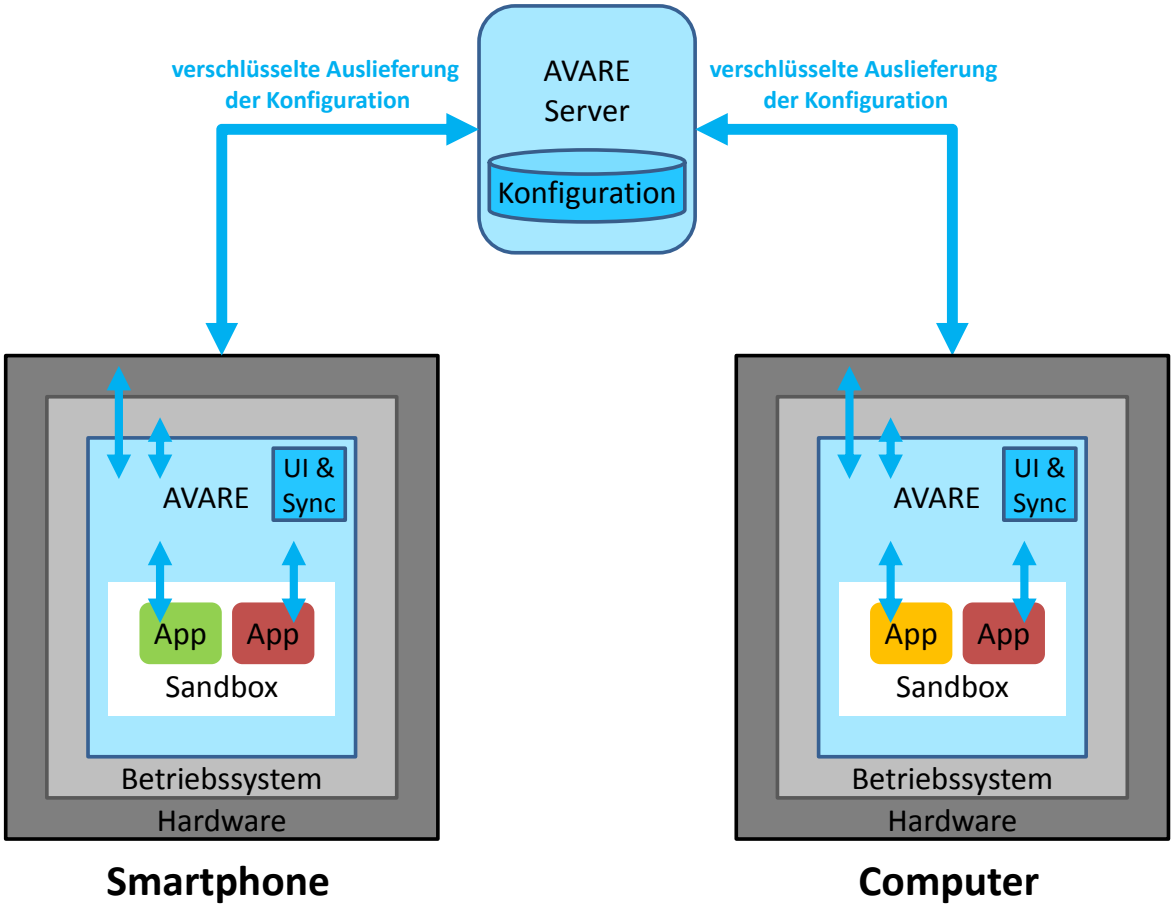

Abbildung 5.6: Arbeitsprinzip von AVARE, in Anlehnung an [AOP+17]

Zur Anwendung der Regeln auf dem Client stellt AVARE eine UI- und Synchronisationskomponente und eine Sandbox zur Verfügung. Die Synchronisationskomponente tauscht mit dem Server die Privacy-Einstellungen aus und stellt sie der AVARE-Instanz auf dem Gerät zur Verfügung. Mithilfe der UI-Komponente kann der Nutzer die Privacy-Einstellungen anpassen. Änderungen an den Privacy-Einstellungen werden mit dem Server synchronisiert und stehen dadurch allen Geräten zur Verfügung. Die Anwendungen auf den Geräten werden nicht direkt auf dem Betriebssystem ausgeführt, sondern in einer Sandbox. Auf diese Weise greifen die Anwendungen nicht direkt auf das Betriebssystem, die Hardware oder auf Sensordaten zu. Stattdessen kann AVARE den Datenfluss zwischen den Anwendungen und den Komponenten, die sich außerhalb der Sandbox befinden, überwachen und manipulieren, sofern es die Privacy-Einstellungen erfordern. Durch den Einsatz der Sandbox sind keine Änderungen am Betriebssystem oder an den Anwendungen notwendig [AEG+18].

6 Implementierung

Zur Realisierung des Konzepts wird, wie bereits in Abschnitt 4.1 diskutiert wurde, eine Privacy-Kontrolle in den Adaptern und eine Privacy-Kontrolle in der IoT-Plattform benötigt. Da die Privacy-Kontrolle auf viele verschiedene Smart Devices verteilt ist, wird eine Möglichkeit benötigt, die Konfiguration der Privacy-Einstellungen zentral zu definieren und dann auf die verschiedenen Smart Devices zu verteilen.

In Abschnitt 3.1 wurden verschiedene Privacy-Mechanismen für Smart Devices beschrieben und mit dem in Abschnitt 5.3 vorgestellten Berechtigungsrichtlinienmodell der PMP und dessen in Abschnitt 5.4 beschriebenen Erweiterung ACCESSORS verglichen. Dabei zeigte sich, dass nur ACCESSORS die Definition von feingranularen, kontextsensitiven, erweiterbaren und datenzentrierten Privacy-Regeln zur Reglementierung der Datenquellen und Datensenken erlaubt. Deshalb wird für die Privacy-Kontrollen an den Adaptern in den Smart Devices ACCESSORS verwendet.

Die Privacy-Kontrolle auf der IoT-Plattform muss private Muster aus verschiedenen Quellen verschleiern und öffentliche Muster erhalten können. Von den in Abschnitt 3.2 verglichenen Privacy-Mechanismen für Datenstromsysteme bietet lediglich PATRON eine musterbasierte Zugriffskontrolle. Die anderen Privacy-Mechanismen arbeiten attributbasiert und filtern dadurch unnötig viele Daten aus dem Datenstrom heraus. Deshalb wird die Privacy-Kontrolle auf der IoT-Plattform mithilfe von PATRON implementiert.

Die Verteilung der Privacy-Einstellungen auf die Smart Devices erfolgt durch ein Verfahren, das an AVARE angelehnt ist. Die Privacy-Einstellungen werden dabei zentral auf der IoT-Plattform gespeichert und von dort an die Smart Devices ausgeliefert. Die Smart Devices besitzen eine Synchronisationskomponente, die die Privacy-Einstellungen des Smart Devices mit den Privacy-Einstellungen auf der IoT-Plattform synchronisiert.

Im folgenden Abschnitt 6.1 wird untersucht, inwiefern MIALinx, PATRON, ACCESSORS und AVARE gemeinsam die in Abschnitt 2.2 beschriebenen Anforderungen erfüllen. Anschließend wird in Abschnitt 6.2 beschrieben, wie die Privacy-Kontrolle in den Smart Devices realisiert wird. Danach geht es in Abschnitt 6.3 um die Zugriffssteuerung, die auf der IoT-Plattform stattfindet. Zum Schluss wird in Abschnitt 6.4 die Verteilung der Konfiguration mithilfe des an AVARE angelehnten Verfahrens beschrieben.

6.1 Evaluation der Teilsysteme

In diesem Abschnitt wird untersucht, inwiefern die in Abschnitt 5 vorgestellten Systeme die in Abschnitt 2.2 beschriebenen Anforderungen erfüllen. Die Anforderungen in Tabelle 6.1 sind in vier Gruppen unterteilt. Zur ersten Gruppe gehören die Anforderungen A1 und A2. Hier geht es um die Definition von Wenn-Dann-Regeln und die Echtzeitverarbeitung der Sensordaten durch diese Regeln. Diese Anforderungen werden von MIALinx erfüllt. In MIALinx ist es möglich, Wenn-Dann-Regeln zu definieren und die ankommenden Sensordaten in Echtzeit zu verarbeiten. PATRON ermöglicht zwar auch die Verarbeitung der Sensordaten in Echtzeit, allerdings geht es bei PATRON um die Verschleierung privater Muster und nicht um die Definition von Wenn-Dann-Regeln. Die PMP, ACCESSORS und AVARE erlauben weder die Definition von Wenn-Dann-Regeln zur Verarbeitung der Daten, noch die Verarbeitung von Datenströmen aus mehreren Quellen in Echtzeit.

Die zweite Gruppe in Tabelle 6.1 umfasst die Anforderungen A3 bis A6 und bezieht sich auf die Verschleierung privater Muster. In MIALinx können im Wenn-Bestandteil der Regeln Muster in Form von Bedingungen erkannt werden. Ein Muster ist in diesem Fall eine bestimmte Sequenz an Ereignissen, die als Bedingung modelliert werden kann. MIALinx bietet jedoch keine Möglichkeit, Muster zu verschleiern. PATRON erfüllt hingegen alle Anforderungen in dieser Gruppe. Mit PATRON ist es möglich, öffentliche und private Muster zu definieren. PATRON untersucht die ankommenden Datenströme auf private Muster und verschleiert diese. Dabei werden die öffentlichen Muster erkannt und bleiben erhalten. Durch die Verschleierung werden keine False Positives hinzugefügt. Die PMP, ACCESSORS und AVARE erlauben keine Verschleierung privater Muster.

In der dritten Gruppe in Tabelle 6.1 geht es um die Reglementierung der Sensoren und Aktuatoren. Dazu zählen die Anforderungen A7 bis A13. MIALinx und PATRON erfüllen keine dieser Anforderung, da die beiden Systeme der Verarbeitung von Wenn-Dann-Regeln bzw. der Verschleierung von Mustern und nicht der Reglementierung einzelner Sensoren oder Aktuatoren dienen. Einzige Ausnahme ist Anforderung A11, die von MIALinx erfüllt wird, da es in MIALinx ohne großen Aufwand möglich ist, neue Sensoren oder Aktuatoren hinzuzufügen. Die PMP erfüllt hingegen alle Anforderungen in dieser Gruppe außer A13. Mit dem Berechtigungsrichtlinienmodell der PMP ist es möglich, Sensoren und Aktuatoren zu reglementieren. Die Reglementierung kann feingranular und kontextsensitiv erfolgen. Neue Sensoren und Aktuatoren können ohne großen Aufwand hinzugefügt werden, da die PMP einfach erweiterbar ist. Es ist auch möglich, die Übertragungsfrequenz der Sensordaten einzuschränken. Allerdings ist es mit dem Berechtigungsrichtlinienmodell der PMP nicht möglich, datenzentrierte Privacy-Regeln zu definieren. Diese Anforderung wird jedoch von ACCESSORS erfüllt, das eine Erweiterung der PMP darstellt. ACCESSORS erfüllt alle Anforderungen, die auch PMP erfüllt und ermöglicht zusätzlich die Definition datenzentrierter Privacy-Regeln.

Die letzte Gruppe in Tabelle 6.1 besteht nur aus Anforderung A14. Hier geht es darum, die Privacy-Einstellungen an einer zentralen Stelle zu definieren und dann an die verschiedenen

Anforderung	MIALinx	PATRON	PMP	ACCES-SORS	AVARE
A1 Wenn-Dann-Regeln	✓	✗	✗	✗	✗
A2 Echtzeitverarbeitung	✓	✓	✗	✗	✗
A3 Öffentliche Muster	✓	✓	✗	✗	✗
A4 Private Muster	✗	✓	✗	✗	✗
A5 Muster erhalten	n/a	✓	n/a	n/a	n/a
A6 Keine False Positives	n/a	✓	n/a	n/a	n/a
A7 Sensoren reglementieren	✗	✗	✓	✓	✗
A8 Aktuatoren reglementieren	✗	✗	✓	✓	✗
A9 Feingranularität	✗	✗	✓	✓	✗
A10 Kontextsensitivität	✗	✗	✓	✓	✗
A11 Erweiterbarkeit	✓	n/a	✓	✓	✗
A12 Datenzentriertheit	✗	✗	✗	✓	✗
A13 Übertragungsfrequenz	✗	✗	✓	✓	✗
A14 Zentrale Konfiguration	✗	✗	✗	✗	✓

Tabelle 6.1: Evaluation der Teilsysteme

Smart Devices auszuliefern. Diese Anforderung wird nur von AVARE erfüllt, da nur AVARE die zentrale Speicherung und anschließende Verteilung von Privacy-Regeln in einer heterogenen Umgebung ermöglicht.

Die Evaluation der in Abschnitt 5 vorgestellten Systeme zeigt, dass keines der Systeme die in Abschnitt 2.2 beschriebenen Anforderungen vollständig erfüllt. Wie in Tabelle 6.1 zu sehen, können jedoch durch Kombination von MIALinx, PATRON, ACCESSORS und AVARE alle genannten Anforderungen erfüllt werden.

6.2 Privacy-Kontrolle in den Smart Devices

Um Datenquellen und Datensenken zu reglementieren, wird das Berechtigungsrichtlinienmodell ACCESSORS aus Abschnitt 5.4 verwendet. Abbildung 6.1 zeigt, wie ACCESSORS mit den Privacy-Filtern in den Datenquellen und Datensenken zusammenarbeitet. Um die einzelnen Komponenten besser referenzieren zu können, sind die Komponenten in Abbildung 6.1 mit Kleinbuchstaben von a bis j beschriftet. Die Privacy-Filter in den Adaptern (a) sind die Datenquellen (b), die durch die Privacy-Regeln (e) reglementiert werden sollen. Anders als in Abbildung 6.1 dargestellt, gibt es mehr als nur einen Privacy-Filter. Aus diesen Quellen (b) können Informationen (c) abgeleitet werden. Aus den Daten des GPS-Sensors kann beispielsweise der Standort des Nutzers abgeleitet werden. Jede Information (c) ist mit einem bestimmten Datenproduzenten (d) verbunden. Eine Privacy-Regel (e) kann nun festlegen, unter welchen Bedingungen (j) und in welchem Kontext (i) die Daten des Datenproduzenten (d) weitergegeben werden dürfen. Auf die gleiche Weise werden Privacy-Regeln (e) für die Privacy-Filter in den Adaptern der Datensenken (f) definiert. Eine Senke (g) kann dabei ein beliebiger Privacy-Filter (f) sein. Anders als in Abbildung 6.1 dargestellt, gibt es mehr als nur einen Privacy-Filter für die Datensenken. Ein Datenkonsument (h) ist mit einer oder mehreren Senken (g) verbunden, die von ihm gesteuert werden. Eine Senke (g) kann wiederum von einem oder mehreren Datenkonsumenten (h) gesteuert werden. Der Nutzer kann durch Privacy-Regeln (e) festlegen, unter welchen Bedingungen (j) und in welchem Kontext (i) der Datenkonsument (h) Befehle verarbeiten darf.

Ein Kontext (i) beschreibt eine bestimmte Situation, in der die Privacy-Regeln (e) gültig sind. Eine Privacy-Regel (e) kann beispielsweise nur in einem bestimmten Zeitraum oder nur an einem bestimmten Standort gelten. Es können auch komplexere Situationen modelliert werden. Die dazu benötigten Informationen werden aus den Datenquellen (b) abgeleitet. Beispielsweise kann eine Privacy-Regel (e) nur dann gelten, wenn der Nutzer zu Hause ist. Die Bedingungen (j), unter denen Daten weitergeleitet oder Befehle reglementiert werden, können auf unterschiedliche Arten modelliert werden. Wie in Abschnitt 5.4 beschrieben, können dazu boolesche Bedingungen, Integer-Bedingungen, Enumeration-Bedingungen oder String-Bedingungen genutzt werden. Die Reduzierung der Genauigkeit des Standorts auf 100 Meter kann beispielsweise als Integer-Bedingungen modelliert werden. Die Filterung bestimmter Attribute aus einem Adressbuch kann hingegen mit einer Enumeration-Bedingung erfolgen. Dazu werden die Attribute aufgezählt, die weitergegeben werden dürfen.

Im Gegensatz zu dem in Abschnitt 5.4 vorgestellten ACCESSORS gibt es hier keine Service Features. Das liegt daran, dass alle Daten, die weitergegeben werden, an die IoT-Plattform weitergegeben werden. Daher gibt es nur ein einziges Service Feature. Die Modellierung eines einzigen Service Features würde das Modell unnötig verkomplizieren und darüber hinaus keinen weiteren Vorteil bringen. Soll festgelegt werden, dass die Daten einer Datenquelle nur an eine bestimmte Datensenke weitergegeben werden dürfen, wird dazu eine weitere Privacy-Regel definiert, die die Weitergabe der Daten im Privacy-Filter für die Datensenke (f) einschränkt. Eine Filterung der Daten im Privacy-Filter für die Datenquelle (a) ist

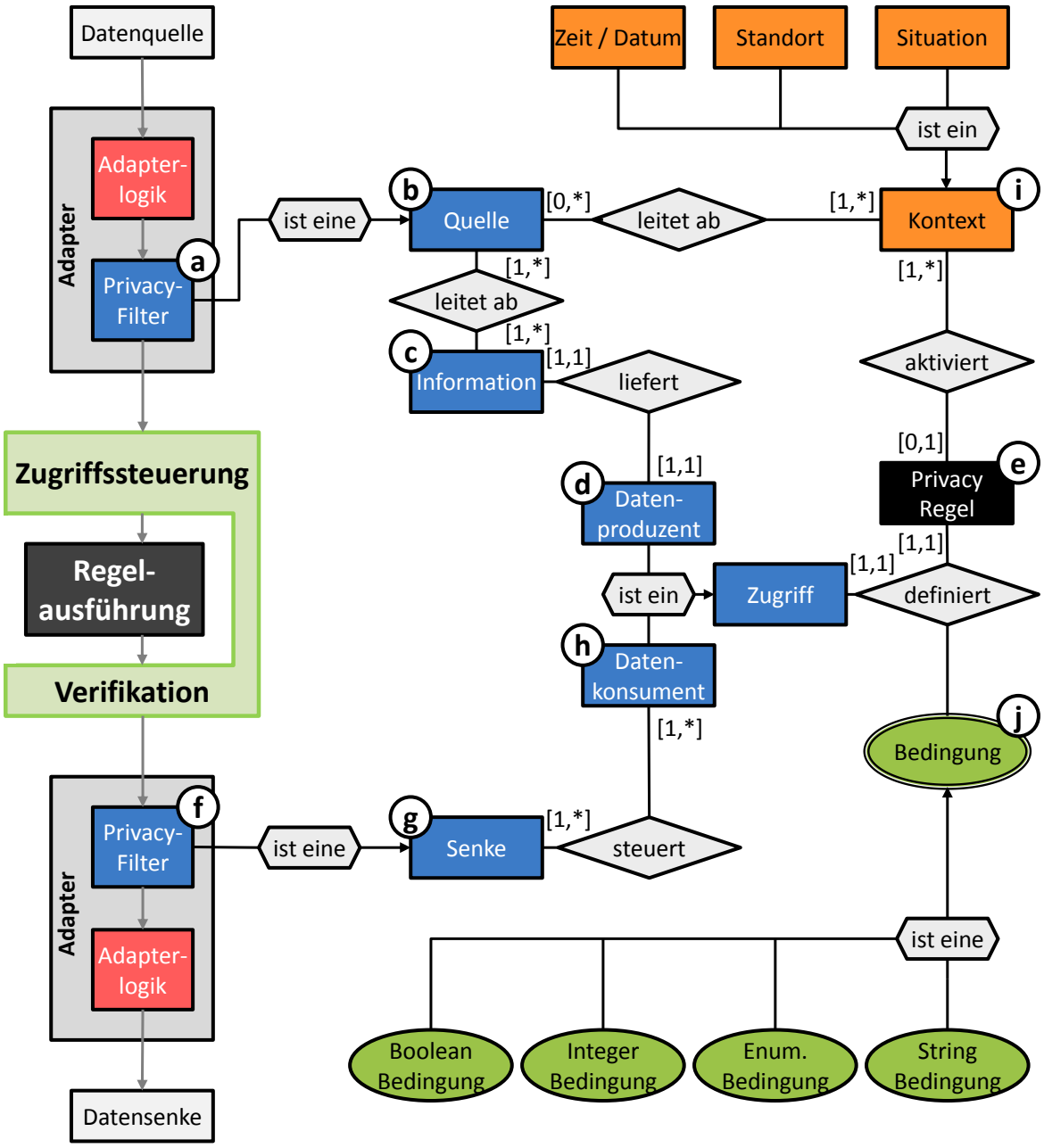

Abbildung 6.1: Privacy-Kontrolle in den Smart Devices durch ACCESSORS

nicht möglich, da der Privacy-Filter für die Datenquelle nicht weiß, welche Datensenken es gibt und an welche dieser Datensenken die IoT-Plattform die Daten weiterleiten wird.

Der Vorteil von ACCESSORS besteht in der Abstraktion der Datenquellen und Datensenken. Auf diese Weise sind die Regeln unabhängig von einer konkreten Technologie. Der Standort des Nutzers kann beispielsweise mithilfe des GPS-Sensors im Smartphone oder mithilfe des GPS-Sensors des Autos ermittelt werden. Die Privacy-Regel, die den Zugriff

auf die Standortdaten einschränkt, gilt jedoch automatisch für beide Technologien. Es wird also nicht der Zugriff auf einen bestimmten Sensor reglementiert, sondern der Zugriff auf die Daten, die ein Sensor zur Verfügung stellt. Die Abstraktion erlaubt es außerdem, ohne großen Aufwand neue Datenquellen und Datensenken hinzuzufügen. Durch die Abstraktion des Kontexts können zudem beliebig komplexe Situationen modelliert werden, in denen die Privacy-Regel aktiviert werden sollen. Die Abstraktion der Bedingung ermöglicht es, feingranulare Bedingungen zu formulieren. Anstatt also den Zugriff auf die Daten vollständig zu blockieren, können feingranulare Bedingungen formuliert werden, die definieren, unter welchen Umständen auf die Daten zugegriffen werden darf.

6.3 Privacy-Kontrolle in der IoT-Plattform

Die Privacy-Kontrolle auf der IoT-Plattform wird mithilfe von PATRON implementiert. Abbildung 6.2 zeigt, wie PATRON in die Zugriffssteuerung auf der IoT-Plattform integriert wird. Die Adapter der Datenquellen liefen die Datenströme, in denen nach privaten Mustern gesucht werden soll. Um die privaten Muster verschleiern zu können, muss zunächst ermittelt werden, welche Verschleierungstechnik für den Datenstrom die besten Ergebnisse liefert. Dazu wird für jedes private Muster eine Simulation durchgeführt. In der Simulation wird der Datenstrom mithilfe der Verschleierungstechniken *blockieren*, *vertauschen* und *verändern* verschleiert.

Zur Verschleierung des privaten Musters kommen die in Abschnitt 5.2 beschriebenen Verschleierungstechniken *blockieren*, *vertauschen* und *verändern* in Betracht. Das Blockieren des Standorts macht jedoch keinen Sinn, da dadurch beispielsweise das öffentliche das Muster *„Nutzer kommt nach Hause“* nicht mehr erkannt werden kann. Das Vertauschen der Standortdaten ist ebenfalls problematisch, da durch eine geänderte zeitliche Reihenfolge der Standortdaten der Eindruck entstehen könnte, dass sich der Nutzer von dem Haus weg bewegt, obwohl sich der Nutzer auf das Haus zubewegt. Auch in diesem Fall würde das Muster *„Nutzer kommt nach Hause“* zerstört werden. Die dritte Verschleierungstechnik ist das Verändern der Standortdaten. Hält der Nutzer tatsächlich in der Nähe eines Supermarktes an, um einkaufen zu gehen, kann der Standort des Nutzers an eine andere Position verschoben werden. Beispielsweise können alle Standorte des Nutzers in einem 200-Meter-Umkreis um den Supermarkt an einen zufälligen anderen Ort außerhalb des Umkreises verschoben werden. Ist dieser Ort nicht allzu weit von dem tatsächlichen Standort des Nutzers entfernt, kann das Muster *„Nutzer kommt nach Hause“* weiterhin erkannt werden, während das Muster *„Nutzer kauft gerade ein“* verschleiert wird.

Nach der Durchführung der Simulation mit den drei Verschleierungstechniken wird für jede Verschleierungstechnik mithilfe einer Metrik berechnet, wie erfolgreich die Verschleierung war. Dazu wird die Qualitätsmetrik in Formel 5.1 verwendet, die berücksichtigt, wie viele öffentliche und private Muster noch vorhanden sind und wie viele False Positives hinzugefügt wurden. Die Verschleierungstechnik, die die meisten öffentlichen Muster erhält, die

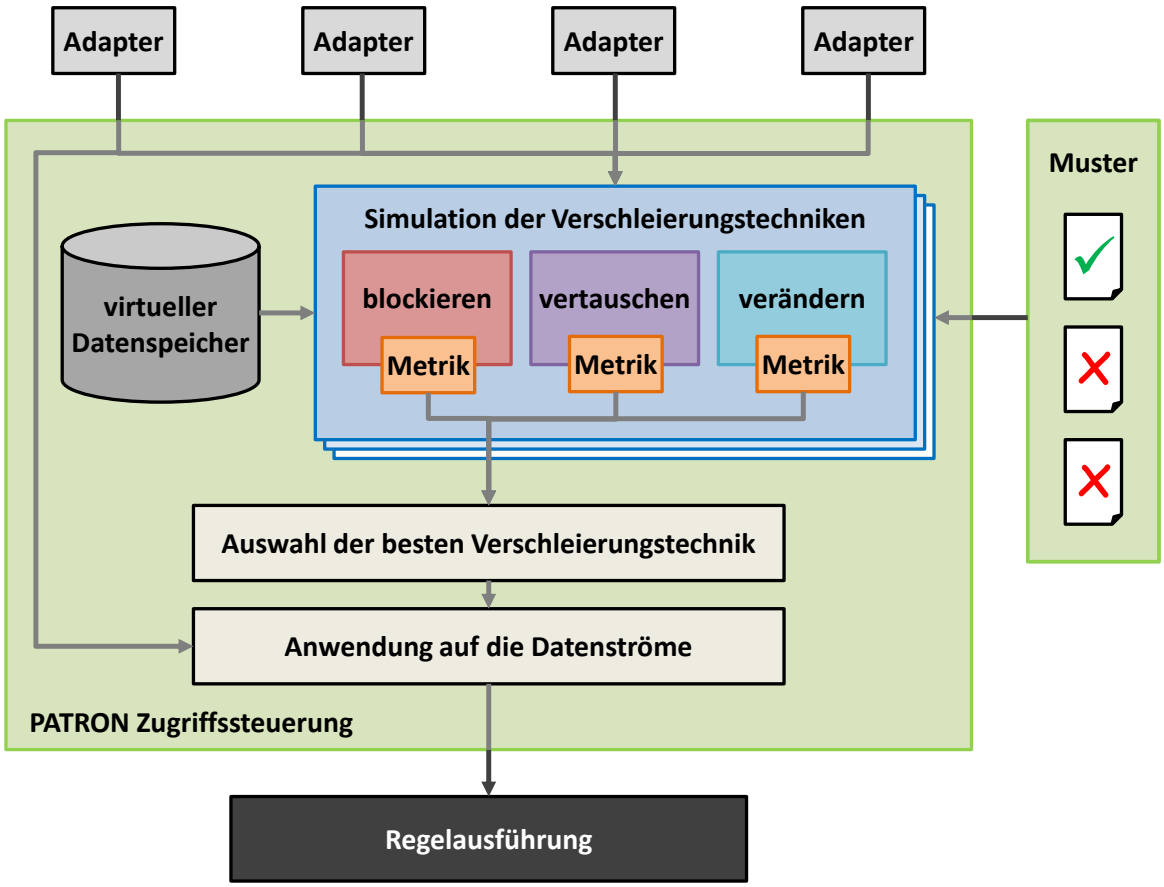

Abbildung 6.2: Funktionsweise der Zugriffssteuerung, vgl. [SDM+17]

meisten privaten Muster entfernt und am wenigsten False Positives hinzufügt, weist den höchsten Metrikwert auf.

Bei der Simulation werden dabei nicht nur die neuen Daten berücksichtigt, sondern auch die Daten, die bereits in der Vergangenheit verarbeitet wurden. Auf diese Weise ist sichergestellt, dass auch aus der Kombination von Daten, die über einen längeren Zeitraum verarbeitet werden, keine privaten Muster preisgegeben werden. Die Daten, die in der Vergangenheit verarbeitet wurden, werden in einem virtuellen Datenspeicher gespeichert. Die Daten in diesem Datenspeicher werden verschlüsselt gespeichert, damit nur die Zugriffssteuerung Zugriff darauf hat. Nachdem die beste Verschleierungstechnik ermittelt wurde, wird diese auf den Datenstrom angewendet. Der resultierende Datenstrom enthält keine privaten Muster mehr und kann dadurch für die Regelausführung verwendet werden.

Die öffentlichen und privaten Mustern, die der Zugriffssteuerung zur Verfügung gestellt werden, werden auf der IoT-Plattform gespeichert. Aus den Bedingungen im Wenn-Bestandteil der Wenn-Dann-Regeln können die öffentlichen Muster abgeleitet werden. Die Wenn-Dann-Regeln sind im Regelspeicher (Nummer 5 in Abbildung 4.3) gespeichert. Es könnte jedoch passieren, dass der Anbieter der Regelverarbeitungskomponente (Nummer 4)

den Datenstrom auf weitere Muster untersucht, die er nach außen nicht bekannt gibt, um gegenüber Wettbewerbern im Vorteil zu sein. Sind diese Muster im Regelspeicher nicht vorhanden, stuft die Zugriffssteuerung diese Muster als neutrale Muster ein. Dadurch kann es passieren, dass die Muster durch die Zugriffssteuerung (Nummer 3) entfernt werden oder dass False Positives eingeführt werden.

Im Gegensatz zu den öffentlichen Mustern können die privaten Muster nicht automatisch ermittelt werden. Deshalb müssen die privaten Muster durch den Nutzer definiert werden. Die Definition der privaten Muster durch den Nutzer ist jedoch in der Praxis sehr problematisch, da die meisten Nutzer die Gefahren für die Privacy nicht einschätzen können [Pöt08]. Das gleiche Problem besteht bei der Definition der Privacy-Regeln für die Sensoren und Aktuatoren durch den Nutzer. Deshalb kann es sinnvoll sein, dass Privacy-Experten vorab Privacy-Regeln und private Muster für verschiedene Situationen definieren. Anhand der vorhandenen Sensoren und Aktuatoren können dem Nutzer anschließend diejenigen Regeln und Muster vorgeschlagen werden, die in dem konkreten Kontext Sinn ergeben. Der Nutzer kann anschließend basierend auf der Ausgangskonfiguration seine eigenen Anpassungen vornehmen. Durch diese Vorgehensweise ist sichergestellt, dass zumindest die wichtigsten Risiken für die Privacy des Nutzers adressiert werden.

Neben der Definition der Muster ist auch die Gewichtung der einzelnen Muster wichtig. Obwohl die Gewichte durch die Verifikation ständig angepasst werden, ist es trotzdem notwendig, die einzelnen Muster initial richtig zu gewichten. Die Gewichtung der öffentlichen Muster kann durch Analyse der Wenn-Dann-Regeln im Regelspeicher (Nummer 5 in Abbildung 4.3) erfolgen. Dabei werden öffentliche Muster, die für die Bedingungen vieler Wenn-Dann-Regeln benötigt werden, höher gewichtet, wie Muster, die nur für eine Wenn-Dann-Regel benötigt werden. Außerdem könnten Muster, die für Wenn-Dann-Regeln benötigt werden, die häufig ausgeführt werden, höher gewichtet werden. Die Gewichtung der privaten Muster ist schwieriger, da sie nicht automatisiert erfolgen kann. Denkbar wäre beispielsweise, dass die Privacy-Experten bei der Definition der privaten Muster auch die Gewichte festlegen. Außerdem könnte der Nutzer die privaten Muster, die ihm besonders wichtig sind, kennzeichnen. Gekennzeichnete Muster können dann höher gewichtet werden. Auf diese Weise kann der Nutzer das System auch ohne Vorwissen nach seinen Privacy-Wünschen konfigurieren.

Nach der Ausführung der Wenn-Dann-Regeln in der Regelausführungskomponente (Nummer 4) erfolgt die Verifikation. Die Aufgabe der Verifikation ist es, sicherzustellen, dass tatsächlich alle privaten Muster verschleiert werden und dabei die öffentlichen Muster weiterhin erhalten bleiben. Zudem dürfen keine False Positives eingeführt werden. War das Muster *„Nutzer kommt nach Hause“* im ursprünglichen Datenstrom vorhanden, muss es auch nach der Verschleierung noch vorhanden sein. War es jedoch nicht vorhanden, darf es auch nach der Verschleierung nicht auftreten. Um dies zu überprüfen, werden alle Datenströme, die an die Regelausführungskomponente geschickt werden, und alle Datenströme, die die Regelausführungskomponente verlassen, überprüft. Sollte ein privates Muster auch nach der Verschleierung noch vorhanden sein oder ein öffentliches Muster durch die Verschleierung zerstört werden, wird das betroffene Muster in Zukunft höher gewichtet. Da die

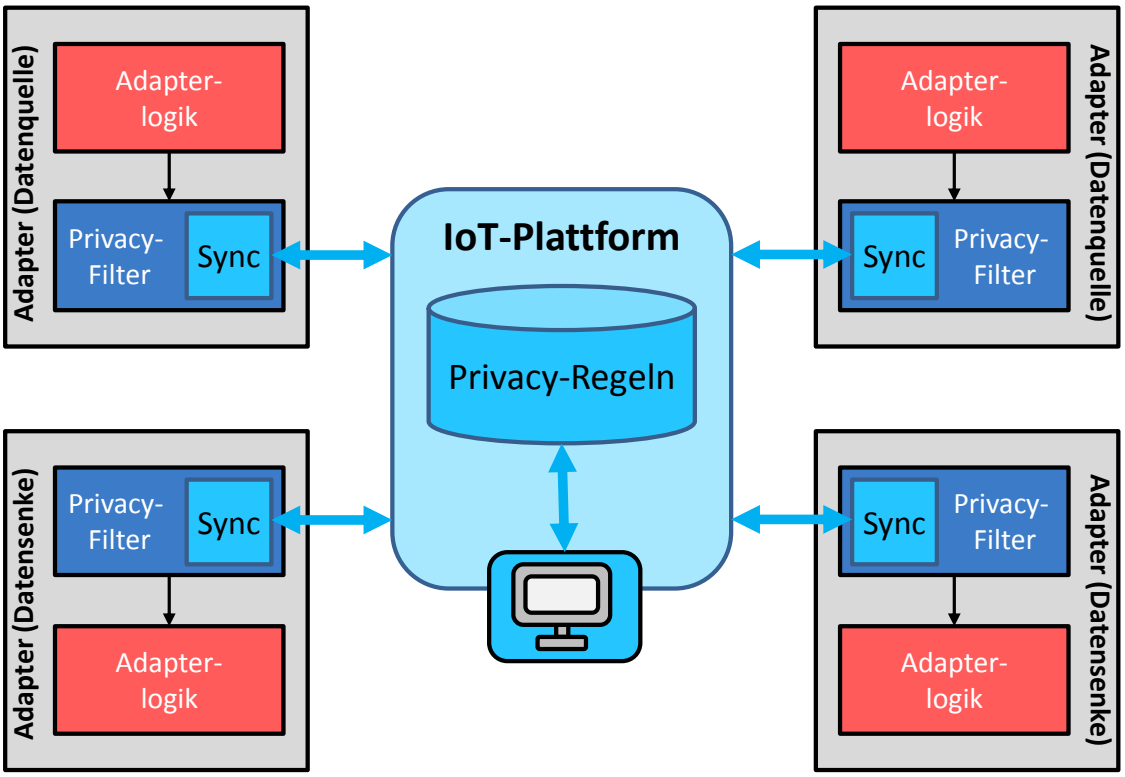

Abbildung 6.3: Verteilung der Privacy-Einstellungen

Gewichtung die Auswahl der Verschleierungstechnik beeinflusst, wird so in Zukunft eine andere Verschleierungstechnik gewählt, die die privaten Muster zuverlässig verschleiert und dabei alle öffentlichen Muster erhält.

6.4 Verteilung der Privacy-Einstellungen

Die Verteilung der Privacy-Einstellungen auf die Smart-Devices ist an AVARE angelehnt. In Abbildung 6.3 sind die Adapter von zwei Datenquellen (oben) und von zwei Datensenken (unten) zu sehen. Die Privacy-Filter der Adapter besitzen eine Synchronisationskomponente, die die Privacy-Regeln von der IoT-Plattform empfängt, damit diese für die Filterung verwendet werden können. Die Privacy-Regeln sind zentral auf der IoT-Plattform gespeichert. Dort kann der Nutzer mithilfe eines Front-Ends neue Regeln erstellen sowie bestehende Regeln anpassen oder löschen. Wurden auf der IoT-Plattform Regeln verändert, die einen bestimmten Privacy-Filter betreffen, wird die Veränderung sofort an den entsprechenden Privacy-Filter gesendet. Ist die Verbindung zu dem Smart Device unterbrochen, werden die neuen Privacy-Regeln an das Smart Device gesendet, sobald das Smart Device wieder erreichbar ist und Daten an die IoT-Plattform sendet.

Die Kommunikation zwischen der IoT-Plattform und den Smart Devices wird durch ein asymmetrisches Verschlüsselungsverfahren gesichert, um das Verändern der gesendeten Privacy-Regeln durch Dritte zu verhindern. Dazu verschlüsselt die IoT-Plattform die Privacy-Regeln mithilfe des privaten Schlüssels. Das Smart Device empfängt die so verschlüsselten Privacy-Regeln und entschlüsselt sie mit dem öffentlichen Schlüssel. Mithilfe von CHARIOT [GÖM18] kann dabei ein Großteil des Berechnungsaufwands von dem Smart Device auf eine leistungsstärkere Komponente ausgelagert werden. Durch dieses Vorgehen wird außerdem sichergestellt, dass die Privacy-Regeln, die ein Smart Device empfängt, tatsächlich von der IoT-Plattform kommen.

7 Prototyp

In diesem Abschnitt wird die Realisierung des Prototyps beschrieben. Der Prototyp soll in der Lage sein, Sensordaten zu verfälschen oder einzuschränken, private Muster in den Datenströmen der Sensordaten zu verschleiern, auf öffentliche Muster zu reagieren und die Verwendung der Aktuatoren zu reglementieren. Ziel des Prototyps ist es, die Realisierung von SPYaware anhand eines realitätsnahen Anwendungsfalls aus dem Smart-Home-Umfeld zu demonstrieren. In Abschnitt 7.1 wird zunächst das Anwendungsbeispiel vorgestellt, das prototypisch umgesetzt werden soll. Anschließend wird in Abschnitt 7.2 darauf eingegangen, wie das Anwendungsbeispiel in dem Prototyp umgesetzt wurde.

7.1 Anwendungsszenario

In Abbildung 7.1 ist ein Anwendungsbeispiel zu sehen, das zeigt, wie SPYaware in einem Smart Home eingesetzt werden kann und prototypisch umgesetzt wird. In diesem Anwendungsszenario geht es darum, die Heizung einzuschalten, wenn der Nutzer nach Hause kommt. Zusätzlich soll der Backofen vorgewärmt werden, wenn der Nutzer nach 17 Uhr nach Hause kommt. Dazu muss erkannt werden, wenn der Nutzer sich auf das Haus zubewegt. Allerdings soll verschleiert werden, wenn der Nutzer sich auf dem Heimweg auf den Supermarkt zubewegt. Wie in Abbildung 7.1 zu sehen, bestehen die Datenquellen aus dem GPS-Sensor des Autos, dem Adressbuch des Nutzers und einer Uhr. Das GPS-Signal wird benötigt, um den aktuellen Standort des Nutzers festzustellen. Die Adressen des Hauses und des Supermarktes sind in dem Adressbuch gespeichert. Die Uhr wird benötigt, da der Backofen erst nach 17 Uhr eingeschaltet werden darf.

Jeder Datenquelle ist ein Adapter zugeordnet, der wiederum aus Adapterlogik und Privacy-Filter besteht. Die Adapterlogik für den GPS-Sensor wandelt die Positionsangaben des Sensors in einen Standort um. Der Privacy-Filter sorgt anschließend dafür, dass der ermittelte Standort nur auf 100 Meter genau weitergegeben wird. Dazu wird die in Abschnitt 4.4 beschriebene Filtertechnik eingesetzt. Zunächst wird zufällig ein Winkel zwischen 0 und 360 Grad gewählt. Anschließend wird die tatsächliche Position um eine zufällige Entfernung in Richtung des gewählten Winkels verschoben. Die zufällige Entfernung muss dabei kleiner als 100 Meter sein. Die Adapterlogik für das Adressbuch ruft die Adressdaten aus dem Adressbuch ab und leitet sie an den Privacy-Filter weiter. Der Privacy-Filter sorgt dafür, dass nur die Datensätze des Hauses und des Supermarktes weitergegeben werden. Außerdem werden nicht alle Attribute, sondern nur der Name, der Längengrad und der Breitengrad weitergegeben, da nur diese Daten für die weitere Verarbeitung benötigt werden. Die Adap-

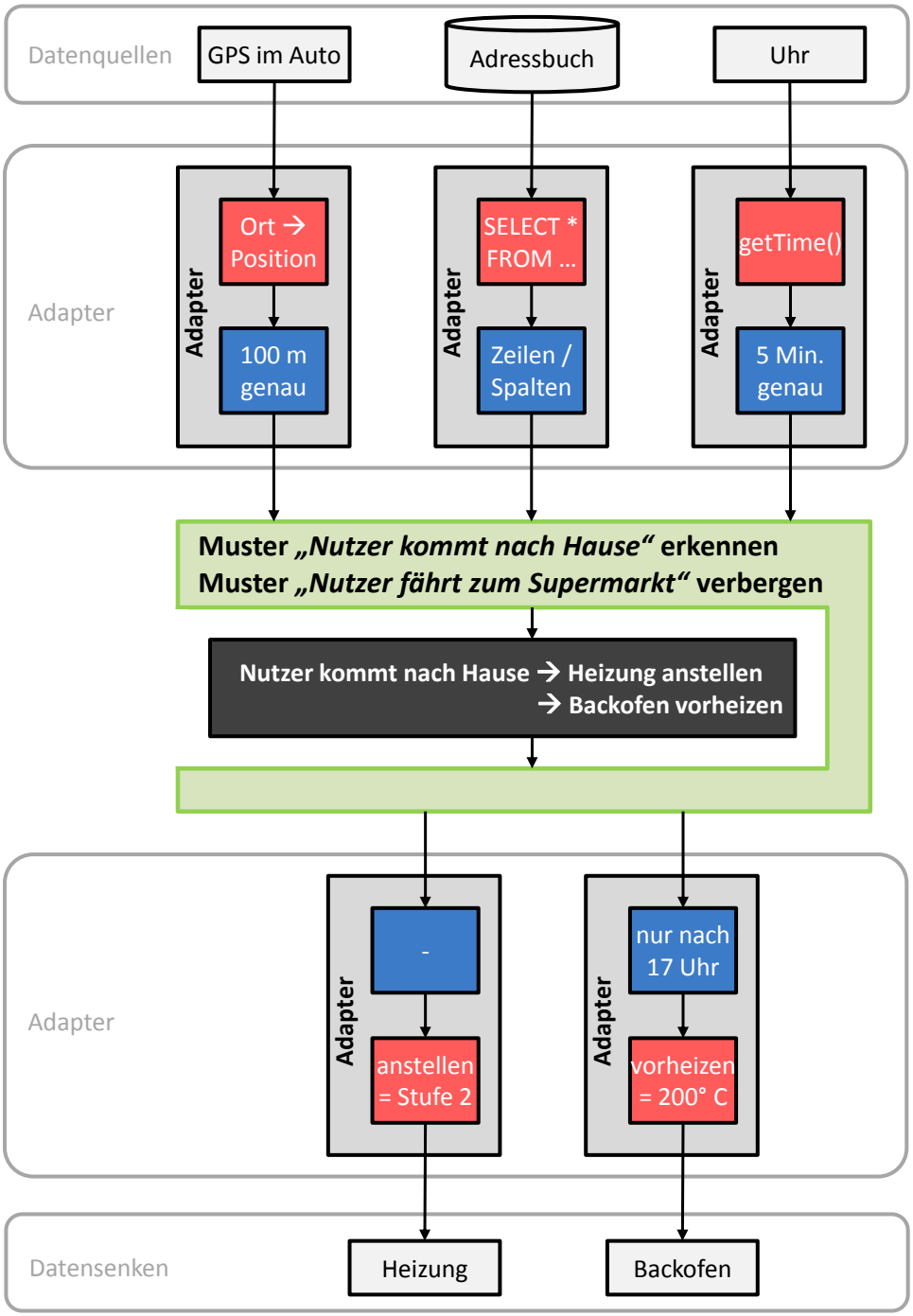

Abbildung 7.1: Anwendungsbeispiel

terlogik der Uhr hat die Aufgabe, die Uhrzeit bereitzustellen. Dabei ist es notwendig, die empfangene Uhrzeit zunächst so umzurechnen, dass sie der Uhrzeit in der lokalen Zeitzone entspricht. Der Privacy-Filter reduziert die Präzision der Uhrzeit, indem er die empfangene Uhrzeit auf volle 5 Minuten rundet.

In diesem Beispiel gibt es nur zwei Wenn-Dann-Regeln. Wenn der Nutzer nach Hause kommt, soll die Heizung angeschaltet werden. Zudem soll, wenn der Nutzer nach Hause kommt, der Backofen vorgeheizt werden. Die Regelausführungskomponente analysiert die eingehenden Datenströme und prüft, ob die Bedingungen im Wenn-Bestandteil einer Regel erfüllt sind. In diesem Beispiel ist das der Fall, wenn sich aus dem aktuellen Standort, der mithilfe des GPS-Sensors ermittelt wurde, ergibt, dass sich der Nutzer auf das Haus zubewegt. Wenn die Bedingung der Wenn-Dann-Regel erfüllt ist, löst die Regelausführungskomponente die beiden Aktionen aus, die die Heizung einschalten und den Backofen vorheizen.

Bevor die Sensordaten jedoch an die Regelausführungskomponente geschickt werden können, muss zunächst sichergestellt werden, dass die Datenströme keine privaten Muster enthalten. Dafür ist die Zugriffssteuerung zuständig. In diesem Beispiel soll das Muster *„Nutzer kommt nach Hause“* erhalten bleiben. Das ist notwendig, da in der Regelausführungskomponente die Bedingung der Wenn-Dann-Regel nicht mehr erfüllt werden kann, wenn dieses Muster entfernt wird. Das Muster *„Nutzer kauft gerade ein“* soll hingegen verschleiert werden. Dieses private Muster wurde durch den Nutzer definiert, da er nicht möchte, dass nachvollzogen werden kann, wo und wann er einkauft. Dies könnte beispielsweise dann relevant sein, wenn sich in dem Smart Home ein intelligenter Kühlschrank befindet, der die benötigten Lebensmittel selbst bestellt. Der Anbieter dieses Kühlschranks könnte an dem Einkaufsverhalten des Nutzers interessiert sein, um seinen Dienst noch weiter auf die Bedürfnisse des Nutzers hin zu optimieren. Der Nutzer möchte sich hingegen nicht an einen einzigen Anbieter binden und unabhängig und unbemerkt von dem Anbieter des Kühlschranks einkaufen können.

Die Adapter für die Datensenken empfangen die Befehle von der Regelausführungskomponente und leiten sie an die Datensenken weiter. In diesem Beispiel gibt es zwei Datensenken. Die erste Datensenke ist die Heizung, die gemäß den Wenn-Dann-Regeln eingeschaltet werden soll. Der Backofen, der vorgeheizt werden soll, ist die zweite Datensenke. Zu jeder Datensenke existiert ein Adapter, der wiederum aus dem Privacy-Filter und der Adapterlogik besteht. Für die Heizung ist in diesem Beispiel keine Filterung notwendig. Der Privacy-Filter leitet daher die Daten genauso weiter, wie er sie empfängt. Die Adapterlogik für die Heizung übersetzt den Befehl, die Heizung einzuschalten, in den Befehl, den Heizungsthermostat auf Stufe 2 zu stellen. Für den Backofen wird ein Privacy-Filter benötigt, der sicherstellt, dass der Befehl, den Backofen vorzuheizen, gemäß den Wünschen des Nutzers nur nach 17 Uhr ausgeführt wird. Erhält der Privacy-Filter vor 17 Uhr den Befehl, den Backofen vorzuheizen, wird der Befehl ignoriert. Auf diese Weise wird verhindert, dass die Heizung tagsüber eingeschaltet wird, wenn der Nutzer nicht zuhause ist und die höheren Raumtemperaturen tagsüber das Einschalten der Heizung nicht erfordern. Dadurch, dass das häufige Ein- und Ausschalten der Heizung vermieden wird, wird nicht nur Energie eingespart, sondern auch die Bauteile der Heizung geschont. Die Adapterlogik für den Backofen übersetzt den Befehl, den Backofen vorzuheizen, in den Befehl, die Temperatur des Backofens auf 200 Grad Celsius zu erhöhen.

7.2 Umsetzung im Prototyp

Der Fahrweg des Nutzers wird nicht auf einer echten Landkarte, sondern auf einem 100.000 Meter breiten und 100.000 Meter hohen Feld simuliert. Dementsprechend sind die Koordinaten des Büros, des Hauses und des Supermarkts keine echten Längen- und Breitengrade, sondern Werte im Intervall [0, 100.000). In dem Feld existieren neben den drei genannten Gebäuden keine weiteren Objekte. Das Feld ist eben, das heißt es gibt keine Berge und keine Erdkrümmung. Abbildung 7.2 zeigt das Feld und den darin eingezeichneten Fahrweg des Nutzers. Die nummerierten Kreise stellen die Positionen des Fahrtwegs dar, die mithilfe des simulieren GPS-Signals bereits abgefragt wurden. Um eine neue Position zu erzeugen, wird die aktuelle Position um einen zufälligen Wert nach links oder rechts und um einen zufälligen Wert nach oben oder unten verschoben. Dabei wird sichergestellt, dass der Nutzer das Feld nicht verlässt.

Die Daten des Adressbuchs werden aus einer CSV-Datei eingelesen. Die CSV-Datei enthält 500 Adressen. Zu jedem Datensatz sind der Name, die vollständige Adresse, Telefonnummer, E-Mail-Adresse, Webseite und die Koordinaten gespeichert. Bei der Uhrzeit handelt es sich nicht um die echte Uhrzeit. Stattdessen wird die Uhrzeit bei jedem Aufruf um eine Minute erhöht.

Die Privacy-Filter sind wie in Abschnitt 4.4 beschrieben implementiert. Zur Verschleierung des Standorts wird zunächst ein zufälliger Winkel gewählt. Anschließend wird die tatsächliche Position um eine zufällige Entfernung in diese Richtung verschoben. Dabei wird darauf geachtet, dass das Feld nicht verlassen wird. Aus dem Adressbuch werden lediglich die Datensätze des Hauses und des Supermarkts weitergeleitet. Außerdem werden nicht alle Attribute, sondern nur der Name und die Koordinaten weitergeleitet. Zur Verschleierung der Uhrzeit wird die Minutenzahl auf das nächstgelegene Vielfache von fünf gerundet.

Um die Zugriffssteuerung zu realisieren, wird zunächst der Vektor zwischen der letzten Position und der aktuellen Position berechnet. In Abbildung 7.2 ist dies der Vektor zwischen Position 5 und Position 6. Links und rechts von diesem Vektor befinden sich zwei Hilfsvektoren, die um den Winkel 0,5 im Bogenmaß nach links bzw. nach rechts verschoben sind. Diese Hilfsvektoren sind in Abbildung 7.2 grau dargestellt. Anschließend wird der Vektor zwischen der letzten Position und dem Haus berechnet. Dieser Vektor ist in Abbildung 7.2 grün dargestellt. Nun wird überprüft, ob der grüne Vektor zwischen den beiden grauen Hilfsvektoren liegt. Ist dies der Fall, bewegt sich der Nutzer auf das Haus zu. Auf die gleiche Art und Weise wird überprüft, ob sich der Nutzer auf den Supermarkt zubewegt. Der Winkel zwischen der letzten Position und dem Supermarkt ist in Abbildung 7.2 rot dargestellt. Wie in Abbildung 7.2 zu sehen, bewegt sich Nutzer nicht auf den Supermarkt zu. Daher ist in diesem Fall keine Verschleierung notwendig.

Wenn der Nutzer sich auf den Supermarkt zubewegt, gibt es drei Verschleierungsmodi, um das private Muster zu verschleiern. Der erste Modus verschleiert das private Muster vollständig und nimmt dabei auch in Kauf, dass das öffentliche Muster verloren geht. Dazu wird immer dann, wenn sich der Nutzer auf den Supermarkt zubewegt, die aktuelle Position un-

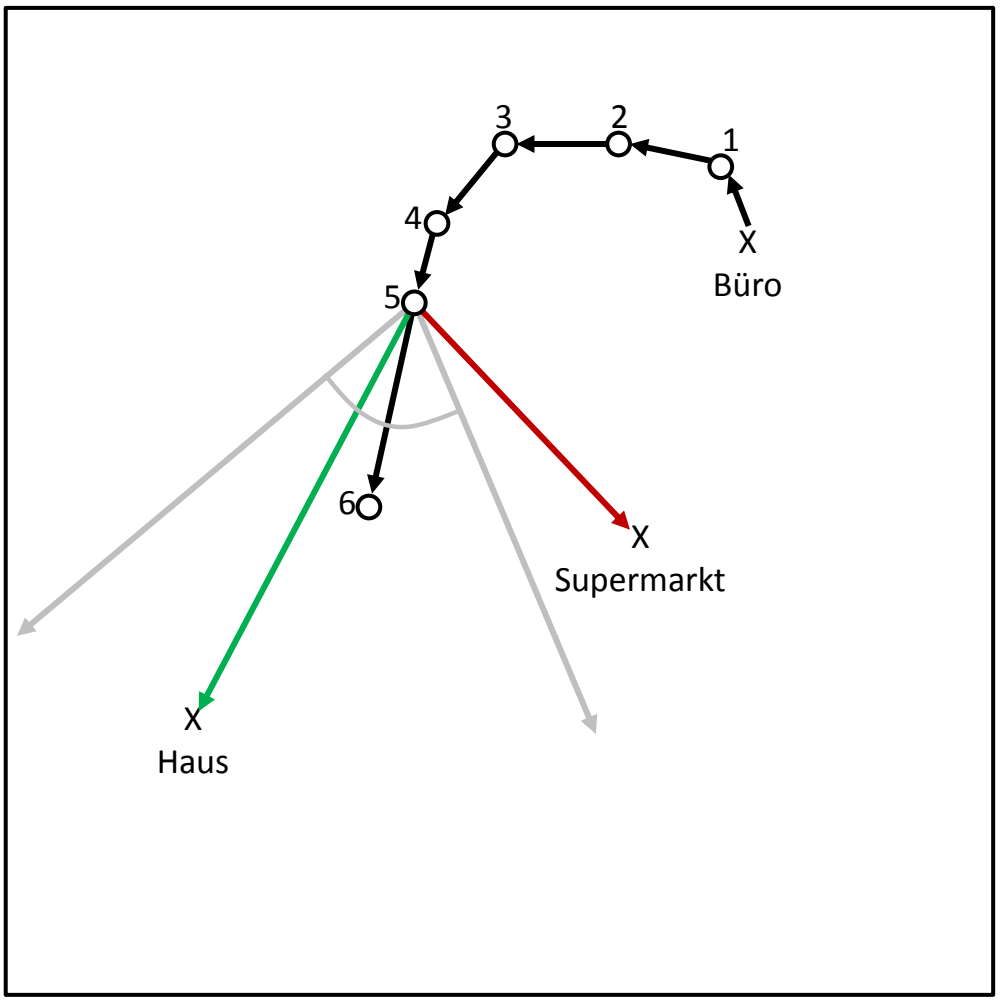

Abbildung 7.2: Fahrweg des Nutzers

terdrückt und nicht weitergegeben. Der zweite Modus sorgt dafür, dass das öffentliche Muster in jedem Fall weitergegeben wird. Dabei wird in Kauf genommen, dass auch das private Muster weitergegeben wird. Dazu wird die aktuelle Position nur dann unterdrückt, wenn sich der Nutzer auf den Supermarkt, aber nicht auf das Haus zubewegt.

Der dritte Modus versucht einen Ausgleich zwischen den beiden anderen Modi zu bieten und dabei sowohl die Privacy des Nutzers zu respektieren, als auch die Servicequalität der Anwendung zu maximieren. Wie in Abbildung 7.3 zu sehen, werden dazu der Winkel zwischen dem Positionsvektor und dem Vektor, der auf das Haus zeigt (grün), sowie der Winkel zwischen dem Positionsvektor und dem Vektor, der auf den Supermarkt zeigt (rot), berechnet. Ist wie in Abbildung 7.3 der erstgenannte Winkel kleiner, bewegt sich der Nutzer mehr auf das Haus als auf den Supermarkt zu. In diesem Fall ist das öffentliche Muster stärker als das private Muster. Durch die Weitergabe der aktuellen Position wird die Privacy des Nutzers nur geringfügig verletzt, da das private Muster nur sehr schwach ausgeprägt ist. Das öffentliche Muster ist hingegen stark ausgeprägt und sollte deshalb nicht durch die Ver-

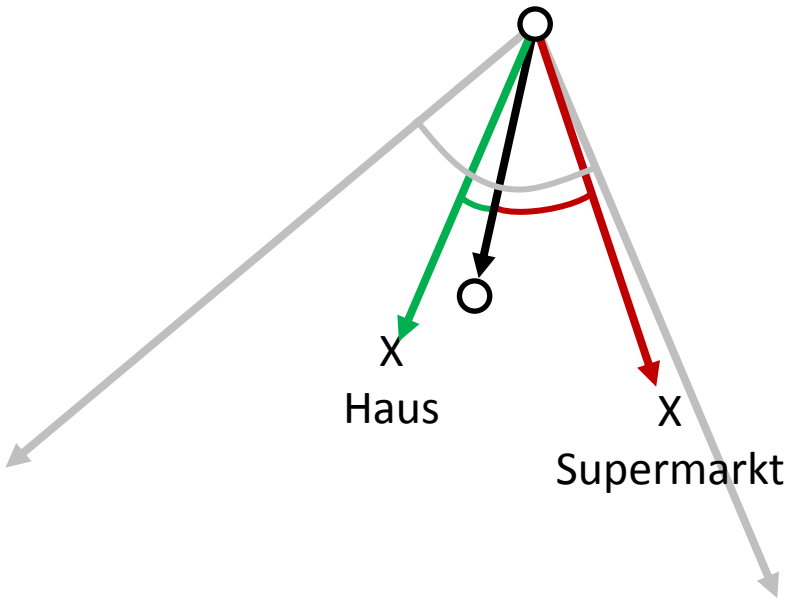

Abbildung 7.3: Situationsabhängige Verschleierung der Position

schleierung zerstört werden, um die Servicequalität der Anwendung nicht zu stark zu beeinträchtigen. Befinden sich sowohl der grüne, als auch der rote Vektor zwischen den beiden grauen Hilfsvektoren, wird die aktuelle Position deshalb weitergegeben.

Ist der zweitgenannte Winkel kleiner, bewegt sich der Nutzer mehr auf den Supermarkt als auf das Haus zu. Daher ist das private Muster stärker als das öffentliche Muster. Deshalb muss die aktuelle Position des Nutzers unbedingt unterdrückt werden, da die Weitergabe der aktuellen Position die Privacy des Nutzers stark verletzen würde. Gleichzeitig ist das öffentliche Muster nur schwach ausgeprägt. Die Unterdrückung der aktuellen Position ist daher nur eine kleine Beeinträchtigung der Servicequalität. Befinden sich sowohl der grüne, als auch der rote Vektor zwischen den beiden grauen Hilfsvektoren, wird die aktuelle Position deshalb nicht weitergegeben.

Nach der Verschleierung folgt die Überprüfung der Wenn-Dann-Regeln. Die beiden in Abbildung 7.1 zu sehenden Wenn-Dann-Regeln sind im Prototyp hartkodiert. Eine Verifikation der ausgehenden Datenströme erfolgt nicht, da es im Prototyp keinen Sinn ergeben würde. Anders als in PATRON vorgesehen, werden die empfangenen Positionen nicht verschlüsselt und persistent in einem virtuellen Datenspeicher gespeichert. Stattdessen wird nur die jeweils letzte Position im Arbeitsspeicher gespeichert. Anstatt echte Aktuatoren zu steuern, wie der Zustand der Heizung und die Temperatur des Backofens stattdessen lediglich auf der Konsole ausgegeben.

8 Evaluation

In diesem Abschnitt wird SPYaware evaluiert. Dazu wird zunächst in Abschnitt 8.1 untersucht, wie die Anforderungen aus Abschnitt 2.2 in SPYaware umgesetzt wurden. In Abschnitt 8.2 wird SPYaware mit einem verwandten System verglichen. Danach wird in Abschnitt 8.3 untersucht, inwieweit SPYaware die Privacy des Nutzers gewährleistet.

8.1 Anforderungsevaluation

SPYaware ist ein Privacy-System, das PATRON mit ACCESSORS kombiniert und in MIALinx integriert. Die Reglementierung der Datenquellen und Datensenken erfolgt mithilfe von Zugriffsregeln, die mit ACCESSORS spezifiziert werden. Die musterbasierte Verschleierung von PATRON sorgt dafür, dass die privaten Muster verschleiert werden und dabei die öffentlichen Muster erhalten bleiben. Die Muster können sich dabei auf Datenströme aus mehreren Datenquellen erstrecken. Die so verschleierten Daten werden von MIALinx analysiert. Dabei wird überprüft, ob der Wenn-Bestandteil einer der zuvor definierten Wenn-Dann-Regeln erfüllt ist. Ist dies der Fall, wird die Aktion im Dann-Bestandteil der Regel ausgeführt. Die Verteilung der Privacy-Konfiguration auf die Smart Devices erfolgt analog zu dem Verteilungsmechanismus von AVARE.

Die folgende Auflistung zeigt die Anforderungen aus Abschnitt 2.2 und deren Umsetzung in SPYaware:

Regelverarbeitung

A1 Wenn-Dann-Regeln

In SPYaware können mithilfe des Front-Ends Wenn-Dann-Regeln grafisch modelliert werden. Die Wenn-Dann-Regeln werden im Regelspeicher gespeichert. Die Regelausführungskomponente analysiert die ankommenden Daten und prüft, ob der Wenn-Bestandteil einer Regel erfüllt ist. Ist dies der Fall, werden die Aktionen im Dann-Bestandteil der Regel ausgeführt. Durch die Analyse der ankommenden Daten kann die Regelausführungskomponente dem Nutzer weitere Regeln vorschlagen. Die Regelkonfigurationskomponente optimiert bestehende Regeln, indem sie beispielsweise Regeln mit gleichem Wenn-Bestandteil zusammenfasst. Erstellt der Nutzer beispielsweise die Regeln *„Wenn der Nutzer nach Haus kommt, soll die Heizung angeschaltet werden“* und *„Wenn der Nutzer nach Haus kommt, soll der Backofen vorgewärmt werden“*, kann die Regelkonfigurations-

komponente diese beiden Regeln zu der Regel *„Wenn der Nutzer nach Haus kommt, soll die Heizung angeschaltet und der Backofen vorgewärmt werden“* zusammenfassen.

A2 Echtzeitverarbeitung

Die Echtzeitverarbeitung der ankommenden Daten wird bereits durch MIALinx unterstützt. Da SPYaware auf MIALinx aufbaut, unterstützt es ebenfalls die Echtzeitverarbeitung der ankommenden Daten. Ändert sich beispielsweise der Fahrweg des Nutzers durch eine Umleitung, erhält die Regelausführungskomponente die entsprechenden Standortdaten und löst die Regel mit dem Wenn-Bestandteil *„Wenn der Nutzer nach Hause kommt“* erst später aus.

Musterverschleierung

A3 Öffentliche Muster

SPYaware ermöglicht die Definition öffentlicher Muster. Die öffentlichen Muster entsprechen hier den Bedingungen im Wenn-Bestandteil der Wenn-Dann-Regeln. Bei der Verschleierung wird versucht, die privaten Muster zu verschleiern und dabei die öffentlichen Muster so gut wie möglich zu erhalten. In Anwendungsszenario 1 in Abschnitt 2.1 ist beispielsweise das Muster *„Nutzer bewegt sich auf das Haus zu“* ein öffentliches Muster.

A4 Private Muster

Private Muster können durch SPYaware erkannt und verschleiert werden. Dazu untersucht die Zugriffssteuerung die ankommenden Daten. Enthalten die Daten beispielsweise das private Muster *„Nutzer fährt zum Supermarkt“*, wird dieses durch die Verschleierungstechniken *blockieren*, *vertauschen* oder *verändern* entfernt. Welche Verschleierungstechnik sich am besten eignet, wird mithilfe einer Simulation ermittelt.

A5 Muster erhalten

SPYaware versucht bei der Verschleierung der privaten Muster die öffentlichen Muster so gut wie möglich zu erhalten. Dazu wird mithilfe einer Simulation anhand einer Qualitätsmetrik ermittelt, welche Verschleierungstechnik die privaten Muster verschleiert, ohne dabei die öffentlichen Muster zu zerstören.

A6 Keine False Positives

Bei der Verschleierung der privaten Muster wird darauf geachtet, keine öffentlichen Muster einzuführen, die zuvor nicht vorhanden waren. Um dies zu gewährleisten, enthält die Qualitätsmetrik, die für die Ermittlung der besten Verschleierungstechnik verwendet wird, einen Term, der die Güte einer Verschleierungstechnik, die False Positives einführt, verschlechtert.

Reglementierung der Sensoren und Aktuatoren

A7 Sensoren reglementieren

Die Reglementierung der Sensoren und der anderen Datenquellen erfolgt mithilfe von Privacy-Filtern. Die Privacy-Filter entscheiden anhand zuvor definierter Privacy-Regeln, welche Daten an das Datenstromsystem weitergegeben werden dürfen. Die Privacy-Filter sind Bestandteil der Adapter, die jeder Datenquelle zugeordnet sind. Mithilfe eines Privacy-Filters ist es beispielsweise möglich, den Standort des Nutzers nur auf 100 Meter genau weiterzugeben.

A8 Aktuatoren reglementieren

Genau wie bei den Datenquellen ist auch den Datensenken ein Adapter zugeordnet. Dieser Adapter enthält einen Privacy-Filter, der die Reglementierung der Datensenken ermöglicht. Die Reglementierung der Datensenken erfolgt wie bei den Datenquellen durch zuvor definierte Privacy-Regeln. Beispielsweise kann der Privacy-Filter sicherstellen, dass maximal zehn SMS-Nachrichten pro Tag verschickt werden.

A9 Feingranularität

Durch die Verwendung von ACCESSORS als Berechtigungsrichtlinienmodell ermöglicht SPYaware die Definition feingranularer Privacy-Regeln. Auf diese Weise kann die Privacy des Nutzers sichergestellt werden, ohne die Ausführung der Wenn-Dann-Regeln zu verhindern. Wird der Standort des Nutzers beispielsweise nur auf 100 Meter genau übertragen, reicht dies trotzdem noch für die Erkennung des Musters *„Nutzer bewegt sich auf das Haus zu“* aus.

A10 Kontextsensitivität

SPYaware ermöglicht durch die Verwendung von ACCESSORS die Definition kontextsensitiver Privacy-Regeln. Beispielsweise kann festgelegt werden, dass eine Privacy-Regel nur zu einer bestimmten Uhrzeit gilt.

A11 Erweiterbarkeit

Das verwendete Berechtigungsrichtlinienmodell von SPYaware ermöglicht die einfache Integration neuer Geräte in das bestehende System. Da sich die Privacy-Regeln immer auf bestimmte Daten und nicht auf bestimmte Geräte beziehen, ist es egal, von welchem Gerät die Daten stammen. Daher ist die Integration neuer Geräte auch ohne Anpassung der Privacy-Regeln möglich. Allerdings muss das Privacy-Modell, das beschreibt, welche Informationen von welchem Sensor abgeleitet werden können, angepasst werden.

A12 Datenzentriertheit

Wie bereits erwähnt, beziehen sich die Privacy-Regeln immer auf bestimmte Daten und nicht auf bestimmte Geräte. Dadurch ist die Reglementierung der Datenquellen und Datensenken datenzentriert. Auf diese Weise muss der Nutzer lediglich festlegen, welche Daten er geheim halten möchte. Er muss dabei nicht wissen, von welchen Sensoren diese Daten erfasst werden.

A13 Übertragungsfrequenz

Durch die Definition einer entsprechenden Privacy-Regel ist es auch möglich, die Übertragungsfrequenz der Daten einzuschränken. Die zeitliche Filterung wurde in Abschnitt 4.4 vorgestellt kann mit allen andern Filtertechniken kombiniert werden.

Konfiguration

A14 Zentrale Konfiguration

Mithilfe des in Abschnitt 4.10 vorgestellten Verteilungsmechanismus können die Privacy-Regeln an einer zentralen Stelle definiert und von dort auf die einzelnen Geräte verteilt werden. Die Geräte können sich dabei in einem heterogenen Umfeld befinden. Beispielsweise können sich einige Sensoren im Smart Home und andere Sensoren in einem Auto befinden.

8.2 Vergleich mit verwandten Systemen

In diesem Abschnitt wird die Architektur von THOR [SSM18] aus Abschnitt 3 mit der Architektur von SPYaware verglichen. Ein Vergleich mit den anderen in Abschnitt 3 vorgestellten Systemen ist nicht sinnvoll, da diese Systeme jeweils nur eine einzelne Anforderungsgruppe betrachten. So dienen *Apex* [NKZ10], *MockDroid* [BRS+11], *AppGuard* [BGH+14], *RetroSkeleton* [DC13], die *PMP* [Sta13] und *ACCESSORS* [SM18] lediglich der Reglementierung der Smart Devices. Das Konzept von *Lindner und Meier* [LM06], *ACStream* [CCF+09], *StreamShield* [NLB+09] und *PATRON* [SDM+17] beschäftigen sich ausschließlich mit dem Schutz der Daten in Datenstromsystemen. Das *Active Directory* [Hau17], die *Active Directory Rights Management Services* [Hau17], das *Simple Network Management Protocol* [MS05], *CHARIOT* [GÖM18] und das Authentifizierungsprotokoll von Gritti et al. [GÖM19] betrachten lediglich die Verteilung der Einstellungen auf verschiedene Geräte bzw. die sichere Kommunikation zwischen den Geräten.

In Tabelle 8.1 werden SPYaware und THOR miteinander vergleichen. Wie bereits in Abschnitt 8.1 gezeigt, erfüllt SPYaware alle Anforderungen. Da THOR ebenfalls MIALinx für die Regelverarbeitung nutzt, ist es auch mit THOR möglich, Wenn-Dann-Regeln zu definieren (Anforderung A1) und in Echtzeit auszuwerten (Anforderung A2). Für die Verschleierung wird PATRON verwendet, das es erlaubt, öffentliche Muster zu erkennen (Anforderung A3) und private Muster zu verschleiern (Anforderung A4). Bei der Verschleierung der privaten Muster wird versucht, die öffentlichen Muster zu erhalten (Anforderung A5) und keine False Positives hinzuzufügen (Anforderung A6).

Zur Reglementierung der Sensoren verwendet THOR die PMP. Mit der PMP können Sensoren (Anforderung A7) mithilfe von feingranularen (Anforderung A9), kontextsensitiven (Anforderung A10) und erweiterbaren (Anforderung A11) Privacy-Regeln reglementiert werden. Mithilfe der PMP ist es auch möglich, die Übertragungsfrequenz der Sensordaten

Anforderung	THOR	SPYaware
A1 Wenn-Dann-Regeln	✓	✓
A2 Echtzeitverarbeitung	✓	✓
A3 Öffentliche Muster	✓	✓
A4 Private Muster	✓	✓
A5 Muster erhalten	✓	✓
A6 Keine False Positives	✓	✓
A7 Sensoren reglementieren	✓	✓
A8 Aktuatoren reglementieren	✗	✓
A9 Feingranularität	✓	✓
A10 Kontextsensitivität	✓	✓
A11 Erweiterbarkeit	✓	✓
A12 Datenzentriertheit	✗	✓
A13 Übertragungsfrequenz	✓	✓
A14 Zentrale Konfiguration	(✓)	✓

Tabelle 8.1: Anforderungsevaluation und Vergleich mit THOR

einzuschränken (Anforderung A13). Allerdings ist in THOR die Reglementierung der Aktuatoren nicht vorgesehen (Anforderung A8). Dies ist im Kontext eines Smart Homes jedoch notwendig, um beispielsweise die Anzahl der SMS, die pro Tag versendet werden dürfen, einzuschränken, um so die Kosten für den Nutzer zu begrenzen.

Da THOR das Berechtigungsrichtlinienmodell der PMP nutzt und nicht ACCESSORS, ist die Definition datenzentrierter Privacy-Regeln nicht möglich (Anforderung A12). Die Privacy-Regeln in dem Berechtigungsrichtlinienmodell der PMP beziehen sich immer auf ein bestimmtes Service Feature, das eine Funktion einer Anwendung bzw. eines Geräts darstellt. Beispielsweise kann mit dem Berechtigungsrichtlinienmodell der PMP eine Privacy-Regel

```
Modus: Maximale Privacy
Öffentliches Muster: "Nutzer kommt nach Hause"
Privates Muster "Nutzer fährt zum Supermarkt"

96,23% der öffentlichen Muster erkannt (102 von 106)
100,00% der privaten Muster verschleiert (191 von 191)


Modus: Maximale Servicequalität
Öffentliches Muster: "Nutzer kommt nach Hause"
Privates Muster "Nutzer fährt zum Supermarkt"

100,00% der öffentlichen Muster erkannt (104 von 104)
98,94% der privaten Muster verschleiert (187 von 189)


Modus: Situationsabhängig
Öffentliches Muster: "Nutzer kommt nach Hause"
Privates Muster "Nutzer fährt zum Supermarkt"

99,05% der öffentlichen Muster erkannt (104 von 105)
99,47% der privaten Muster verschleiert (188 von 189)
```

Abbildung 8.1: Screenshot der Ausgabe des Prototyps

definiert werden, die sich auf den GPS-Sensor bezieht. Mit ACCESSORS ist es hingegen möglich, eine Privacy-Regel zu definieren, die sich auf Standortdaten bezieht, unabhängig davon, welches Gerät die Standortdaten liefert. Im Kontext eines Smart Homes ist dies von Vorteil, da die Nutzer zwar wissen, welche Daten sie geheim halten wollen, jedoch nicht, welche Sensoren diese Daten erfassen.

Die zentrale Konfiguration der Privacy-Regeln ist auch in THOR möglich (Anforderung A14). Allerdings werden in THOR für die Definition der öffentlichen Muster Datenanalysten benötigt. Die Definition der privaten Muster und der Privacy-Regeln für die PMP erfolgt durch Datenschutzbeauftragte. Die Nutzer eines Smart Homes sind jedoch keine Fachleute für die Datenanalyse und den Datenschutz und kennen sich daher mit der Definition von Analysezielen und Datenschutzzielen nicht aus. Deshalb müssen die Definition der Muster und die Konfiguration der Privacy-Regeln besonders einfach sein und softwareseitig unterstützt werden.

8.3 Gewährleistung der Privacy

In diesem Abschnitt wird untersucht, inwieweit SPYaware die Privacy des Nutzers gewährleistet. Dazu wurde ein Prototyp implementiert, der in Abschnitt 7 beschrieben wird. Der Prototyp versucht mithilfe der (simulierten) Daten eines GPS-Sensors das öffentliche Muster *„Nutzer kommt nach Hause“* zu erkennen. Um die Privacy des Nutzers zu schützen, soll das private Muster *„Nutzer fährt zum Supermarkt“* verschleiert werden. Um das private Muster

Verschleierungsmodus	Erhaltene öffentliche Muster	Verschleierte private Muster	Anteil erhaltene öffentliche Muster	Anteil verschleierter privater Muster
Maximale Privacy	102 von 106	191 von 191	96,23 %	100,00 %
Maximale Servicequalität	104 von 104	187 von 189	100,00 %	98,94 %
Situationsabhängige Verschleierung	104 von 105	188 von 189	99,05 %	99,47 %

Tabelle 8.2: Anteil der verschleierten und der erhalten gebliebenen Muster

zu verschleiern, wird die Verschleierungstechnik *blockieren* verwendet. Sobald der GPS-Sensor einen Standort übermittelt, der zusammen mit den zuvor übermittelten Standorten ein privates Muster bilden würde, wird dieser Standort aus dem Datenstrom entfernt.

Der Prototyp unterstützt die drei Verschleierungsmodi *maximale Privacy*, *maximale Servicequalität* und die *situationsabhängige Verschleierung*. Abbildung 8.1 zeigt die Ausgabe des Prototyps. Diese Ausgabe ist nicht für den Nutzer des Privacy-Systems bestimmt, sondern dient lediglich der Analyse der verschiedenen Verschleierungsmodi in diesem Abschnitt. In Tabelle 8.2 ist der Anteil der privaten Muster, die erfolgreich verschleiert wurden, und der Anteil der öffentlichen Muster, die bei der Verschleierung erhalten gebliebenen sind, zu sehen.

Im Verschleierungsmodus *maximale Privacy* wurden alle privaten Muster verschleiert. Dazu war es jedoch in 4 von 106 Fällen notwendig, auch das öffentliche Muster aus dem Datenstrom zu entfernen. Daher könnten nur 96,23 % der öffentlichen Muster erhalten werden. Die False-Negative-Rate beträgt also 3,77 %. Die False-Negative-Rate beschreibt den Anteil der öffentlichen Muster, die im Datenstrom vorhanden waren und durch die Verschleierung zerstört wurden. Die Verwendung dieses Modus ist dann sinnvoll, wenn die Privacy des Nutzers unter allen Umständen gewährleistet werden soll und man dafür bereit ist, hinzunehmen, dass die Servicequalität der Anwendung leidet. Dies ist beispielsweise dann der Fall, wenn für die Anwendung besonders Privacy-kritischen Daten wie das Bewegungsprofil des Nutzers oder medizinischen Daten benötigt werden.

Das andere Extrem ist der Verschleierungsmodus *maximale Servicequalität*. In diesem Modus wurden alle öffentlichen Muster erhalten, damit die Anwendung ohne Einschränkungen funktionieren kann. Die False-Negative-Rate beträgt 0,0 %. Allerdings konnten in diesem Modus nur 187 der insgesamt 189 privaten Muster verschleiert werden. Damit alle öffentlichen Muster erhalten bleiben konnten, war es in 2 Fällen notwendig, das private Muster preiszugeben. Dieser Modus wird dann verwendet, wenn die uneingeschränkte Funktion der Anwendung wichtiger ist, wie die Gewährleistung der Privacy des Nutzers. Die kann bei-

spielsweise bei sicherheitskritischen Anwendungen wie Brandmeldeanlagen der Fall sein. Auch ein finanzieller Vorteil wie ein günstigerer Versicherungstarif kann dazu führen, dass der Nutzer bereit ist, einen Teil seiner Privacy aufzugeben.

Die *situationsabhängige Verschleierung* versucht einen Ausgleich zwischen den beiden Extremen *maximale Privacy* und *maximale Servicequalität* zu bieten. Dabei wird versucht, die Privacy des Nutzers so gut wie möglich sicherzustellen und trotzdem die Servicequalität der Anwendung so wenig wie möglich einzuschränken. Wie in Tabelle 8.2 zu sehen, konnten in diesem Modus 99,05 % der öffentlichen Muster erhalten werden. Die False-Negative-Rate beträgt lediglich 0,95 %. Dabei wurden 99,47 % der privaten Muster verschleiert. Es blieben also mehr öffentliche Muster erhalten, wie im Verschleierungsmodus *maximale Privacy*. Gleichzeitig konnten mehr private Muster verschleiert werden, wie im Verschleierungsmodus *maximale Servicequalität*. Jedoch ist es auch in diesem Modus nicht möglich, alle privaten Muster zu verschleiern und dabei alle öffentlichen Muster zu erhalten.

Das liegt daran, dass es im Allgemeinen nicht möglich ist, beide Ziele vollständig zu erreichen. Das folgende Beispiel verdeutlicht dieses Problem: Angenommen in der Ereignissequenz A→B→C soll das private Muster A→C verschleiert werden, um die Privacy des Nutzers zu schützen. Dabei sollen die beiden öffentlichen Muster A→B und B→C erhalten bleiben, um die Servicequalität der Anwendung zu gewährleisten. Wird das Ereignis A oder B entfernt, wird dadurch das öffentliche Muster A→B zerstört. Durch das Entfernen von Ereignis B oder C wird das öffentliche Muster und B→C zerstört. Die Verschleierungstechnik *blockieren* kommt deswegen nicht infrage. Die Verschleierungstechnik *verändern* kommt ebenfalls nicht infrage, da durch das Verändern von Ereignis A oder von Ereignis B das öffentliche Muster A→B zerstört wird und durch das Verändern von Ereignis B oder Ereignis C das öffentliche Muster B→C zerstört wird.

Die Verschleierungstechnik *vertauschen* kann ebenfalls nicht angewendet werden. Um das öffentliche Muster A→B nicht zu zerstören, muss das Ereignis A weiterhin vor dem Ereignis B auftreten. Damit das öffentliche Muster B→C nicht zerstört wird, muss das Ereignis B weiterhin vor dem Ereignis C auftreten. Daher muss aufgrund der Transitivität das Ereignis A vor dem Ereignis C auftreten. Deshalb ist es nicht möglich, das private Muster A→C zu verschleiern und gleichzeitig die beiden öffentlichen Muster A→B und B→C zu erhalten.

In einem speziellen Fall kann es aber trotzdem möglich sein, beide Ziele vollständig zu erreichen. Angenommen die Ereignissequenz lautet nicht A→B→C sondern A→C→B. In diesem Fall ist es durch die Verschleierungstechnik *vertauschen* möglich, das private Muster A→C zu verschleiern und dabei das öffentliche Muster A→B zu erhalten. Dazu werden die beiden Ereignisse A und C vertauscht. In der neuen Sequenz C→A→B wurde das private Muster A→C entfernt, ohne das öffentliche Muster A→B zu zerstören.

9 Zusammenfassung und Ausblick

In einem Smart Home gibt es eine Vielzahl von Sensoren und Aktuatoren. Die Daten der Sensoren werden an ein Datenstromsystem gesendet, das die Daten analysiert und die Aktuatoren entsprechend zuvor definierter Regeln steuert. Die Sensoren erfassen dabei auch persönliche Daten über den Nutzer. Werden diese Daten miteinander kombiniert, kann das eine Gefahr für die Privacy des Nutzers darstellen.

Es gibt bereits Berechtigungsrichtlinienmodelle wie ACCESSORS, mit denen die Weitergabe der Sensordaten kontextsensitiv und feingranular eingeschränkt werden kann. Außerdem gibt es mit PATRON ein System für den Schutz privater Informationen in Datenstromsystemen. Mit PATRON ist es möglich, private Muster in den Datenströmen zu verschleiern und dabei die öffentlichen Muster zu erhalten. Jedoch gibt es noch kein System, das die Weitergabe der Sensordaten kontextsensitiv und feingranular einschränken, private Muster aus den Datenströmen aus mehreren Datenquellen verschleiern und die Verwendung der Aktuatoren reglementieren kann.

Daher wurde in dieser Arbeit das Privacy-System SPYaware entwickelt, das PATRON mit ACCESSORS kombiniert und in das Regelverarbeitungssystem MIALinx integriert. Die Privacy-Einstellungen sollen dabei zentral definiert und anschließend auf die verschiedenen Komponenten verteilt werden. Dazu wurden zunächst Anwendungsfälle im Kontext eines Smart Homes identifiziert. Anschließend wurden aus diesen Anwendungsfällen Anforderungen an SPYaware abgeleitet.

SPYaware erweitert die Adapter der Datenquellen und Datensenken um einen Privacy-Filter. Dieser Privacy-Filter entscheidet, welche Daten an das Datenstromsystem weitergeleitet werden. Es wurden verschiedene Filtertechniken vorgestellt, anhand derer die Daten gefiltert werden können. Diese Filtertechniken können mithilfe von kontextsensitiven und feingranularen Regeln modelliert werden. Alle Daten, die das Datenstromsystem erreichen, werden in der Zugriffssteuerung auf private Muster untersucht. Sind in den Datenströmen private Muster enthalten, werden diese aus den Datenströmen entfernt. Dabei wird darauf geachtet, dass die öffentlichen Muster, die für die Regelverarbeitung benötigt werden, erhalten bleiben. Nach der Regelverarbeitung wird verifiziert, dass alle privaten Muster entfernt wurden. Die anschließende Reglementierung der Datensenken erfolgt auf dieselbe Weise wie die Reglementierung der Datenquellen. Die Privacy-Regeln für die Datenquellen und Datensenken werden zentral definiert und anschließend an die Smart Devices ausgeliefert.

Um die Realisierung von SPYaware anhand eines realitätsnahen Anwendungsfalls aus dem Smart-Home-Umfeld zu demonstrieren, wurde ein Prototyp entwickelt. Der Prototyp ist in der Lage, Sensordaten zu verfälschen oder einzuschränken, private Muster in den Daten-

strömen der Sensordaten zu verschleiern, auf öffentliche Muster zu reagieren und die Aktuatoren zu reglementieren.

Anschließend wurde SPYaware anhand der zuvor definierten Anforderungen evaluiert. Dabei zeigte sich, dass SPYaware alle Anforderungen erfüllt. Außerdem wurde mithilfe des Prototyps untersucht, inwieweit SPYaware die Privacy des Nutzers gewährleistet. Der Prototyp unterstützt drei Verschleierungsmodi, mit denen entweder die maximale Privacy sichergestellt, die maximale Servicequalität gewährleistet oder situationsabhängig entschieden werden kann, wie die privaten Muster verschleiert werden. Der entwickelte Prototyp lieferte für das gegebene Anwendungsszenario in allen drei Verschleierungsmodi sehr gute Ergebnisse. Es zeigte sich allerdings auch, dass es nicht in allen Situationen möglich ist, die Privacy vollständig zu gewährleisten und dabei gleichzeitig die uneingeschränkte Servicequalität der Anwendung zu erhalten.

Die Privacy-Filter in den Datensenken sind dafür vorgesehen, die Privacy des Nutzers sicherzustellen. Beispielsweise kann eine Privacy-Regel festlegen, dass eine SMS-Nachricht nur an einen bestimmten Empfängerkreis gesendet werden darf. Neben den Privacy-Filtern könnten die Adapter für die Datensenken noch um weitere Filter erweitert werden. Ein Safety-Filter könnte dazu genutzt werden, die Sicherheit der Aktuatoren zu gewährleisten. Beispielsweise wäre es denkbar, dass der Backofen in einem Smart Home nicht länger als zwei Stunden auf maximaler Temperatur betrieben werden darf, da er sonst beschädigt wird. Dies könnte durch eine entsprechende Safety-Regel sichergestellt werden.

Zudem könnten ein zusätzlicher Verifikationsfilter genutzt werden, um die Befehle des Regelverarbeitungssystems zu verifizieren. Für die Temperatur des Backofens kommen beispielsweise nur Temperaturen zwischen 0 und 250 Grad Celsius infrage. Der Verifikationsfilter könnte überprüfen, ob die einzustellende Temperatur in diesem Intervall liegt. Möchte das Regelverarbeitungssystem fälschlicherweise die Temperatur des Backofens auf 2.000 Grad Celsius einstellen, würde der Verifikationsfilter diesen Befehl nicht an den Backofen weiterleiten. Die Safety-Regeln und die Verifikationsregeln können wie die Privacy-Regeln mithilfe von ACCESSORS definiert werden. Die Parameter für den Safety-Filter und für den Verifikationsfilter müssen von den einzelnen Aktuatoren zur Verfügung gestellt werden. So können sich beispielsweise das Temperaturintervall und die maximale Betriebsdauer bei maximaler Temperatur zwischen unterschiedlichen Backofenmodellen unterscheiden.

Zukünftige Arbeiten könnten sich außerdem damit beschäftigen, wie das Authentifizierungsprotokoll von Gritti et al. [GÖM19] in SPYaware integriert werden kann, um die Privacy des Nutzers bei der Authentifizierung der Smart Devices gegenüber externen Anbietern sicherzustellen. Beispielsweise könnte ein intelligenter Kühlschrank, wenn er feststellt, dass die Milch zur Neige geht, eine Bestellung bei einem bestimmten Supermarkt aufgeben wollen. In diesem Szenario ist der intelligente Kühlschrank die Datenquelle, welche die Bestelldaten an das Datenstromsystem schickt. Das Datenstromsystem leitet die Bestellung an den Supermarkt weiter, der in diesem Szenario die Datensenke ist.

Um sich gegenüber dem Supermarkt zu authentifizieren, übermittelt der intelligente Kühlschrank die Attribute Kundennummer, Gerätenummer und Modell an das Datenstromsystem. Das Datenstromsystem weiß jedoch, dass zur Authentifizierung gegenüber dem Supermarkt lediglich die Kundennummer benötigt wird. Deshalb entfernt das Datenstromsystem die Attribute Gerätenummer und Modell aus dem Berechtigungsnachweis. Auf diese Weise erfährt der Supermarkt nicht, welches Kühlschrankmodell der Nutzer besitzt. Da auch die Gerätenummer nicht übermittelt wird, kann der Supermarkt auch nicht feststellen, ob die Bestellungen immer von dem gleichen Kühlschrank stammen. Über das Kühlschrankmodell und die Anzahl der Kühlschränke könnte der Supermarkt Rückschlüsse auf die Anzahl der Personen, die in dem Haushalt des Nutzers leben, ziehen. Da das Datenstromsystem diese Informationen aus dem Berechtigungsnachweis entfernt, wird verhindert, dass diese persönliche Information dem Supermarkt bekannt wird.

Literaturverzeichnis

[AAL11] Arbeitsgruppen „Schnittstellenintegration und Interoperabilität“ und „Kommunikation“ der BMBF/VDE Innovationspartnerschaft AAL (2011). *AAL-Anwendungsszenarien*. Berlin: VDE Verlag GmbH. Verfügbar unter https://www.dke.de/resource/blob/846816/cb41e7588e697cd2a12abb3100d8dc57/aal-anwendungsszenarien-data.pdf (zuletzt abgerufen am 23.10.2018).

[ABF+18] Alpers, S., Betz, S., Fritsch, A., Oberweis, A., Schiefer, G., Wagner, M. (2018). *Citizen Empowerment by a Technical Approach for Privacy Enforcement*. In: Proceedings of the 8th International Conference on Cloud Computing and Services Science, 589-595.

[AEG+18] Alpers, S., Erdogan, E., Gapp, S., Fritsch, A., Miller-Askar, A., Oberweis, A., Schiefer, G., Wagner, M. (2018). PRIVACY - AVARE: Selbstdatenschutz für Bürger mithilfe von Open-Source-Software. In: Science Track FrOSCon 2018.

[AHS12] Assam, R., Hassani, M., Seidl, T. (2012). *Differential private trajectory protection of moving objects*. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on GeoStreaming, 68-77.

[AOP+17] Alpers, S., Oberweis, A., Pieper, M., Betz, S., Fritsch, A., Schiefer, G., Wagner, M. (2017). *PRIVACY-AVARE: An approach to manage and distribute privacy settings*. In: Proceedings of the 2017 3rd IEEE International Conference on Computer and Communications, 1460-1468.

[APW17] Alpers, S., Pieper, M., Wagner, M. (2017). *Herausforderungen bei der Entwicklung von Anwendungen zum Selbstdatenschutz*. In: Tagungsband der 47. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 1061-1072.

[BAK+11] Banuri, H., Alam, M., Khan, S., Manzoor, J., Ali, B., Khan, Y., Yaseen, M., Tahir, M. N., Ali, T., Alam Q., Zhang, X. (2012). *An Android runtime security policy enforcement framework*. In: Personal and Ubiquitous Computing 16(6), 631-641.

[BGH+14] Backes, M., Gerling, S., Hammer, C., Maffei, M., von Styp-Rekowsky, P. (2014). *AppGuard - Fine-grained policy enforcement for untrusted Android applications*. In: Data Privacy Management and Autonomous Spontaneous Security, 213-231. Berlin, Heidelberg: Springer.

[BO11] Barrera, D., Van Oorschot, P. (2011). *Secure software installation on smartphones*. In: IEEE Security & Privacy 9(3), 42-48.

[BRS+11] Beresford, A. R., Rice, A., Skehin, N., Sohan, R. (2011). *Mockdroid: trading privacy for application functionality on smartphones*. In: Proceedings of the 12th workshop on mobile computing systems and applications, 49-54.

[CBM+15] Cortés, R., Bonnaire, X., Marin, O., Sens, P. (2015). *Stream processing of healthcare sensor data: studying user traces to identify challenges from a big data perspective*. In: Procedia Computer Science 52, 1004-1009.

[CCF+09] Cao, J., Carminati, B., Ferrari, E., Tan, K. L. (2009). *ACStream: Enforcing Access Control over Data Streams*. In: 2009 IEEE 25th International Conference on Data Engineering, 1495-1498.

[CCF+12] Conti, M., Crispo, B., Fernandes, E., Zhauniarovich, Y. (2012). *CRêPE: A System for Enforcing Fine-Grained Context-Related Policies on Android*. In: IEEE Transactions on Information Forensics and Security 7(5), 1426-1438.

[CFS+14] Chen, C. Y., Fu, J. H., Sung, T., Wang, P. F., Jou, E., Feng, M. W. (2014). *Complex event processing for the Internet of Things and its applications*. In: Proceedings of the 2014 IEEE International Conference on Automation Science and Engineering, 1144-1149.

[CLH+14] Chen, X., Li, J., Huang, X., Li, J., Xiang, Y., Wong, D. S. (2014). *Secure outsourced attribute-based signatures*. In: IEEE Transactions on Parallel and Distributed Systems 25(12), 3285-3294.

[DC13] Davis, B., Chen, H. (2013). *RetroSkeleton: retrofitting android apps*. In: Proceeding of the 11th annual international conference on Mobile systems, applications, and services, 181-192.

[DLY+06] Davidoff, S., Lee, M. K., Yiu, C., Zimmerman, J., Dey, A. K. (2006). Principles of smart home control. In: International conference on ubiquitous computing, 19-34. Berlin, Heidelberg: Springer.

[Eck13] Eckert, C. (2013). *IT-Sicherheit: Konzepte - Verfahren - Protokolle*. 8., aktualisierte und korrigierte Auflage. München: Oldenbourg.

[EU16] Europäisches Parlament und Rat der Europäischen Union (2016). *Verordnung (EU) 2016/679 des Europäischen Parlaments und des Rates vom 27. April 2016 zum Schutz natürlicher Personen bei der Verarbeitung personenbezogener Daten, zum freien Datenverkehr und zur Aufhebung der Richtlinie 95/46/EG*.

[FBJ+12] Fragkaki, E., Bauer, L., Jia, L., Swasey, D. (2012). *Modeling and enhancing android's permission system*. In: European Symposium on Research in Computer Security, 1-18. Berlin, Heidelberg: Springer.

[FHE+12] Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D. (2012). *Android permissions: User attention, comprehension, and behavior*. In: Proceedings of the eighth symposium on usable privacy and security, 3.

[GÖM18] Gritti, C., Önen, M., Molva, R. (2018). *CHARIOT: Cloud-Assisted Access Control for the Internet of Things*. In: Proceedings of the 16th International Conference on Privacy, Security and Trust, 1-6.

[GÖM19] Gritti, C., Önen, M., Molva, R. (2019). *Privacy-preserving delegatable authentication in the Internet of Things*. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing.

[Hau17] Hauenherm, E. (2017). *Effiziente Kommunikation im Unternehmen: Konzepte & Lösungen mit Microsoft-Plattformen*. München: Hanser Verlag.

[HHJ+11] Hornyack, P., Han, S., Jung, J., Schechter, S., Wetherall, D. (2011). *"These Aren't the Droids You're Looking For": Retrofitting Android to Protect Data from Imperious Applications*. In: Proceedings of the 18th ACM conference on Computer and communications security, 639-652.

[ISO13] International Organization for Standardization (2013). *ISO/IEC 27001:2013 Information technology – Security techniques – Information security management systems – Requirements*.

[JMV+12] Jeon, J., Micinski, K. K., Vaughan, J. A., Fogel, A., Reddy, N., Foster, J. S., Millstein, T. (2012). *Dr. Android and Mr. Hide: Fine-Grained Permissions in Android Applications*. In: Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices, 3-14.

[KLC02] Keogh, E., Lonardi, S., Chiu, B. Y.-C. (2002). *Finding Surprising Patterns in a Time Series Database in Linear Time and Space*. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 550-556.

[KMS11] Kumar, Y., Munjal, R., Sharma, H. (2011). *Comparison of symmetric and asymmetric cryptography with existing vulnerabilities and countermeasures*. In: International Journal of Computer Science and Management Studies 11(03), 60-63.

[KSC14] Kim, S., Sung, M. K., Chung, Y. D. (2014). *A framework to preserve the privacy of electronic health data streams*. In: Journal of biomedical informatics 50, 95-106.

[LKS+16] Ludwig, T., Kotthaus, C., Stein, M., Durt, H., Kurz, C., Wenz, J., Doublet, T., Becker, M., Pipek, V., Wulf, V. (2016). *Arbeiten im Mittelstand 4.0 - KMU im Spannungsfeld des digitalen Wandels*. In: HMD Praxis der Wirtschaftsinformatik 53(1), 71-86.

[LM06] Lindner, W., Meier, J. (2006). *Securing the borealis data stream engine*. In: 10th International Database Engineering and Applications Symposium, 137-147.

[LOW08] Li, J., Ooi, B. C., Wang, W. (2008). *Anonymizing streaming data for privacy protection*. In: 2008 IEEE 24th International Conference on Data Engineering, 1367-1369.

[Moo08] Moore, A. (2008). *Defining privacy*. In: Journal of Social Philosophy 39(3), 411-428.

[MS05] Mauro, D. R., Schmidt, K. (2005). *Essential SNMP: Help for System and Network Administrators*. 2. Auflage. Sebastopol: O'Reilly Media.

[NKZ10] Nauman, M., Khan, S., Zhang, X. (2010). *Apex: Extending Android Permission Model and Enforcement with User-defined Runtime Constraints*. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, 328-332.

[NLB+09] Nehme, R. V., Lim, H. S., Bertino, E., Rundensteiner, E. A. (2009). *StreamShield: A stream-centric approach towards security and privacy in data stream environments*. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 1027-1030.

[PDT+18] Palanisamy, S. M., Dürr, F., Tariq, M. A., Rothermel, K. (2018). *Preserving Privacy and Quality of Service in Complex Event Processing through Event Reordering*. In: Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems, 40-51.

[Poh04] Pohl, H. (2004). *Taxonomie und Modellbildung in der Informationssicherheit*. In: Datenschutz und Datensicherung 28(11), 678-685.

[Pöt08] Pötzsch, S. (2008). Privacy awareness: *A means to solve the privacy paradox?* In: IFIP Summer School on the Future of Identity in the Information Society, 226-236. Berlin, Heidelberg: Springer.

[RCF+11] Russello, G., Crispo, B., Fernandes, E., Zhauniarovich, Y. (2011). *Yaase: Yet another android security extension*. In: 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, 1033-1040.

[SAB+18] Stach, C., Alpers, S., Betz, S., Dürr, F., Fritsch, A., Mindermann, K., Palanisamy, S. M., Schiefer, G., Wagner, M., Mitschang, B., Oberweis, A., Wagner, S. (2018). *The AVARE PATRON: A Holistic Privacy Approach for the Internet of Things*. In: Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, 372-379.

[SDM+17] Stach, C., Dürr, F., Mindermann, K., Palanisamy, S. M., Tariq, M. A., Mitschang, B., Wagner, S. (2017). *PATRON - Datenschutz in Datenstromverarbeitungssystemen*. In: Tagungsband der 47. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 1085-1096.

[SDM+18] Stach, C., Dürr, F., Mindermann, K., Palanisamy, S. M., Wagner, S. (2018). *How a Pattern-based Privacy System Contributes to Improve Context Recognition*. In: Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops, 238-243.

[SM13] Stach, C., Mitschang, B. (2013). *Privacy Management for Mobile Platforms - A Review of Concepts and Approaches*. In: Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management, 305-313.

[SM14] Stach, C., Mitschang, B. (2014). *Design and implementation of the privacy management platform*. In: Proceedings of the 2014 IEEE 15th International Conference on Mobile Data Management, 69-72.

[SM18] Stach, C., Mitschang, B. (2018). *ACCESSORS. A Data-Centric Permission Model for the Internet of Things*. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy, 30-40.

[SMA+17] Scoccia, G. L., Malavolta, I., Autili, M., Di Salle, A., Inverardi, P. (2017). *User-centric Android flexible permissions*. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, 365-367.

[SPH11] Schreckling, D., Posegga, J., Hausknecht, D. (2012*). Constroid: data-centric access control for android*. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing, 1478-1485.

[SSM18] Stach, C., Steimle, F., Mitschang, B. (2018). *THOR - Ein Datenschutzkonzept für die Industrie 4.0*. In: Tagungsband der 48. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 71-83.

[Sta13] Stach, C. (2013). *Wie funktioniert Datenschutz auf Mobilplattformen?*. In: Tagungsband der 43. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2072-2086.

[SWV14] Sipior, J. C., Ward, B. T., Volonino, L. (2014). *Privacy concerns associated with smartphone use*. In: Journal of Internet Commerce 13(3-4), 177-193.

[VRM+17] Vashi, S., Ram, J., Modi, J., Verma, S., Prakash, C. (2017). *Internet of Things (IoT): A Vision, Architectural Elements, and Security Issues*. In: 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), 492-496.

[Way14] Waye, L. (2014). *Privacy integrated data stream queries*. In: Proceedings of the 2014 International Workshop on Privacy & Security in Programming, 19-26.

[WHS+16] Wieland, M., Hirmer, P., Steimle, F., Gröger, C., Mitschang, B., Rehder, E., Lucke, D., Rahman, O. A., Bauernhansl, T. (2016). *Towards a Rule-based Manufacturing Integration Assistant*. In: Procedia CIRP 57, 213-218.

[WLL+10] Wu, M., Lu, T. J., Ling, F. Y., Sun, J., Du, H. Y. (2010). *Research on the architecture of Internet of Things*. In: Proceedings of the 2010 3rd International Conference on Advanced Computer Theory and Engineering, 484-487.

[WSM+17] Wieland, M., Steimle, F., Mitschang, B., Lucke, D., Einberger, P., Schel, D., Luckert, M., Bauernhansl, T. (2017). *Rule-Based Integration of Smart Services Using the Manufacturing Service Bus*. In: Proceedings of the 2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation, 1-8.

[XSA12] Xu, R., Saïdi, H., Anderson, R. J. (2012). *Aurasium: Practical Policy Enforcement for Android Applications*. In: Proceedings of the 21st USENIX Security Symposium, 539-552.

[ZCW+14] Zhang, Z.-K., Cho, M. C. Y., Wang, C.-W., Hsu, C.-W., Chen, C.-K., Shieh, S. (2014). *IoT Security: Ongoing Challenges and Research Opportunities*. In: Proceedings of the 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, 230-234.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

_____

Ort, Datum, Unterschrift