Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 1085

PATRON — Datenschutz in Datenstromverarbeitungssystemen

Christoph Stach[1], Frank Dürr[1], Kai Mindermann[1], Saravana Murthy Palanisamy[1], Muhammad Adnan Tariq[1], Bernhard Mitschang[1], Stefan Wagner[1]

Abstract: Aufgrund der voranschreitenden Digitalisierung gewinnt das *Internet der Dinge* (*IoT*) immer mehr an Bedeutung. Im *IoT* werden Geräte mit Sensoren ausgestattet und miteinander vernetzt. Dadurch werden neuartige Anwendungen ermöglicht, in denen Sensordaten miteinander kombiniert und in höherwertige Informationen umgewandelt werden. Diese Informationen verraten viel über den Nutzer und müssen daher besonders geschützt werden. Häufig hat der Nutzer allerdings keine Kontrolle über die Verarbeitung seiner Daten. Auch kann er das Ausmaß der daraus ableitbaren Informationen nicht ermessen. Wir stellen daher einen neuartigen Kontrollmechanismus vor, der private Informationen im *IoT* schützt. Anstelle von abstrakten Datenschutzregeln für einzelne Sensoren definiert der Nutzer Muster, die geschützt werden müssen. Der Nutzer definiert die zu verheimlichenden Informationen natürlichsprachlich, und ein Domänenexperte setzt diese in formale Regeln um. Sind diese Regeln zu restriktiv, so kann die Anwendung ihre angedachte Funktionalität nicht erbringen. Daher muss bezüglich der Servicequalität ein Kompromiss zwischen gewünschter Privatheit und benötigter Funktionalität gefunden werden.

Keywords: Datenschutz; Zugriffskontrolle; Datenströme; Internet der Dinge; Privatheit; Sensoren.

1 Einleitung

Heutzutage werden immer mehr Geräte mit Sensoren ausgestattet, die unterschiedlichste Aspekte ihrer Umgebung erfassen (z. B. den Standort) und daraus höherwertige Informationen ableiten (z. B. in welchem Kontext sie gerade verwendet werden). Im *Internet der Dinge* (engl. *Internet of Things*, *IoT*) werden diese Geräte miteinander vernetzt. Durch Informationsaustausch stehen damit den *IoT*-Anwendungen Daten aus den unterschiedlichsten Domänen zur Verfügung. Abb. 1 zeigt eine typische Architektur für eine solche Anwendung. Sensordaten werden in einem Datenstromverarbeitungssystem (*DSS*) miteinander verknüpft und mit historischen Daten aus weiteren Quellsystemen angereichert. Das daraus gewonnene Wissen wird Anwendungen zur Verfügung gestellt. Aufgrund der großen und vielfältigen Menge an verfügbaren Daten werden neuartige Anwendungen ermöglicht.

Die fortschreitende Digitalisierung ist zugleich Fluch und Segen. Während durch die Kombination einzelner Sensordaten neues Wissen gewonnen werden kann, stellt genau dies

[1] Universität Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Vorname.Nachname@informatik.uni-stuttgart.de

doi:10.18420/in2017_110

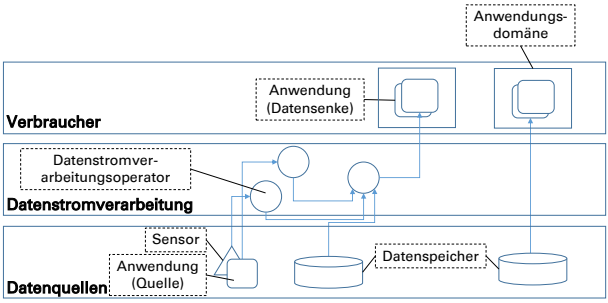

Abb. 1: Beispielarchitektur von Anwendungen für das *Internet der Dinge*

ebenfalls eine Gefahr dar. Jedes Sensordatum für sich betrachtet erscheint möglicherweise harmlos und für die Ausführung einer Anwendung erforderlich. Welches Wissen sich daraus ableiten lässt, kann der Nutzer[2] jedoch nicht überblicken. Daher versucht der Gesetzgeber zunehmend die Rechte der Nutzer zu stärken, z. B. durch die neue EU-Datenschutz-Grundverordnung. Der Nutzer kann jedoch nicht überprüfen, ob sich eine Anwendung an diese Verordnung hält. Daher bedarf es technischer Lösungen, mit denen der Nutzer seine Datenschutzanforderungen formulieren kann und die verifizieren können, ob diese Anforderungen eingehalten werden. Aktuelle Datenschutzsysteme decken nur Teilaspekte dieser Problematik ab und sind nicht an komplexe *IoT*-Anwendungen angepasst.

Zu diesem Zweck erbringen wir die folgenden vier Beiträge: (*I*) Wir beschreiben einen typischen Anwendungsfall für das *IoT*, in dem sehr viele vertrauliche Daten anfallen. (*II*) Ausgehend von diesem Beispiel leiten wir Anforderungen an ein Datenschutzsystem für das *IoT* ab. (*III*) Zur Erfüllung dieser Anforderungen stellen wir mit PATRON[3] ein neuartiges Datenschutzsystem fürs *IoT* vor, das sich einfach in die Beispielarchitektur von *IoT*-Anwendungen (siehe Abb. 1) einpassen lässt. Mithilfe dieser Erweiterungen ist es dem Nutzer möglich, seine Datenschutzanforderungen zu formulieren. Er definiert zu schützende Muster, d. h. Attributkombinationen, die von PATRON nicht gegenüber Anwendungen preisgegeben werden dürfen. Hierfür stehen PATRON mehrere unterschiedliche Techniken zur Verfügung, von denen PATRON sich stets diejenige aussucht, die die Servicequalität am besten erhält. Servicequalität bedeutet in diesem Fall, dass eine Anwendung mit möglichst vielen Daten versorgt wird, damit sie den größtmöglichen Nutzen erbringen kann, ohne dabei die vertraulichen Muster zu veröffentlichen. Die Güte der Konfiguration (d. h. inwiefern sie alle vertraulichen Daten schützt) wird fortlaufend überprüft. (*IV*) Wir evaluieren PATRON und zeigen, inwiefern dieser Ansatz die aufgestellten Anforderungen erfüllt.

Der Rest dieses Artikels ist wie folgt gegliedert: Zunächst wird in Abschnitt 2 ein *IoT*-Anwendungsfall im Bereich *E-Health* beschrieben. Abschnitt 3 stellt auf Basis dieses Beispiels einen Anforderungskatalog an ein *IoT*-Datenschutzsystem auf. In Abschnitt 4

[2] Mit dem Begriff „Nutzer“ seien jeweils beide Geschlechter gleichermaßen adressiert.
[3] PATRON steht für **P**riv**a**cy in S**tr**eam Pr**o**cessi**n**g.

werden verwandte Arbeiten betrachtet. Das PATRON-System ist Gegenstand von Abschnitt 5. Abschnitt 6 evaluiert hierauf unseren Ansatz und überprüft, inwiefern PATRON die aufgestellten Anforderungen erfüllt. Abschließend fasst Abschnitt 7 den Artikel zusammen.

2 Anwendungsfall E-Health

Von der voranschreitenden Digitalisierung und der Vernetzung von Sensoren kann das Gesundheitswesen profitieren. Im sogenannten *E-Health* nehmen Patienten mit speziellen Messgeräten notwendige Untersuchungen weitestgehend selbstständig vor. Die Gesundheitsdaten werden anschließend gesammelt, analysiert und an den zuständigen Arzt weitergeleitet [Ku13]. Auf diese Weise werden nicht nur Behandlungskosten reduziert, sondern auch die behandelnden Ärzte stark entlastet. Insbesondere bei chronischen Erkrankungen, die regelmäßig Therapiemaßnahmen benötigen, ist *E-Health* daher nützlich [BL01]. Im Folgenden wird gezeigt, welche Daten gesammelt werden und wer diese Daten verarbeitet.

Diabetes ist nicht heilbar, jedoch lassen sich die Folgeerscheinungen abschwächen. Hierfür müssen die Patienten u. a. regelmäßige Blutzuckermessungen vornehmen und ein Diabetestagebuch führen. Besonders für Kinder ist dies eine große Belastung. Mit sogenannten *Serious Games* kann eine Motivation für die jungen Patienten geschaffen werden. Knöll [Kn10] stellt mit Candy Castle ein Konzept eines Gesundheitsspiels für diabeteskranke Kinder vor. In der Android-basierten Implementierung dieses Konzepts [SS12; St16], erhält der Spieler ein virtuelles Schloss, das seine Gesundheit repräsentiert. Dieses Schloss wird regelmäßig von *dunklen „Mächten“* angegriffen. Als Gegenmaßnahme muss der Spieler Mauern errichten und so sein Königreich verteidigen. Das errichten der Mauer „kostet“ eine Blutzuckermessung sowie Angaben zu seinen Aktivitäten, seinen Mahlzeiten sowie seinem Empfinden. Da der Ort der Blutzuckermessung ebenfalls für die Analyse der Werte eine Rolle spielen kann [Kn10], wird jede Messung mit einer Ortsangabe annotiert.

Diese Werte werden mittels Sensoren automatisch erfasst. Der Blutzuckerwert kann mittels eines Glucometers erfasst werden. Die Aktivitäten des Spielers lassen sich anhand von Bewegungsmustern über Orientierungs-, Lage- und Beschleunigungssensoren ermitteln [Kw11]. Die Broteinheiten einer Mahlzeit können anhand von Fotos bestimmt werden [Al12]. Die Laune des Spielers lässt sich durch ein Mikrofon bestimmen [Me12]. Die Standortdaten können mittels eines GPS-Sensors ermittelt werden. All diese Sensoren befinden sich bereits in einem handelsüblichen Smartphone oder sie können sehr leicht daran angeschlossen werden. Die Sensordaten werden zu medizinischen Datensätzen zusammengefasst, auf dem Smartphone gespeichert und zur Verarbeitung an ein Back-End geschickt. Dort werden die eingehenden Datenströme analysiert, gespeichert und Ärzten zugänglich gemacht. Als weitere Funktion ermöglicht *Candy Castle* Eltern Zugriff auf die Gesundheits- und Standortdaten ihrer Kinder zu erhalten. Dies ist z. B. bei einem Zuckerschock erforderlich.

Dieses Beispiel zeigt, dass durch eine solche *IoT*-Anwendung sehr viele Daten gesammelt werden, die ein breites Spektrum an Rückschlüssen über den Nutzer zulassen. Daher sind

neben *Patienten*, *Ärzten* und *Eltern* auch noch weitere Gruppen an diesen Daten interessiert. Z. B. wollen Versicherungen den Behandlungsverlauf mitverfolgen, und auf diese Weise Tarife dynamisch anpassen und Städtebauer wollen durch die Kombination aus Gesundheits- und Standortdaten besonders gesunde Orte identifizieren.

Es ist jedoch offensichtlich, dass nicht jede Partei den gleichen Zugriff auf die Daten haben darf. Beispielsweise sollten Ärzte stets vollen Zugang zu allen Gesundheitsdaten haben, während Städtebauer nur anonymisierte und aggregierte Daten benötigen. Die Zugriffsrechte sollten bei Bedarf allerdings flexibel entzogen werden können. Eltern sollen z. B. nur in einem Notfall Zugriff auf die Standortsdaten ihrer Kinder haben.

3 Anforderungsanalyse

Motiviert von dem oben angeführten realitätsnahen *E-Health*-Szenario lassen sich die folgenden 8 Anforderungen an ein Datenschutzsystem ableiten. Zu jeder Anforderung wird jeweils ein Beispiel aus dem beschriebenen Anwendungsfall *Candy Castle* gegeben.

(*A*) **Einfache Konfiguration.** Die Konfiguration des Datenschutzsystems muss sehr einfach sein, damit der Nutzer einerseits bereit ist, das System zu verwenden, und er andererseits in der Lage ist, die Datenschutzregeln entsprechend seiner Wünsche zu erstellen. Die Nutzer von *IoT*-Anwendungen sind häufig keine IT-Spezialisten, so dass die Konfiguration für sie möglichst natürlichsprachlich erfolgen muss. In dem gegebenen *E-Health*-Szenario könnte eine solche Konfiguration beispielsweise lauten: „Meine Versicherung darf keinen Zugriff auf meine Standortdaten haben."

(*B*) **Musterorientiert.** Die meisten Datenschutzsysteme beschränken sich darauf, einzelne Sensorwerte nur verfälscht oder gar nicht an eine Anwendung weiterzuleiten. Ein Nutzer kann allerdings nicht absehen, welche Erkenntnisse aus einer Kombination von unterschiedlichen Sensorwerten gewonnen werden kann. Daher muss der Nutzer die Möglichkeit haben, Muster zu definieren, die verheimlicht werden sollen. In dem gegebenen *E-Health*-Szenario könnte ein solches Muster beispielsweise lauten: „Meine Versicherung darf nichts über mögliche, bislang noch nicht diagnostizierte Erkrankungen erfahren." Dabei wird nicht weiter spezifiziert, welche Kombinationen von Sensordaten auf eine solche Erkrankung hinweisen könnten.

(*C*) **Datenstromverarbeitung.** Sensoren liefern in der Regel einen kontinuierlichen Fluss von Daten. Sollen die Daten schnell verarbeitet werden, so muss die *IoT*-Anwendung direkt auf dem Datenstrom arbeiten. Das Datenschutzsystem muss daher in der Lage sein, eine strombasierte Verarbeitung von Daten zu unterstützen. In dem gegebenen *E-Health*-Szenario könnte ein Anwendungsfall für eine strombasierte Verarbeitung in der Analyse von Gesundheitswerten in Echtzeit bestehen. Diese Daten müssen schnell verarbeitet werden, um unmittelbar auf Zustandsveränderungen reagieren zu können, z. B. durch die Zuführung von Insulin bei einem Anstieg des Blutzuckerwerts.

(*D*) **Verarbeitung von historischen Daten.** Neben einer strombasierten Verarbeitung der Sensordaten, müssen *IoT*-Anwendungen diese Daten häufig ebenfalls dauerhaft abspeichern, um auch Analysen auf historischen Daten durchführen zu können oder aus Gründen der Nachweispflicht. Das Datenschutzsystem muss daher auch in der Lage sein, etwa Verschleierungstechniken auf Bestandsdaten anwenden zu können. In dem gegebenen *E-Health*-Szenario könnte ein Anwendungsfall für eine solche Verarbeitung in der Langzeitanalyse von Gesundheitswerten bestehen, z. B. um einen Gesamtüberblick über einen Krankheitsverlauf zu erhalten.

(*E*) **Datenschutzdomänen.** Mehrere Parteien können eigene Anwendungen haben, die von einer gemeinsamen *IoT*-Anwendung mit Daten versorgt werden. Da für jede Partei unterschiedliche Berechtigungen gelten können, muss ein *IoT*-Datenschutzsystem unterschiedliche Datenschutzdomänen unterstützen, für die jeweils eigene Datenschutzregeln angewandt werden. In dem gegebenen *E-Health*-Szenario könnte für die Domäne „Arzt“ gelten, dass dessen Anwendungen unbeschränkten Zugriff auf sämtliche Daten erhalten, während für die Domäne „Eltern“ gilt, dass diese ausschließlich in einem Notfall auf die Standortdaten Zugriff bekommen.

(*F*) **Verifikation.** Ein Nutzer muss einem Datenschutzsystem vollständig vertrauen können. Da er allerdings nicht sehen kann, ob das System seine Daten entsprechend seiner Vorgaben schützt, ist eine stetige Verifikation erforderlich. D. h. das Datenschutzsystem muss überwachen, welche Daten an eine Anwendung weitergegeben werden und ob sich daraus eines der vom Nutzer als zu schützend spezifizierten Muster ableiten lässt.

(*G*) **Flexibel.** Die Muster lassen sich in der Regel auf unterschiedliche Arten verschleiern, die je nach Anwendung und Situation unterschiedlich gut geeignet sind. Das Datenschutzsystem muss daher in der Lage sein, unterschiedliche Verschleierungstechniken zu unterstützen und diese abhängig vom jeweiligen Anwendungsfall anwenden. In dem gegebenen *E-Health*-Szenario könnte beispielsweise eine Strategie für die Verschleierung von Bewegungsmustern so aussehen, dass die Ortsdaten verfälscht werden, und eine andere, dass die Reihenfolge der besuchten Orte verändert wird. Während die Verfälschung der Ortsdaten für die Versicherung sinnvoll wäre, da diese keine Information über die genauen Aufenthaltsorte benötigt, kann die Veränderung der Reihenfolge für die Städtebauer angewandt werden, da diese zwar die exakten Ortsangaben benötigen, sie aber nicht wissen müssen, wann die Orte besucht wurden.

(*H*) **Servicequalität erhalten.** Der Nutzer ist trotz der Verschleierung der von ihm definierten Muster dennoch an einer hohen Servicequalität interessiert. Die Servicequalität drückt in diesem Fall aus, wie viel Nutzen von einer Anwendung ausgeht. Werden einer Anwendung beispielsweise überhaupt keine Daten zur Verfügung gestellt, so ist zwar garantiert, dass diese keine der Muster erkennen kann, allerdings kann eine Anwendung ohne Daten auch nicht arbeiten. Das Datenschutzsystem muss daher darauf achten, dass es Verschleierungstechniken wählt, die zwar die kritischen Muster verbergen, die Anwendung aber dennoch mit einem Maximum an Informationen versorgt. In dem gegebenen *E-Health*-Szenario könnte beispielsweise ein Muster

lauten „Die Eltern dürfen nicht erfahren, dass das Kind vor der Schule bei einem Fastfood-Restaurant war.“ Anstelle den Eltern keine Standortdaten zu geben, reicht es aus, die Reihenfolge der Standorte zu vertauschen. Dadurch erscheint es, dass das Kind nach der Schule bei dem Restaurant war, wodurch das Muster nicht auftritt.

4 Verwandte Arbeiten

Unseres Wissens nach gibt es aktuell kein Datenschutzsystem für das *IoT*, das sich mit allen relevanten Aspekten befasst, d. h. einer nutzerfreundlichen Regelerstellung, der Verifikation der Regeln sowie Datenschutzmechanismen für Datenströme und historische Daten. Im Folgenden werden daher die verwandten Arbeiten in diesen vier Aspekten separat betrachtet.

Die **Spezifikation von Datenschutzregeln** ist für den durchschnittlichen Nutzer sehr schwierig, da er die Konsequenzen seiner Einstellungen nicht überblicken kann [Fe11]. Zur Definition von Sicherheitsanforderungen gibt es mit *STPA-Sec* [YL13] einen Ansatz mit dem sich die Schwachstellen einer Anwendung aufzeigen lassen. Der Nutzer kann anschließend geeignete Maßnahmen wählen, um diese Schwachstellen zu beseitigen. *STPA-Priv* [Sh16] passt diesen systemtheoretischen Ansatz dahingehend an, dass auch Datenschutzrisiken damit identifiziert werden können. Die eigentliche Erstellung der Datenschutzregeln oder gar deren Durchsetzung wird in *STPA-Priv* nicht berücksichtigt.

Zur **Verifikation der Effektivität von Datenschutzregeln** gibt es Ansätze, die an den traditionellen Softwaretestprozess angelehnt sind. Martin und Xie [MX07] stellen ein Modell vor, das Beispieldaten an eine Anwendung schickt und deren Ausgaben dahingehend untersucht, ob die zum Einsatz kommenden Datenschutzregeln alle vertraulichen Daten herausgefiltert haben. Weicht die Ausgabe von der erwarteten Ausgabe ab, so sind die Regeln entweder zu strikt (Ausgabemenge ist kleiner als erwartet) oder nicht ausreichend restriktiv (Ausgabemenge beinhaltet vertrauliche Daten). Liegt von der Anwendung ein formales Modell vor, so lassen sich dessen Sicherheitseinstellungen ebenfalls mit Techniken aus der Aussagelogik automatisch überprüfen [AC08]. All diese Techniken sind allerdings nicht in der Lage, komplexere Muster, wie sie im *IoT*-Umfeld benötigt werden, oder deren Auswirkungen auf die Servicequalität zu erkennen.

Es gibt viele Ansätze, mit denen die **Verarbeitung von Datenströmen** reguliert werden kann (z. B. *DEFCON* [Mi10] oder die Arbeit von Lindner und Meier [LM06]). Diese fokussieren sich allerdings auf einzelne Attribute, die bestimmten Operatoren nicht zur Verfügung stehen dürfen. Wie in dem *E-Health*-Szenario gezeigt, ist dies allerdings zu restriktiv. Vielmehr sollen manche Werte nur unter bestimmten Bedingungen nicht zur Verfügung gestellt werden. Systeme wie *ACStream* [Ca09] berücksichtigen daher auch den Kontext unter dem ein Operator ein Datum verarbeiten möchte. Die Zugriffskontrolle erfolgt allerdings auch hier auf Attributen und nicht auf komplexeren Mustern.

Die **Verarbeitung von Daten in Datenbanksystemen** kann über feingranulare Zugriffskontrollmechanismen reguliert werden, wie sie beispielsweise von Wang et al. [Wa07]

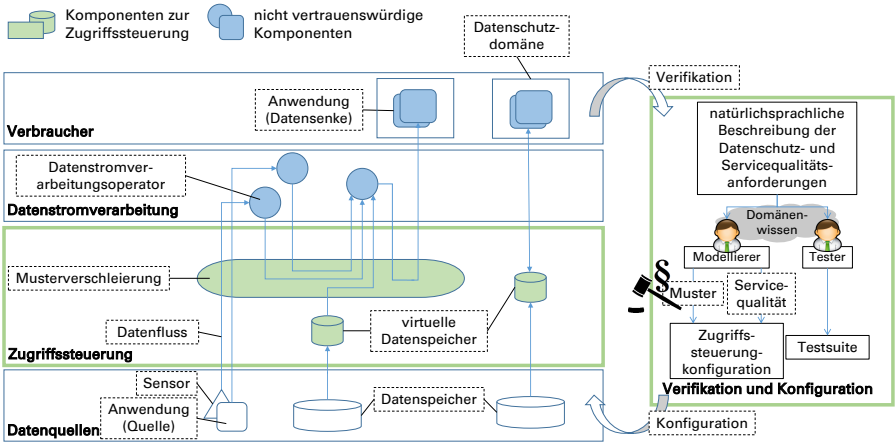

Abb. 2: Die PATRON-Architektur fürs *Internet der Dinge*

vorgestellt werden. Dabei werden Zugriffsrechte an bestimmte Nutzergruppen vergeben. Auch diese Regulierungen basieren auf einzelnen Attributen und nicht auf komplexen Mustern. Brown et al. [Br12] stellen Isolationstechniken für mandantenfähige Umgebungen vor, wie sie für Cloud-Anwendungen benötigt werden. Dabei liegt der Fokus auf der Isolation der Datenbestände einzelner Nutzer und nicht auf der Verschleierung bestimmter Muster.

5 Das PATRON-System

Abschnitt 4 zeigt, dass es aktuell kein umfassendes Datenschutzsystem für das *IoT*-Umfeld gibt. Mit PATRON führen wir ein solches System ein. Dabei wird die Standardarchitektur für *IoT*-Anwendungen um zwei Schichten erweitert: die **Verifikations- und Konfigurationsschicht** und die **Zugriffssteuerungsschicht** (siehe Abb. 2). Während die Verifikations- und Konfigurationsschicht orthogonal zu den restlichen Schichten steht, wird die Zugriffssteuerungsschicht zwischen die Datenquellen- und Verarbeitungsschicht eingebunden.

Die Verifikations- und Konfigurationsschicht stellt die Schnittstelle zum Nutzer dar. Hier kann er seine Datenschutzanforderungen und die gewünschte Servicequalität spezifizieren. Diese Spezifikation wird anschließend in eine Konfiguration für die Zugriffssteuerungsschicht umgewandelt. Zusätzlich findet in dieser Schicht die Verifikation der Datenschutzeinstellungen statt. Abschnitt 5.1 geht auf Details zu dieser Schicht ein. Die Zugriffssteuerungsschicht setzt die Datenschutzeinstellungen um. Hierfür erfolgt jeder Datenaustausch zwischen einer Datenquelle (z. B. Sensoren, Anwendungen oder Datenspeicher) und einer Datensenke (z. B. Datenstromverarbeitungsoperatoren oder Anwendungen) über diese Schicht. Dadurch ist sie in der Lage, die Datenströme zu manipulieren, um vertrauliche Muster geheim zu halten. Welche Techniken dabei zum Einsatz kommen, ist in Abschnitt 5.2 näher beschrieben.

5.1 Verifikations- und Konfigurationsschicht

Um dem Nutzer eine möglichst einfache Konfiguration des Systems bieten zu können, formuliert er seine Anforderungen als natürlichsprachliche Beschreibung. Diese Beschreibung wird von einer Gruppe Modellierer analysiert und bestimmt, welche Muster Informationen preisgeben können, die den Anforderungen widersprechen. Hierfür ist Domänenwissen erforderlich, da beispielsweise ein Experte auf dem Gebiet *Industrie 4.0* in der Regel nicht weiß, welche Informationen sich aus unterschiedlichen Gesundheitsdaten ableiten lassen. Das Ergebnis dieser Analyse hält jeder Modellierer in einer Menge von Mustern, die es geheim zu halten gilt, sowie einer Beschreibung der benötigten Servicequalität fest. Diese Artefakte sind in einer formalen Sprache formuliert und können daher die Zugriffsschicht automatisch konfigurieren. Bei der Beschreibung der Servicequalität wird darauf eingegangen, wie hoch die Rate der *False Positives* bzw. *False Negatives* maximal sein darf. D. h. wie häufig darf eine Anwendung durch die Veränderung der Daten eine Situation falsch einschätzen. Ein *False Positive* würde bedeuten, dass eine Anwendung ein Ereignis entdeckt (z. B. „Es kam zu einem Zuckerschock“), das nicht aufgetreten ist, während *False Negative* bedeuten, dass ein aufgetretenes Ereignis nicht erkannt wurde (z. B. ein realer Zuckerschock wurde nicht erkannt). Aus den so erstellten Modellen wird eines ausgesucht, das zur Konfiguration des Systems herangezogen wird, während die Restlichen als Testfälle zur Überprüfung der Güte des ausgewählten Modells genutzt werden. Diese Testfälle werden auf sämtliche Daten, die aus der Zugriffssteuerungsschicht geschickt werden, angewandt, und es wird geprüft, ob darin ein vertrauliches Muster enthalten ist. Hierdurch kann die Konfiguration von PATRON zuverlässig verifiziert werden.

Bei der Erstellung der Muster kann es zu zwei Problemen kommen. Einerseits können die Datenschutzanforderungen eines Nutzers ein Muster erforderlich machen, das eine Anwendung vollständig obsolet machen würde. Ein Beispiel hierfür ist, dass ein Nutzer das Muster „Zuckerschock“ verheimlichen will, während die Anwendung genau dies erkennen möchte. Dadurch kommt es zu einer *False-Negatives*-Rate von 100 % wodurch die Servicequalität minimal ist. Andererseits können die Muster gegen geltendes Recht verstoßen. Es muss beispielsweise geprüft werden, ob ein Nutzer Hinweise auf Vorerkrankungen vor seiner Versicherung geheim halten darf. Daher müssen die modellierten Muster durch Juristen, die in der jeweiligen Domäne spezialisiert sind, justifiziert werden. Während die Modellierer Werkzeugunterstützung erhalten können (z. B. mittels *STPA-Priv* [Sh16]) und die Umwandlung von der natürlichsprachlichen Beschreibung in die formalen Muster zumindest halbautomatisch erfolgen kann, erfolgt die juristische Prüfung rein manuell, da es sich hierbei in der Regel um Einzelfallentscheidungen handelt.

5.2 Zugriffssteuerungsschicht

In der Zugriffssteuerungsschicht werden die definierten Muster herausgefiltert. Zu diesem Zweck kommen unterschiedliche Techniken zum Einsatz. Die Zugriffssteuerungsschicht

wählt die jeweils passende Technik aus, um eine möglichst hohe Servicequalität zu erzielen, d. h. möglichst wenige *False Positives* bzw. *False Negatives* zu verursachen. Die durch den Modellierer angegebene Servicequalität stellt dabei eine untere Schranke dar.

Zur Realisierung dieser Filtertechniken werden in PATRON in der Zugriffssteuerungsschicht zwei Komponenten eingeführt: die Musterverschleierungskomponente, die vorgegebene Muster aus einem Datenstrom entfernt und virtuelle Datenspeicher, die Daten langfristig speichern können und bei Anfragen ebenfalls keine geheimen Muster preisgeben. Im Folgenden werden diese beiden Komponenten näher beschrieben.

Musterverschleierung. Die Musterverschleierung wird durch Komponenten in der Zugriffssteuerungsschicht realisiert, die Datenquellen von Datensenken trennen. Eine Senke kann sowohl eine Anwendung als auch ein Operator in einem *DSS* sein. Der Datenfluss wird in den Komponenten analysiert und daraufhin untersucht, ob sich daraus die zu schützenden Muster ableiten lassen. Dabei werden nicht nur die aktuellen Daten berücksichtigt sondern auch sämtliche Daten, die in der Vergangenheit an die Senke weitergeleitet wurden. Zur Verschleierung der Muster stehen diesen Komponenten mehrere Möglichkeiten zur Verfügung, z. B. indem bestimmte Daten nicht weitergeleitet werden, die zeitliche Abfolge der Daten verändert wird oder die Daten durch Rauschen verfremdet werden. Welche dieser Techniken angewandt wird, hängt von der Anwendung ab, an die die Daten weitergeleitet werden. Hierbei muss berücksichtigt werden, wie die Daten verarbeitet werden (z. B. hat eine Änderung an der zeitlichen Abfolge der Daten keinen Einfluss, wenn diese anschließend in aggregierter Form weiterverarbeitet werden) und welchen Einfluss die Änderung auf die Servicequalität der Anwendung hat (z. B. kann ein Muster $\{A; B\}$ dadurch verschleiert werden, indem das Datum A nicht weitergeleitet wird; dies sorgt aber auch dafür, dass $\{A; C\}$ nicht mehr erkannt werden kann, obwohl dieses Muster unkritisch ist). Daher werden die Techniken stets so ausgewählt, dass alle kritischen Muster verschleiert werden, der Einfluss auf die Servicequalität allerdings minimal ist.

Virtuelle Datenspeicher. Virtuelle Datenspeicher sind geschützte Container, die Anwendungen oder einem *DSS* Daten zur Verfügung stellen. Nach außen verhalten sie sich wie herkömmliche Datenbanken. In den virtuellen Datenspeichern werden Daten aus Quellsystemen (sowohl Sensordaten als auch Daten aus Datenbanken) vorgehalten. Diese Daten werden bei der Einspeisung bereits an die Datenschutzeinstellungen angepasst, d. h. die Daten werden bereits teilweise verschleiert oder aggregiert abgespeichert. Anwendungen können Daten direkt von dem virtuellen Datenspeicher anfordern. In diesem Fall werden die eingehenden Anfragen dahingehend umgeschrieben, dass keine der vorgegebenen Muster erkannt werden können. Zusätzliche Schutzmaßnahmen (z. B. Datenverschlüsselung) sorgen dafür, dass die Daten der Nutzer geschützt sind (siehe Stach und Mitschang [SM15; SM16]). Darüber hinaus können die virtuellen Datenspeicher aber auch mit den Operatoren des *DSSs* zusammenarbeiten. Ähnlich wie in der Kappa-Architektur [Kr14] dient der Datenspeicher als Quellsystem für das *DSS*. Zusätzlich zu den Daten liegt im virtuellen Datenspeicher aber auch eine Konfiguration für das *DSS*, d. h. die anzuwendenden Musterverschleierungstechniken, vor. Dadurch kann das *DSS* mit den Daten der Datenspeicher arbeiten, als ob

diese von einem weiteren Sensor stammen würden, und es können sich eventuell ergebende Muster herausgefiltert werden.

Dass beide Komponenten nötig sind, zeigt sich an dem beschriebenen *E-Health*-Szenario. So kann es für eine Analyse nötig sein, dass aktuelle Gesundheitswerte unter Berücksichtigung des Krankheitsverlaufs des letzten Jahrs untersucht werden müssen. Hierfür sind sowohl Sensordaten als auch historische Daten aus einem Datenspeicher nötig. Nur durch die Integration beider Komponenten in der Zugriffssteuerungsschicht kann sichergestellt werden, dass nicht nur jede Komponente für sich betrachtet keine Muster preisgibt, sondern auch, dass dies durch die Kombination der beiden nicht der Fall ist.

6 Evaluation

Im Folgenden soll untersucht werden, inwiefern PATRON die in Abschnitt 3 aufgestellten Anforderungen an ein Datenschutzsystem für das *IoT* erfüllt.

Aufgrund der natürlichsprachlichen Beschreibung der Anforderungen durch den Nutzer ist die Konfiguration von PATRON sehr einfach (*A*). Er kann hierbei Zusammenhänge beschreiben, die durch Modellierer in komplexe Muster übersetzt werden. Dadurch ist der Schutz nicht auf einzelne Attribute beschränkt, was zu restriktiv für ein solches Datenschutzsystem wäre (*B*). Die Zugriffssteuerungsschicht kann sowohl Sensordaten aus Strömen als auch historische Daten aus Datenspeichern verarbeiten und Muster darin verbergen (*C*) und (*D*). Zusätzlich ist sie durch die enge Integration der beiden Verarbeitungsarten in der Lage, Muster zu verbergen, die sich erst durch die Kombination von Daten aus beiden Quellen ergeben. Dadurch dass Domänenexperten die Muster aus den Anforderungen ableiten, sind diese stets auf die jeweilige Domäne angepasst. D. h. die gleichen Datenschutzanforderungen eines Nutzers führen je nach Domäne zu unterschiedlichen Mustern. Dadurch kann die Servicequalität maximiert werden (*E*). Die Güte der definierten Muster wird dauerhaft überprüft, indem die freigegebenen Daten gegen definierte Testfälle abgeglichen werden. Auf diese Weise kann die Konfiguration des Systems verifiziert werden (*F*). Die Muster legen dabei nur fest was, aber nicht wie geschützt werden soll. Dadurch bleibt PATRON sehr flexibel und kann unterschiedliche Techniken zum Schutz der Muster nutzen, z. B. eine Änderung der Datenreihenfolge oder eine Veränderung von bestimmten Werten (*G*). Aufgrund dieser Flexibilität ist PATRON in der Lage, die Servicequalität zu maximieren, da stets die Technik ausgewählt wird, mit der die zu schützenden Muster verheimlicht werden und die Anwendung dennoch mit der bestmöglichen Datenqualität versorgt werden kann (*H*). PATRON erfüllt somit sämtliche in Abschnitt 3 aufgestellte Anforderungen.

7 Zusammenfassung

IoT-Anwendungen werden zunehmend komplexer bezüglich der darin verwendeten Sensordaten und deren Verknüpfungen. Dadurch lassen sich weitreichende Rückschlüsse über die

Nutzer ziehen. Aus diesem Grund sind umfassende Datenschutzmaßnahmen erforderlich. Der Gesetzgeber reagiert darauf mit einer Verschärfung der EU-Datenschutzgrundverordnung. Aus Nutzersicht sind allerdings auch technische Maßnahmen erforderlich, mit denen der Nutzer die Kontrolle über seine Daten erhält. Die aktuell verfügbaren Datenschutzsysteme sind allerdings nicht auf die speziellen Anforderungen von *IoT*-Anwendungen zugeschnitten.

Mit PATRON stellen wir ein neuartiges Datenschutzsystem für das *IoT* vor. Damit kann der Nutzer sehr einfach seine Datenschutzanforderungen beschreiben. Diese werden von Experten in Muster transformiert und auf Echtzeitdaten und historischen Daten angewandt. Dabei stehen unterschiedliche Algorithmen zur Verfügung. PATRON wählt stets den Algorithmus aus, der die Anwendung mit den besten respektive meisten Daten versorgt, damit dem Nutzer die bestmögliche Servicequalität geboten wird. PATRON analysiert ebenfalls sämtliche Daten, die einer Anwendung zur Verfügung gestellt werden, um verifizieren zu können, dass die zu verheimlichenden Muster daraus nicht abgeleitet werden können. Anhand eines *E-Health*-Szenarios wurde unser Ansatz evaluiert und gezeigt, dass er alle Anforderungen an ein Datenschutzsystem erfüllt. Damit ist der Nutzer in der Lage, jedes beliebige Muster gegenüber einer Anwendung zu verschleiern. Dies wirft allerdings die Frage auf, ob er dies aus rechtlicher Sicht auch darf. Einerseits ist zu klären, wer der Besitzer der Daten ist (z. B. geben Kalendereinträge eines Nutzers auch oft Informationen über andere Teilnehmer an diesen Terminen preis), und andererseits muss geklärt werden, welche Muster nicht verheimlicht werden dürfen (z. B. Hinweise auf Vorerkrankungen gegenüber einer Krankenkasse).

Danksagung

Wir danken der Baden-Württemberg Stiftung für die Förderung der in diesem Artikel vorgestellten Forschungsarbeiten. Beim PARTON-Projekt handelt es sich um einen Forschungsauftrag, der aus Mitteln der Baden-Württemberg Stiftung finanziert wird.

Literatur

[AC08] Armando, A.; Compagna, L.: SAT-based model-checking for security protocols analysis. International Journal of Information Security 7/1, S. 3–32, 2008.

[Al12] Almaghrabi, R. et al.: A novel method for measuring nutrition intake based on food image. In: I2MTC ’12. S. 366–370, 2012.

[BL01] Ball, M. J.; Lillis, J.: E-health: transforming the physician/patient relationship. International Journal of Medical Informatics 61/1, S. 1–10, 2001.

[Br12] Brown, W. J. et al.: Multitenancy - Security Risks and Countermeasures. In: NBIS ’12. S. 7–13, 2012.

[Ca09] Cao, J. et al.: ACStream: Enforcing Access Control over Data Streams. In: ICDE '09. S. 1495–1498, 2009.

[Fe11] Felt, A. P. et al.: The Effectiveness of Application Permissions. In: WebApps '11. S. 1–12, 2011.

[Kn10] Knöll, M.: "On the Top of High Towers . . . " Discussing Locations in a Mobile Health Game for Diabetics. In: MCCSIS '10. S. 61–68, 2010.

[Kr14] Kreps, J.: Questioning the Lambda Architecture, O'Reilly Media, Juli 2014, URL: https://www.oreilly.com/ideas/questioning-the-lambda-architecture.

[Ku13] Kumar, S. et al.: Mobile Health: Revolutionizing Healthcare Through Transdisciplinary Research. Computer 46/1, S. 28–35, 2013.

[Kw11] Kwapisz, J. R. et al.: Activity Recognition Using Cell Phone Accelerometers. ACM SIGKDD Explorations Newsletter 12/2, S. 74–82, 2011.

[LM06] Lindner, W.; Meier, J.: Securing the Borealis Data Stream Engine. In: IDEAS '06. S. 137–147, 2006.

[Me12] Mehta, D. D. et al.: Mobile Voice Health Monitoring Using a Wearable Accelerometer Sensor and a Smartphone Platform. IEEE Transactions on Biomedical Engineering 59/11, S. 3090–3096, 2012.

[Mi10] Migliavacca, M. et al.: DEFCON: High-performance Event Processing with Information Security. In: USENIXATC '10. S. 1–15, 2010.

[MX07] Martin, E.; Xie, T.: A Fault Model and Mutation Testing of Access Control Policies. In: WWW '07. S. 667–676, 2007.

[Sh16] Shapiro, S. S.: Privacy Risk Analysis Based on System Control Structures: Adapting System-Theoretic Process Analysis for Privacy Engineering. In: SPW '16. S. 17–24, 2016.

[SM15] Stach, C.; Mitschang, B.: Der Secure Data Container (SDC) – Sicheres Datenmanagement für mobile Anwendungen. Datenbank-Spektrum 15/2, S. 109–118, 2015.

[SM16] Stach, C.; Mitschang, B.: The Secure Data Container: An Approach to Harmonize Data Sharing with Information Security. In: MDM '16. S. 292–297, 2016.

[SS12] Stach, C.; Schlindwein, L. F. M.: Candy Castle — A Prototype for Pervasive Health Games. In: PerCom '12. S. 501–503, 2012.

[St16] Stach, C.: Secure Candy Castle — A Prototype for Privacy-Aware mHealth Apps. In: MDM '16. S. 361–364, 2016.

[Wa07] Wang, Q. et al.: On the Correctness Criteria of Fine-grained Access Control in Relational Databases. In: VLDB '07. S. 555–566, 2007.

[YL13] Young, W.; Leveson, N.: Systems Thinking for Safety and Security. In: ACSAC '13. S. 1–8, 2013.