Konzepte zum Schutz privater Muster in Zeitreihendaten

IoT-Anwendungen im Spannungsfeld zwischen Servicequalität und Datenschutz

Christoph Stach[1]

Abstract: Obwohl das *Internet der Dinge* (*IoT*) die Voraussetzung für *smarte* Anwendungen schafft, die signifikante Vorteile gegenüber traditionellen Anwendungen bieten, stellt die zunehmende Verbreitung von IoT-fähigen Geräten auch eine immense Gefährdung der Privatheit dar. IoT-Anwendungen sammeln eine Vielzahl an Daten und senden diese zur Verarbeitung an ein leistungsstarkes Back-End. Hierbei werden umfangreiche Erkenntnisse über den Nutzer gewonnen. Erst dieses Wissen ermöglicht die Servicevielfalt, die IoT-Anwendungen bieten. Der Nutzer muss daher einen Kompromiss aus Servicequalität und Datenschutz treffen. Heutige Datenschutzansätze berücksichtigen dies unzureichend und sind dadurch häufig zu restriktiv. Aus diesem Grund stellen wir neue Konzepte zum Schutz privater Daten für das IoT vor. Diese berücksichtigen die speziellen Eigenschaften der im IoT zum Einsatz kommenden *Zeitreihendaten*. So kann die Privatheit des Nutzers gewährleistet werden, ohne die Servicequalität unnötig einzuschränken. Basierend auf den *TICK-Stack* beschreiben wir Implementierungsansätze für unsere Konzepte, die einem *Privacy-by-Design-Ansatz* folgen.

Keywords: Datenschutz; Zeitreihendaten; IoT; DSGVO; ePrivacy-Verordnung; TICK-Stack.

1 Einleitung

Das *Internet der Dinge* (engl. *Internet of Things*, kurz *IoT*) war lange Zeit nicht viel mehr als ein Modewort und Wissenschaft und Wirtschaft versuchten, das damit verbundene Potential anhand von kleineren Pilotprojekten auszuloten. Hierbei zeigte sich, dass die im IoT erfassten Sensordaten in vielen Anwendungsbereichen gewinnbringend eingesetzt werden können, wie beispielsweise im *Smart Home*, im *Gesundheitssektor* oder zur Realisierung der *Industrie 4.0*. Gartner Analysten sehen das Jahr 2017 jedoch als Gipfel des *Hype-Zyklus* für das IoT an und in den kommenden Jahren gilt es, die erarbeiteten *Proof-of-Concept-Projekte* mithilfe von marktfähigen Umsetzungen zu untermauern [Ve17]. Während die Verfügbarkeit von IoT-fähigen Geräten sowohl im privaten als auch im industriellen Umfeld bereits weitestgehend gegeben ist, gibt es softwareseitig noch viele offene Fragen.

Insbesondere die Frage, wie relevantes Wissen aus den Unmengen an verfügbaren Daten gewonnen werden kann, ist entscheidend für die Servicequalität einer IoT-Anwendung. Da IoT-fähige Geräte in der Regel nicht ausreichend Ressourcen besitzen, um selbstständig umfassende Analysen auszuführen, senden sie die erfassten Daten hierzu an eine zentrale

[1] Universität Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Deutschland, Christoph.Stach@ipvs.uni-stuttgart.de

© 2019 GI. This is the author's version of the work. It is posted at https://opencms.uni-stuttgart.de/fak5/ipvs/departments/as/publications/stachch/gi_19_time_series.pdf by permission of GI for your personal use. Not for redistribution. The definitive version was published in *In: David, K. et al. (Hrsg.), Informatik 2019: 50 Jahre Gesellschaft für Informatik - Informatik für Gesellschaft, Tagungsband der 49. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 23.09. - 26.09.2018, Kassel. Gesellschaft für Informatik, Bonn*, pp. 353–366, 2019, doi: 10.18420/inf2019_54.

Verarbeitungskomponente. Hier werden alle erfassten Daten miteinander verknüpft, bereinigt und mittels *Data-Mining-* respektive *Machine-Learning-Techniken* analysiert. Damit die Verarbeitung der Daten echtzeitnah erfolgen kann, muss das Datenvolumen frühzeitig reduziert werden. Hierfür lassen sich die speziellen Eigenschaften dieser *Zeitreihendaten* ausnutzen. Anstelle sämtliche Datenpunkte zu analysieren, genügt es beispielsweise häufig, in den Daten enthaltene charakteristische Muster zu betrachten (z. B. Extremwerte) [Ma17].

Eine noch größere Herausforderung stellt jedoch der Schutz privater Daten dar. Einerseits leben innovative IoT-Anwendungen davon, uneingeschränkten Zugriff auf Nutzerdaten zu erhalten und auf diese Weise ihre Services auf die Bedürfnisse des Nutzers[2] auszurichten. Andererseits stellt just dieser uneingeschränkte Zugriff in Kombination mit den nahezu grenzenlosen Ressourcen, die zur Analyse der Daten zur Verfügung stehen, eine ernstzunehmende Gefährdung der Privatsphäre dar. Während aus rechtlicher Sicht beispielsweise die *DSGVO* den Zugriff und die Verarbeitung von personenbezogenen Daten regelt, bedarf es technischer Lösungen zur Sicherstellung der Datenschutzbestimmungen. Es muss dem Nutzer möglich sein, einfach festzulegen, welche seiner Daten wofür verwendet werden dürfen und ebenfalls, welche Informationen nicht preisgegeben werden dürfen [Zh18].

Hierbei zeigt sich das Spannungsfeld, dem sich IoT-Anwendungen ausgesetzt sehen: Ein Nutzer muss zwischen Servicequalität und Datenschutz abwägen. Je mehr Daten er einer IoT-Anwendung zur Analyse bereitstellt, desto mehr Funktionalität kann sie ihm anbieten – gleichzeitig werden allerdings auch viele private Informationen offengelegt. Erhält eine IoT-Anwendung hingegen keine Daten, ist die Privatheit des Nutzers geschützt, aber die Anwendung wird ineffektiv. Ziel muss es daher sein, einen guten Kompromiss zwischen Servicequalität und Datenschutz zu finden. Hierbei kann davon profitiert werden, dass die erfassten Daten im Rahmen der Analyse zur Reduktion des Datenvolumens ohnehin verdichtet werden müssen. Durch eine geschickte Wahl des Verdichtungsverfahrens, können private Informationen verborgen werden, ohne die Analyseergebnisse zu kompromittieren.

Zu diesem Zweck erbringen wir die folgenden drei Beiträge: (1) Wir stellen sechs Konzepte zum Schutz privater Muster in Zeitreihendaten vor, die die speziellen Eigenschaften dieser Daten ausnutzen. Mithilfe unserer Konzepte ist es möglich, die Privatheit des Nutzers zu gewährleisten und gleichzeitig die Servicequalität von IoT-Anwendungen nicht unnötig einzuschränken. (2) Wir bewerten die eingeführten Konzepte und schätzen ab, welches Konzept sich für welchen Anwendungsfall sich am besten eignet. (3) Wir beschreiben auf den *TICK-Stack*[3] basierende Implementierungsansätze für unsere Schutzkonzepte.

Der Rest dieses Artikels ist wie folgt gegliedert: In Abschnitt 2 wird der Stand der Technik bezüglich IoT-Anwendungen anhand eines Smart-Home-Anwendungsfalls beschrieben. Datenschutzansätze fürs IoT werden in Abschnitt 3 diskutiert. In Abschnitt 4 führen wir neue Datenschutzkonzepte ein und evaluieren diese. Wir diskutieren Implementierungsansätze für diese Schutzkonzepte in Abschnitt 5. Abschließend fasst Abschnitt 6 den Artikel zusammen.

[2] Mit dem Begriff „Nutzer“ seien im Folgenden jeweils alle Geschlechter gleichermaßen adressiert.
[3] siehe https://www.influxdata.com/time-series-platform/

2 Stand der Technik

Im Nachfolgenden führen wir ein Smart-Home-Anwendungsbeispiel ein. Anhand dieses Beispiels beschreiben wir anschließend die technischen Grundlagen, die eine solche IoT-Anwendung ermöglichen, und gehen darauf ein, welche rechtlichen Rahmenbedingungen die DSGVO sowie die *ePrivacy-Verordnung* für solche Anwendungen schaffen.

Anwendungsbeispiel: In Anlehnung an Marikyan et al. [Ma19] eruierten wir folgende drei Nutzungsmöglichkeiten für IoT-Anwendungen in einem Smart Home. So können IoT-fähige Geräte ihre Nutzer bei der *Bewältigung alltäglicher Aufgaben* unterstützen. Hierzu werden mittels Sensoren die gegenwärtigen Aktivitäten der Nutzer erfasst und Aktuatoren reagieren darauf. Denkbare Anwendungsszenarien hierfür sind, dass Senioren automatisch darauf hingewiesen werden können, wenn die Aktivität „Einnahme von Medikamenten“ ausblieb oder dass die Heizung automatisch deaktiviert wird, wenn ein Nutzer ein Fenster öffnet.

IoT-Technologien lassen sich auch dafür nutzen, die *empfundene Lebensqualität* der Nutzer zu steigern. Kameras können besondere Ereignisse, wie beispielsweise eine Party, in Bild und Ton festhalten. Nutzer können diese Aufnahmen indexieren. So ist das Smart Home in der Lage, darauf abgebildete Personen zu erkennen oder neue Aufnahmen automatisch bestimmten Rubriken zuzuordnen. Auf digitalen Fotorahmen können situationsabhängig passende Bilder anzeigen (z. B. abhängig von den anwesenden Personen).

Schließlich stellt auch die *Überwachung* des Smart Homes eine Nutzungsmöglichkeit dar. Hat eine IoT-Anwendung alle typischen Aktivitäten der ihr bekannten Nutzer erfasst, kann sie ebenfalls Abweichungen von diesen Verhaltensmustern erkennen. Liegt ein älterer Nutzer beispielsweise an einem ungewöhnlichen Ort, könnte er gestürzt sein und eine Notsituation vorliegen, oder verschafft sich ein unbekannter Nutzer Zugang zum Haus, so könnte es sich um einen Einbruch handeln. In beiden Fällen, kann automatisch Hilfe gerufen werden.

Es ist offensichtlich, dass solche Anwendungen trotz des großen Nutzungspotentials Bedenken bezüglich des Datenschutzes aufwerfen. All diese Services sind nur möglich, wenn die eingesetzten IoT-fähigen Geräte dauerhaft Daten erfassen, austauschen und verarbeiten können. Da aufgrund der großen Datenmenge die verfügbaren Ressourcen weder für die Speicherung noch die Verarbeitung ausreichen, erfolgt dies häufig auf externen Servern des Serviceanbieters. Nutzer müssen sich daher entscheiden, welche Daten sie bereit sind preiszugeben, um im Gegenzug welche Serviceleistungen zu erhalten [Zh18].

Technische Grundlagen: Um eine solche IoT-Anwendung zu ermöglichen, bedarf es aus technischer Sicht zwei wesentliche Komponenten. Zum einen muss es möglich sein, *große Datenmengen* zu verarbeiten (z. B. um historische Daten zu analysieren und darin enthaltene Verhaltensmuster zu lernen) und zum anderen müssen *Daten in Echtzeit* verarbeitet werden können (z. B. um unmittelbar auf festgestellte Muster in den Live-Daten zu reagieren).

Für derartige Anwendungsfälle hat sich die *Lambda-Architektur* [MW15] als De-Facto--Standard durchgesetzt. Diese ist in Abb. 1 dargestellt. Beliebige Datenquellen, wie *Sensoren*

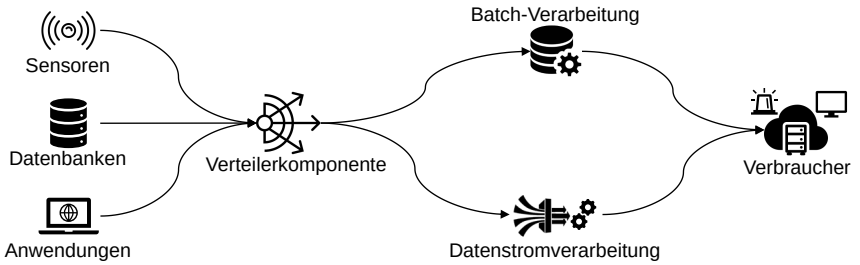


Abb. 1: Die *Lambda-Architektur* in Anlehnung an Marz und Warren [MW15].

(respektive IoT-fähige Geräte), *Datenbanken* oder *Anwendungen* senden ihre Daten an eine *Verteilerkomponente*. Diese reichert eingehende Daten um einen Zeitstempel sowie ein *Topic* an, das den Kontext beschreibt, in dem die Daten erfasst wurden. Aufgrund des Zeitstempels sind sämtliche Daten ab diesem Punkt Zeitreihendaten. Für die Langzeitarchivierung und Historisierung, werden die Daten in einer Datenbank abgelegt. In dieser werden die Daten en gros als *Batch* verarbeitet, wodurch eine hohe Genauigkeit der Analyseergebnisse erzielt wird. Allerdings dauert die Datenverarbeitung dadurch auch mehrere Stunden. Beispielsweise können auf diese Weise Muster gelernt werden, die beschreiben, wie eine bestimmte Aktivität erkannt werden kann. Für die eigentliche Mustererkennung ist dieses Vorgehen aufgrund der hohen Laufzeit jedoch ungeeignet. Daher werden eingehende Daten parallel dazu auch an eine *Datenstromverarbeitungskomponente* geschickt. Hier werden die gelernten Muster auf neue Daten innerhalb eines beschränkten Zeitfensters angewandt. Aufgrund der geringen Datenmenge, die dabei zu jedem Zeitpunkt berücksichtigt wird, sind zwar keine umfassenden Analysen möglich, die Berechnung kann jedoch sehr schnell erfolgen (*echtzeitnah*). Beide Verarbeitungskomponenten stellen ihre Ergebnisse ausgewählten *Verbrauchern* (z. B. Aktuatoren) zur Verfügung. Eine Implementierungsvariante der Lambda-Architektur ist mit dem TICK-Stack in Abschnitt 5 beschrieben.

Rechtliche Situation: Es ist offensichtlich, dass IoT-Anwendungen davon profitieren, wenn sie möglichst viele Daten über den Nutzer erfassen und verarbeiten können. Wachter [Wa18] untersucht daher, inwiefern sich derartige Anwendungen mit der DSGVO vereinbaren lassen. Es ist offensichtlich, dass sich Artikel 5 hierbei als problematisch erweist, da sich eine *Datenminimierung* hochgradig negativ auf die Servicequalität auswirkt. Auch werden Sensordaten im IoT dauerhaft gesammelt, während sich neue Verwendungszwecke erst mit der Zeit ergeben (*Zweckbindung*). Überhaupt ist es quasi unmöglich eine *Transparenz* zu schaffen, da aus den Daten Modelle gelernt werden, die für Menschen nicht nachvollziehbar sind. Artikel 22 reguliert *automatisierte Entscheidungen* und *Profiling*. Da IoT-Anwendungen ohne diese beiden Techniken nicht auskommen, ist eine ausdrückliche *Einwilligung* erforderlich. Die Einwilligungsanfrage muss in *klarer und einfacher Sprache* erfolgen (Artikel 7), was sich aufgrund der komplexen Verarbeitungsweise als zusätzlich schwierig erweist. Die ePrivacy-Verordnung weitet diese Schutzanforderungen außerdem auf *nicht-personenbezogene Daten* sowie *Daten von juristischen Personen* aus [Vo17].

3 Verwandte Arbeiten

Wie der Blick auf den Stand der Technik zeigt, machen die DSGVO und die ePrivacy-Verordnung grundlegende Änderungen an IoT-Anwendungen erforderlich, insbesondere da es für die Anbieter dieser Anwendungen nahezu unmöglich ist, nachträglich nachzuvollziehen, aufgrund welcher Daten eine Aktion ausgelöst wurde. Eine Lösung kann Artikel 25 der DSGVO darstellen. Hier werden explizit *technische Maßnahmen* gefordert, die die Einhaltung des Datenschutzes gewährleisten, d. h. eine IoT-Anwendung soll sich selbst überwachen und regulieren. Wir untersuchen daher, welche Datenschutztechniken aktuell im IoT-Kontext eingesetzt werden und bewerten diese in Hinblick auf den obigen Anwendungsfall.

Zugriffskontrollverfahren: Bourgeois et al. [Bo18] schlagen daher den Einsatz eines rollenbasierten Zugriffskontrollsystems vor. Hierbei werden den an der IoT-Anwendung beteiligten Parteien bestimmte *Rollen* zugeordnet (z. B. betroffene Person, Datenverantwortlicher oder Anfragender). Eine Partei kann auch mehrere Rollen innehaben und diese je nach Anwendungsfall dynamisch wechseln. Jeder Rolle sind Zugriffsrechte auf die Daten(quellen) einer IoT-Anwendung zugeteilt. Reicht diese starre Rollenstruktur nicht aus, können die Zugriffsberechtigungen auch an *Attribute* gebunden werden, die den Nutzer, die angefragten Daten oder den Kontext, in dem ein Nutzer eine Anfrage stellt, beschreiben [Ou16].

Während sich dieser Ansatz in Mehrbenutzersystemen bewährt hat, ergeben sich fürs IoT zwei entscheidende Nachteile: Zum einen kann es dabei leicht zu einer Identitätsfälschung kommen, da die Echtheit der übermittelten Attribute nicht überprüft werden kann [Bh19]. Hierfür wäre eine Verifikation der IoT-fähigen Geräte nötig, die die verwendeten Attribute beisteuern. Dies ist allerdings ressourcenintensiv und kann nicht auf den IoT-fähigen Geräte ausgeführt werden. Gritti et al. [Gr19] stellen daher ein Verfahren vor, mit dem diese Verifikation auf einen *Cloud-Dienst* ausgelagert werden kann. Hierdurch ergibt sich jedoch sowohl für den Anbieter als auch den Nutzer ein zusätzlicher Aufwand. Zum anderen setzt dieser Ansatz voraus, dass sämtliche (private) Daten an den IoT-Anbieter übergeben werden, d. h. sie verlassen den Einflussbereich des Nutzers. Ob und wie gut dort die Zugriffskontrollverfahren angewandt werden, ist für den Nutzer nicht ersichtlich [Bh19].

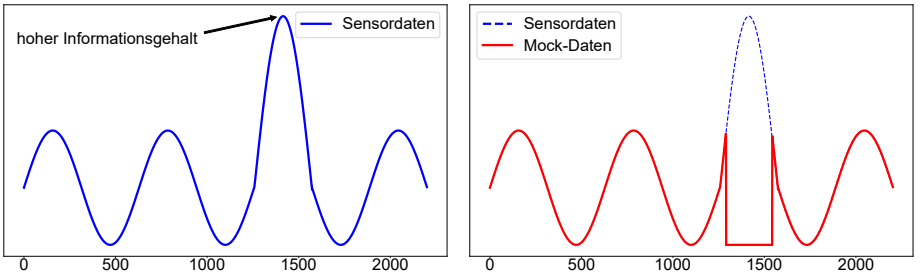
Attributbasierte Verfahren: Um dieser Problematik entgegenzuwirken, kann in der Verteilerkomponente oder der Datenquelle ein *Filter* integriert werden. So können die Daten eines ausgewählten Sensors aus dem Datenstrom herausgefiltert werden, wenn diese private Informationen beinhalten. Ein Nutzer könnte beispielsweise festlegen, dass die Daten eines GPS-Sensors nicht an eine IoT-Anwendung weitergegeben werden. Auch eine feingranulare Filterung ist möglich, indem nur bestimmte Attribute eines Sensors blockiert werden (z. B. der Längen- oder Breitengrad). Der Filter kann zusätzlich mit einem *Kontext* verknüpft werden, der beschreibt, wann der Filter aktiv sein soll (z. B. „nur am Wochenende“). Olejnik et al. [Ol17] stellen ein solches System für IoT-fähige Geräte vor, das ebenfalls für die Verteilerkomponente implementiert werden kann. Selbst wenn die Daten bereits an die Batch- oder Datenstromverarbeitung weitergereicht wurden, gibt es vergleichbare Ansätze, zur Filterung von Datenbankanfragen [PO12] und Datenströmen [Ad11].

Ein Problem, das jedoch all diesen Verfahren inhärent innewohnt, ist deren hohe Restriktivität. Werden beispielsweise sämtliche GPS-Sensordaten herausgefiltert, so können von da an keine standortbezogenen Dienste mehr angeboten werden. Selbst wenn ein Aktivierungskontext genutzt wird, sind diese Dienste immer dann dysfunktional, wenn der Filter aktiv ist.

Musterbasierte Verfahren: Aus diesem Grund führen Stach et al. [St18] ein musterbasiertes Verfahren zum Schutz privater Daten ein. Das Ziel dabei ist, privaten Informationen vor einer IoT-Anwendung zu schützen, ohne die Servicequalität unnötig einzuschränken. Zu diesem Zweck werden auf die Konzepte des *Complex Event Processings* (*CEP*) [EB09] zurückgegriffen. Im CEP werden nicht einzelne Sensorwerte betrachtet, sondern höherwertige Ereignisse, die durch eine Folge von Werten innerhalb eines Zeitfensters repräsentiert werden. Ein Beispiel für ein solches Ereignis ist „Nutzer kommt in ungeheizte Wohnung“, das sich aus „Temperatur ≤ 15 °C“ und „Nutzer nähert sich Smart Home“ zusammensetzt. Auf diese Weise definiert der Nutzer *private Muster*, die er nicht preisgeben möchte, sowie *öffentliche Muster*, die für die Servicequalität relevant sind. Durch eine zeitliche Umsortierung der Ereignisse innerhalb des Datenstroms werden die privaten Muster aufgebrochen und somit verborgen. Eine Qualitätsfunktion bewertet alle möglichen Permutationen. Hierbei wird die Anzahl der erfolgreich verborgenen privaten Muster der der *False Positives* (öffentliche Muster, die durch die Umsortierung entstanden sind) und der *False Negatives* (öffentliches Muster, die durch die Umsortierung verborgen werden) gegenübergestellt. Die Permutation mit der besten Bewertung wird anschließend auf den Datenstrom angewandt.

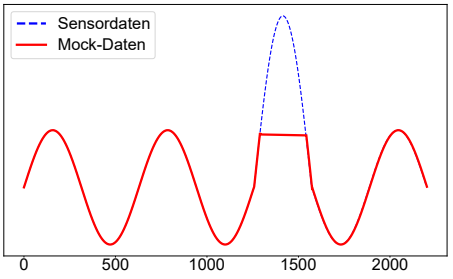
Es ist offensichtlich, dass aufgrund des musterbasierten Ansatzes dieses Verfahren erheblich weniger restriktiv ist. Dadurch ist es möglich, einer IoT-Anwendung Sensordaten weitestgehend unverändert zur Verfügung zu stellen und lediglich bei einigen wenigen den Zeitstempel zu manipulieren, wodurch die Servicequalität erhalten bleibt. Allerdings stellt dieser Ansatz kein Mittel gegen die unnötige Preisgabe von detaillierten Sensordaten dar. Wenn beispielsweise ein öffentliches Muster „Temperatur ≤ 15 °C“ lautet, müsste der Anwendung die exakte Temperatur nicht bekannt sein – eine binäre Aussage („erfüllt“ oder „nicht erfüllt“) wäre ausreichend. Dies wird jedoch bei den Mustern nicht berücksichtigt.

Statistische Verfahren: Auch *Differential Privacy* findet im IoT-Kontext Anwendung. Birman et al. [Bi15] stellen einen solchen Ansatz für *Smart Grids* vor. Hierbei verbleiben die Daten vollständig auf den *Smart Metern* der Nutzer, während das Versorgungsunternehmen nur aggregierte Daten erhält. Differential-Privacy-Techniken stellen sicher, dass diese Daten keine Rückschlüsse auf einen individuellen Smart Meter (d. h. auf einen Nutzer) zulassen, die Datenqualität für eine statistische Auswertung dennoch maximal ist. Eine derartige Anonymisierung lässt sich jedoch nur in solchen Szenarien anwenden, in denen ausschließlich statistische Informationen über eine Menge an Nutzern, nicht aber über einen individuellen Nutzer benötigt werden. Für Anwendungsfälle, wie das in Abschnitt 2 beschriebene Smart-Home-Szenario, eignet sich dieses Vorgehen daher nicht. Hier muss es möglich sein, einen individuellen Nutzer exakt zu bestimmen, um zu entscheiden, welcher seiner Aktuatoren wie auf die Daten eines bestimmten Sensors reagieren soll.

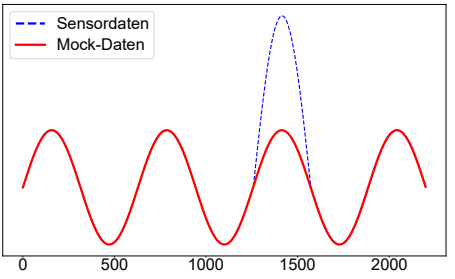


(a) Unveränderte Rohdaten.

(b) Nullwerte als Mock-Daten.



(c) Lineare Interpolation.



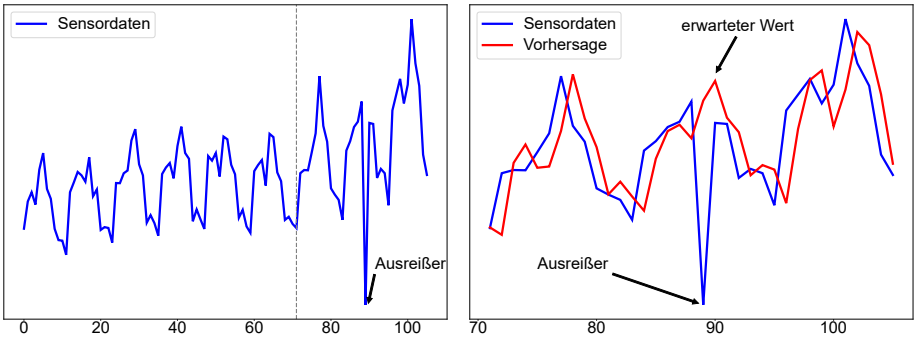
(d) Spline-Interpolation.

Abb. 2: Anwendungsbeispiel für die *Dateninterpolation* und die *Spline-Interpolation*.

4 Datenschutzkonzepte für Zeitreihendaten

Wie die Diskussion der verwandten Arbeiten zeigt, sind diese für die IoT-Domäne entweder zu restriktiv (attributbasierte Verfahren), geben unnötig viele Details weiter (Zugriffskontrollverfahren und musterbasierte Verfahren) oder lassen sich nicht für personenspezifische Anwendungsfälle nutzen (statistische Verfahren). Vielmehr sollte man für diesen Anwendungsbereich die charakteristischen Eigenschaften der Daten sowie der Verarbeitungsarten berücksichtigen. So besitzen bei Zeitreihendaten beispielsweise Ausreißer oft einen höheren Informationsgehalt und sollten daher besonders geschützt werden. Außerdem kommen bei der Verarbeitung dieser Daten häufig Kompressionstechniken zum Einsatz, um der schieren Datenmenge Herr zu werden. Wir stellen im Folgenden sechs Datenschutzkonzepte vor, die auf den Umgang mit Zeitreihendaten ausgelegt sind, d. h. deren Charakteristika ausnutzen.

Dateninterpolation: Datenpunkte, die stark von der Norm abweichen, besitzen oft einen wesentlich höheren Informationsgehalt, da sie auf ein ungewöhnliches Nutzerverhalten schließen lassen. Ein Beispiel hierfür ist in Abb. 2a gegeben. In der Kurve sei die Raumlautstärke über die Zeit aufgetragen. Wie man sieht, folgt diese einem zyklischen Muster (z. B. höhere Lautstärke am Tag als in der Nacht). In der Zeit zwischen $t_1 = 1.250$ und $t_2 = 1.570$ steigt der Lautstärkepegel jedoch unerwartet stark an. Aus diesem Werteverlauf könnte ein Angreifer beispielsweise ableiten, dass zu dieser Zeit mehr Personen als üblich in dem Raum anwesend waren. Möchte der Nutzer dies verbergen, könnte er die kompromittierenden Daten



(a) Unveränderte Rohdaten. (b) Vergleich von Ist-Werten und Vorhersage.

Abb. 3: Anwendungsbeispiel für die *Ausreißererkennung mittels Vorhersagen*.

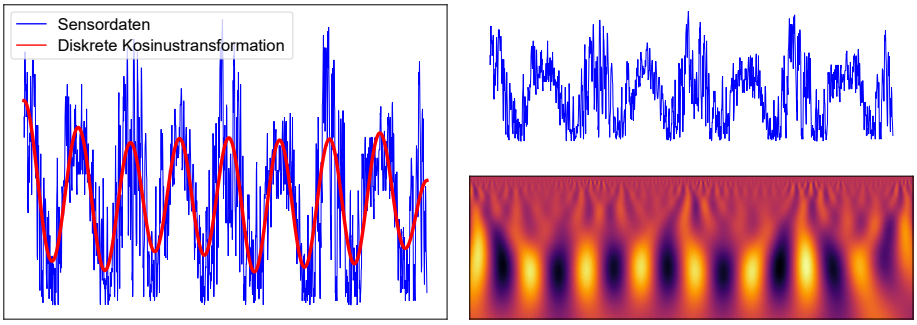
löschen. Dies würde allerdings zu Lücken im Werteverlauf führen und der Angreifer könnte erkennen, dass ihm Informationen vorenthalten wurden, und über den Grund spekulieren.

Ansätze wie *SmarPer* [Ol17] gestatten daher Sensordaten durch *Mock-Daten* (d. h. verfälschte Daten) zu ersetzen, beispielsweise wenn ein bestimmter Grenzwert überschritten wurde. Oft wird hierfür jedoch auf Nullwerte zurückgegriffen (siehe Abb. 2b). Für Zeitreihendaten ist dies allerdings aufgrund des atypischen, rapiden Wertabfalls ebenfalls sehr auffällig.

Wir schlagen daher vor, *lineare Interpolation* hierfür zu nutzen. Zu diesem Zweck reicht es aus, sich die beiden Datenpunkte vor und nach dem zu verschleiernden Bereich zu betrachten und damit die Gleichung $f(t) = y_1 * \frac{t_2-t}{t_2-t_1} + y_2 * \frac{t-t_1}{t_2-t_1}$ aufzulösen (siehe Abb. 2c). Dieses sehr einfache Verfahren führt allerdings zu kantigen Übergängen. Um noch glaubhaftere Mock-Daten zu erzeugen, kann auch auf *Spline-Interpolation* zurückgegriffen werden. Diese berücksichtigen auch die Steigungen am Anfang und Ende des zu füllenden Bereichs, was zu glatteren Übergängen führt (siehe Abb. 2d).

Ausreißererkennung mittels Vorhersagen: Während im obigen Beispiel davon ausgegangen wurde, dass Datenpunkte mit einem hohen Informationsgehalt einfach zu identifizieren sind (z. B. mittels eines Grenzwerts), kann sich dies im Allgemeinen als schwierig erweisen. Zu diesem Zweck kann eine Ausreißererkennung mittels Vorhersagen durchgeführt werden. Betrachtet man den Werteverlauf aus Abb. 3a, so kommt es zum Zeitpunkt $t_1 = 89$ zu einem Ausreißer, also einem Datenpunkt mit potentiell hohem Informationsgehalt. Dieser ist aber nicht offensichtlich, da es z. B. auch zum Zeitpunkt $t_2 = 101$ zu einem Extremum kommt.

In diesem Fall schlagen wir vor, mittels *maschinellen Lernens* eine Prognose für den weiteren Verlauf der Sensorwerte zu erstellen. Hierfür werden historische Daten des Nutzers als Trainingsdaten verwendet, um ein Modell zu lernen. Das heißt, die Werteverläufe der Vergangenheit werden analysiert und es wird eine *Hypothesenfunktion* $h(t)$ aufgestellt, die beschreibt, welcher Sensorwert zum Zeitpunkt t_{n+1} bei einem gegebenen $h(t_n)$ zu



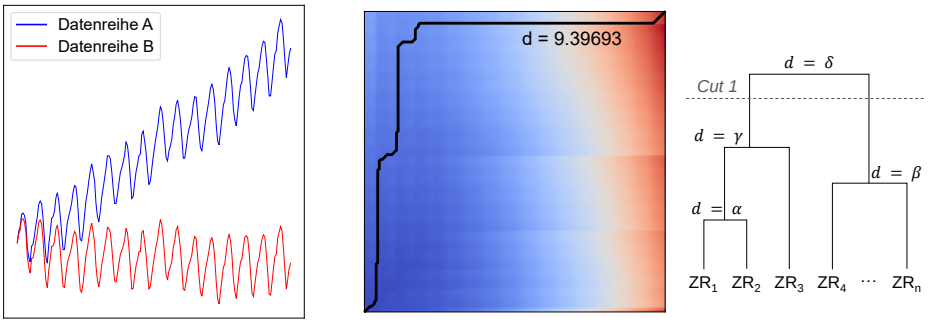
(a) Fourier-Transformation. (b) Wavelet-Transformation.

Abb. 4: Komprimierung von Zeitreihendaten durch *Fourier-* und *Wavelet-Transformationen*.

erwarten ist. Wir verwenden hierfür das *ARIMA-Modell*, das speziell für Zeitreihendaten ausgelegt ist, da es anstelle des vorherigen Sensorwerts das *gleitende Mittel* (Mittelwert der vorangegangenen n Werte) für die Prognose heranzieht. Die Aufteilung in Trainings- und Anwendungsdaten erfolgt zum Zeitpunkt $t_{split} = 71$, d. h. ab diesem Zeitpunkt erfolgt die Vorhersage. Abb. 3b stellt die Ist-Werte und die vorhergesagten Werte einander gegenüber. Weicht der Ist-Wert mehr als ein gegebenes Delta Δ_p von der Prognose ab, so handelt es sich nach unserer Definition um einen Ausreißer, d. h. um ein von der Norm abweichendes Verhalten, das nicht preisgegeben werden darf. Dieses Verfahren liefert glaubwürdige Mock-Daten von Haus aus mit, da im Fall von Ausreißern die prognostizierten Werte hierfür verwendet werden können. Der leichte Versatz der beiden Kurven ergibt sich durch das gleitende Mittel und kann durch Parameteranpassungen weiter verringert werden.

Signalglättung: Je nach Anwendung kann es aber auch erforderlich sein, genau diese Abweichungen von der Norm zu identifizieren. Im Rahmen von *Active Assisted Living* will man beispielsweise automatisch feststellen, ob ein Nutzer gestürzt oder anderweitig in eine Notsituation geraten ist. Hierzu muss gezielt nach ungewöhnlichen Mustern in den Daten gesucht werden, während die exakten Sensorwerte hingegen oft vernachlässigt werden können. Beispielsweise müssen die Daten keine exakten Bewegungsabläufe wiedergeben, wenn es für die Pflegekraft ausreicht, ungefähr zu wissen, wo der Sturz stattgefunden hat.

Zu diesem Zweck schlagen wir eine *Fourier-Transformation* vor. Hierbei werden zunächst die zeitdiskreten, äquidistanten Sensordaten von dem Zeitbereich auf ihren Frequenzbereich abgebildet, indem die Eingangssignalfolge als eine Summe von trigonometrischen Funktionen ausgedrückt wird. Wendet man diese Transformation auf aufeinanderfolgende Zeitfenster an, so erzielt man dabei einen *Bandfiltereffekt*, d. h. bei einer geeigneten Wahl des Zeitfensters lassen sich bestimmte Frequenzen abschwächen. Durch eine anschließende Anwendung der *inversen Fourier-Transformation*, erhält man eine geglättete Version der ursprünglichen Zeitreihe. In Abb. 4a ist dieser Effekt zu erkennen. Hierbei kommt die *diskrete Kosinustransformation* zum Einsatz. Die Sensordaten $x_0,...,x_{N-1}$ werden dabei auf die Frequenzen



(a) Anwendung des Dynamic-Time-Warping-Algorithmus auf Zeitreihendaten. (b) Clustering.

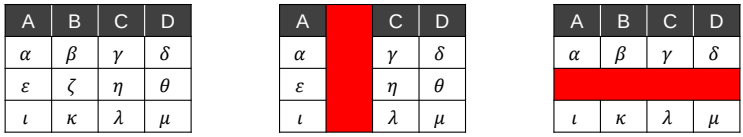
Abb. 5: Anwendungsbeispiel für eine *Clusteranalyse* von Zeitreihendaten.

X_0,...,X_{N-1} mittels folgender Funktion abgebildet: $X_k = \sum_{n=0}^{N-1} x_n * \cos\left[\frac{\pi}{N} * \left(n + \frac{1}{2}\right) * k\right]$. Die Werteverläufe sind weiterhin sichtbar, die Glättung eliminiert jedoch sämtliche Details.

Informationsexploration: Manche Anwendungen kommen mit noch weniger Daten aus. Eine Einbruchserkennung muss beispielsweise nur erkennen, dass ein ungewöhnliches Verhalten vorliegt. Daten über den Normalzustand werden hingegen überhaupt nicht benötigt.

In diesem Fall schlagen wir eine *Wavelet-Transformation* vor. Dabei handelt es sich um eine alternative Zeit-Frequenz-Transformation. Im Gegensatz zur Fourier-Transformation, die eine gleichbleibende Zeitfenstergröße für die Transformation nutzt, skaliert die Wavelet-Transformation das Zeitfenster. Auf diese Weise wird abhängig vom Eingangssignal eine bessere zeitliche Auflösung oder Frequenzauflösung erzielt, was insbesondere bei abrupten Frequenzwechseln zu besseren Ergebnissen führt. In Abb. 4b kommt das *Gaußsche Wellenpaket* als *Wavelet* zum Einsatz. In der grafischen Ergebnisrepräsentation darunter ist klar ersichtlich, dass dadurch Anomalien in den Daten betont werden (Maxima – hell und Minima – dunkel). Diese können im Anschluss der Anwendung gemeldet werden.

Clusteranalyse: Wurden die Anomalien entdeckt, so können diese gruppiert werden. Hierzu empfehlen wir ein *Cluster-basiertes* Verfahren. Zunächst werden die Zeitreihendaten in Abschnitte unterteilt (ein Abschnitt je Anomalie). Diese Abschnitte werden miteinander verglichen, indem paarweise ein Distanzmaß ermittelt wird. In Abb. 5a wird hierfür der *Dynamic-Time-Warping-Algorithmus* [RK04] verwendet. Bei diesem Algorithmus werden zwei unterschiedliche Signale (Datenreihe A und B) aufeinander abgebildet. Mittels unterschiedlicher Transformationen werden die einzelnen Datenpunkte des einen in die des anderen Signals überführt. Jede Transformation ist mit spezifischen Kosten verbunden. Diese Kosten werden in eine Matrix eingetragen. Der Algorithmus findet den *kürzesten* (i. S. v. kostengünstigsten) Weg von dem einen ins andere Signal (siehe Abb. 5a, rechte Seite). Die Kosten für diesen Weg stehen für die Distanz der beiden Signale. Für die Gruppierung stellt jeder Abschnitt stellt zunächst ein eigenes Cluster dar (siehe Abb. 5b). Schrittweise



(a) Ausgangsdaten. (b) Projektion. (c) Selektion.

Abb. 6: Auswirkungen einer *Projektion* und einer *Selektion* auf eine Datenbankanfrage.

werden anschließend die Cluster mit der kleinsten Distanz zusammengefasst, bis alle in einem einzigen Cluster liegen. Abhängig davon, wie viele Gruppen (d. h. Anomalie-Typen) man unterscheiden möchte, trennt man die Hierarchie auf (*agglomerative hierarchische Clusteranalyse*). In Abb. 5b werden zwei Gruppen von Anomalien gebildet (*Cut 1*). Diese können mit einem sprechenden Schlagwort versehen werden, das das zugrundeliegende Ereignis beschreibt (z. B. „Sturz“). Anschließend können die von Stach et al. [St18] vorgestellten Techniken angewandt werden, um deren zeitliche Abfolge zu verschleiern.

Query Rewriting: Selbst wenn die Daten bereits der Batch-Verarbeitung vorliegen und keine privaten Informationen entfernt wurden, können die Daten noch geschützt werden. Hierfür kann *Query Rewriting* genutzt werden, d. h. eine Anfrage wird automatisch vor der Ausführung umgeschrieben. List. 1 verdeutlicht dieses Vorgehen. Stellt der Nutzer beispielsweise die Anfrage a an eine Datenbank, so erhält er alle in der Tabelle „meine_sensoren“ gespeicherten Daten. Eine *Projektion* blendet bestimmte Spalten (d. h. Sensorattribute) aus. Anfrage b sorgt dafür, dass der Nutzer nur noch Temperaturdaten erhält. Eine *Selektion* blendet hingegen Spalten (d. h. Zeitbereiche) aus. Anfrage c gibt nur die Daten der letzten Woche zurück. Die Auswirkungen dieser Manipulationen sind in Abb. 6 veranschaulicht. Schließlich kann eine *Aggregation* die Daten zusätzlich verdichten. So gibt Anfrage d nur die durchschnittliche Temperatur je Raum zurück.

Diskussion der Datenschutzkonzepte: Die sechs vorgestellten Datenschutzkonzepte für Zeitreihendaten operieren auf unterschiedlichen Abstraktionsleveln. Je nach Anwendungsfall kann eine andere Strategie zum Einsatz kommen. Dadurch kann gewährleistet werden, dass private Informationen geschützt werden und dennoch die Servicequalität

```
1 SELECT *
2 FROM "meine_sensoren"
```

(a) Anfrage über alle Daten.

```
1 SELECT "temperatur"
2 FROM "meine_sensoren"
```

(b) Anwendung einer Projektion.

```
1 SELECT *
2 FROM "meine_sensoren"
3 WHERE time > now() - 7d
```

(c) Anwendung einer Selektion.

```
1 SELECT MEAN("temperatur")
2 FROM "meine_sensoren"
3 GROUP BY "raum"
```

(d) Anwendung einer Aggregation.

List. 1: Anwendungsbeispiele für *Query Rewriting*.

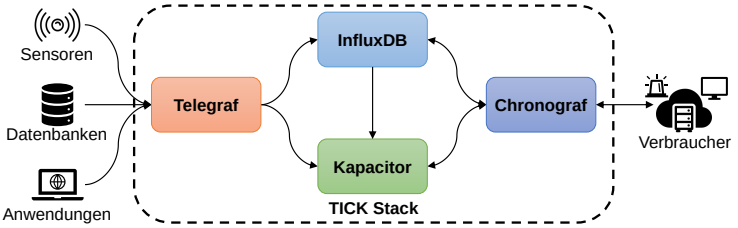


Abb. 7: Der *TICK-Stack* als Implementierung der *Lambda-Architektur*.

aufrechterhalten bleibt, d. h. die IoT-Anwendung wird weiterhin mit ausreichend Daten versorgt, um ihre Analysen durchzuführen. Mittels **Dateninterpolation** können konkrete Werte(bereiche) mit einem hohen Informationsgehalt verborgen werden. Abhängig davon, wie wichtig es ist, dass einem Angreifer diese Manipulation nicht auffällt, kann hierfür beispielsweise eine einfache lineare Interpolation, Splines oder maschinelles Lernen genutzt werden. Die **Vorhersagen**, die letzteres ermöglicht, helfen auch bei der Identifikation von potentiell schützenswerten Datenpunkten. Sollen nicht nur einzelne Datenpunkte verborgen werden, sondern eine Unschärfe auf das gesamte Sensorsignal angewandt werden, eignen sich Fourier-Transformationen zur **Signalglättung**. Die Wavelet-Transformationen führt diese Filterung noch weiter, indem sie Anomalien in den Daten pointiert. Dies dient der **Informationsexploration**. Auf diesen Anomalien kann eine **Clusteranalyse** durchgeführt werden, um die Daten noch weiter zu verdichten. Auch Kombinationen dieser Strategien sind möglich. So können beispielsweise alle Anomalien identifiziert werden und ausgewählte mittels Dateninterpolation verborgen werden. Während diese Strategien proaktiv sind, schützt **Query Rewriting** die Daten auch, wenn sie bereits an die Batch-Verarbeitung weitergeleitet wurden. Dieses Verfahren ist sehr mächtig und kann beliebige Restriktionen auf Anfragen anwenden. Allerdings steigt die Komplexität dieses Verfahrens je nach Restriktion stark an.

Im Nachfolgenden stellen wir Implementierungsansätze für diese Schutzkonzepte vor und diskutieren anhand des TICK-Stacks, wie dadurch *Privacy by Design* ermöglicht wird.

5 Implementierungsansätze

Der TICK-Stack ist eine Implementierung der Lambda-Architektur (siehe Abb. 7). Er besteht aus vier Komponenten: Der *Telegraf* (Verteilerkomponente) sammelt Daten von beliebigen Datenquellen und stellt diese der *InfluxDB* (Batch-Verarbeitung) und dem *Kapacitor* (Datenstromverarbeitung) zur Verfügung. Die InfluxDB ist eine dedizierte Datenbank für Zeitreihendaten während der Kapacitor eine Verarbeitungs-Engine für Zeitreihendaten ist. Neben den Daten von Telegraf hat er auch Zugriff auf die InfluxDB. Der *Chronograf* kann mit diesen beiden Komponenten interagieren und die dort berechneten Analyseergebnisse für Verbraucher aufbereiten. Er stellt gleichzeitig die Schnittstelle zum Nutzer dar, der darüber Anfragen an die InfluxDB und den Kapacitor stellen kann. Somit besitzt der TICK-Stack alle Funktionalitäten, die eine IoT-Anwendung benötigt (siehe Abschnitt 2).

Wir haben die in Abschnitt 4 vorgestellten Konzepte als *Python-Skripte* umgesetzt, die selbst auf einem *Raspberry Pi* effizient ausgeführt werden können. Somit könnten sie beispielsweise in der *Edge*, also an einem vom Nutzer kontrollierten Punkt, die Daten vorfiltern, bevor diese zur Verarbeitung weitergereicht werden. Da die einzelnen Komponenten des TICK-Stacks verteilt ausgeführt werden können, ist es daher möglich den Telegraf ebenfalls in der Edge zu betreiben und darin unsere Skripte zu integrieren. Somit ist sichergestellt, dass die InfluxDB und der Kapacitor keine privaten Daten mehr erhalten. Da diese beiden Komponenten aber selbst ebenfalls Python-Schnittstellen besitzen, können unsere Skripte auch anbieterseitig angewandt werden, um so Privacy by Design zu realisieren. Ein Beispiel hierfür wäre eine rollenbasierte Zugriffsschicht, die dem Chronograf vorgeschaltet ist. Je nach IoT-Anwendung wählt diese Zugriffsschicht die entsprechenden Datenschutzkonzepte aus (entsprechend der Rolle der Anwendung), um dadurch eine DSGVO-konforme Verarbeitung zu gewährleisten.

6 Zusammenfassung

Die massiven Fortschritte, die IoT-fähige Geräte in den letzten Jahren bezüglich Rechenleistung, Übertragungsgeschwindigkeit sowie Sensorik erfahren haben, haben die technischen Voraussetzungen für eine Vielzahl an IoT-Anwendungen geschaffen. Sie durchdringen sämtliche Bereiche des täglichen Lebens, beispielsweise Smart Homes, den Gesundheitssektor oder die Industrie 4.0. Um die Vorteile dieser Anwendungen genießen zu können, müssen Nutzer allerdings viele und zum Teil hochgradig private Daten preisgeben. Heutige Datenschutzlösungen sind allerdings nicht auf die speziellen Eigenschaften der hierbei zum Einsatz kommenden Zeitreihendaten angepasst, wodurch sie unnötig restriktiv sind.

In diesem Artikel stellen wir daher Datenschutzkonzepte speziell für Zeitreihendaten vor. Diese Konzepte berücksichtigen neben der Struktur der Daten auch deren spätere Verarbeitungsarten. So ist die *Dateninterpolation* und die *Ausreißererkennung* vergleichbar mit der Rauschunterdrückung, die auf Sensordaten häufig angewandt wird, während die *Signalglättung* und *Informationsexploration* respektive *Clusteranalyse* oft zur Komprimierung von Sensordaten genutzt werden. Mithilfe von *Query Rewriting* kann der Zugriff auf die Daten weiter eingeschränkt werden. Auf diese Weise lassen sich private Daten zurückhalten und damit gezielt private Informationen vor IoT-Anwendungen verbergen. Gleichzeitig wird die Servicequalität beinahe beibehalten, da viele der Operationen ohnehin zu einem späteren Zeitpunkt angewandt worden wären. Diese Konzepte haben wir als leichtgewichtige Python--Skripte umgesetzt, die sich beispielsweise in den TICK-Stack integrieren lassen. Damit kann die Forderung der DSGVO nach einer handhabbaren Privacy-by-Design-Datenschutzlösung erfüllt werden, ohne die Servicequalität von IoT-Anwendungen unnötig einzuschränken. In zukünftigen Arbeiten gilt es nun zu prüfen, wie sich die Konfiguration der Skripte (d. h. die Spezifikation der Privacy-Richtlinien) nutzerspezifisch automatisieren lässt (z. B. mithilfe von maschinellem Lernen), um so die Belastung für den Nutzer zu reduzieren.

Danksagung: Die in diesem Beitrag vorgestellte Forschungsarbeit entstand aus dem PATRON-Forschungsauftrag, der von der Baden-Württemberg Stiftung finanziert wurde.

Literatur

[Ad11] Adaikkalavan, R. et al.: Multilevel Secure Data Stream Processing. In: DBSec '11. 2011.

[Bh19] Bhattacharjya, A. et al.: Security Challenges and Concerns of Internet of Things (IoT). In: Cyber-Physical Systems. Springer, Cham, Kap. 7, S. 153–185, 2019.

[Bi15] Birman, K. et al.: Building a Secure and Privacy-Preserving Smart Grid. SIGOPS Oper. Syst. Rev. 49/1, S. 131–136, 2015.

[Bo18] Bourgeois, J. et al.: Trusted and GDPR-compliant Research with the Internet of Things. In: IOT '18. 2018.

[EB09] Eckert, M.; Bry, F.: Complex Event Processing (CEP). Informatik-Spektrum 32/2, S. 163–167, 2009.

[Gr19] Gritti, C. et al.: Privacy-Preserving Delegable Authentication in the Internet of Things. In: SAC '19. 2019.

[Ma17] Marjani, M. et al.: Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges. IEEE Access 5/1, S. 5247–5261, 2017.

[Ma19] Marikyan, D. et al.: A systematic review of the smart home literature: A user perspective. Technol. Forecasting Social Change 138/1, S. 139–154, 2019.

[MW15] Marz, N.; Warren, J.: Big Data: Principles and best practices of scalable real-time data systems. Manning Publications Co., Shelter Island, NY, 2015.

[Ol17] Olejnik, K. et al.: SmarPer: Context-Aware and Automatic Runtime-Permissions for Mobile Devices. In: S&P '17. 2017.

[Ou16] Ouaddah, A. et al.: Access control in IoT: Survey & state of the art. In: ICMCS '16. 2016.

[PO12] Pieterse, H.; Olivier, M.: Data Hiding Techniques for Database Environments. In: DigitalForensics '12. 2012.

[RK04] Ratanamahatana, C. A.; Keogh, E.: Everything you know about Dynamic Time Warping is Wrong. In: TDM '04. 2004.

[St18] Stach, C. et al.: How a Pattern-based Privacy System Contributes to Improve Context Recognition. In: PERCOM WKSHPS '18. 2018.

[Ve17] Velosa, A. et al.: Hype Cycle for the Internet of Things, 2018, Report G00340237, Gartner, 17. Juli 2017.

[Vo17] Voss, W. G.: First the GDPR, Now the Proposed ePrivacy Regulation. Journal of Internet Law 21/1, S. 3–11, 2017.

[Wa18] Wachter, S.: Normative challenges of identification in the Internet of Things. Computer Law & Security Review 34/3, S. 436–449, 2018.

[Zh18] Zheng, S. et al.: User Perceptions of Smart Home IoT Privacy. Proc. ACM Hum.-Comput. Interact. 2/CSCW, 200:1–200:20, 2018.