A Comprehensive Approach to Privacy in the Cloud-based Internet of Things[☆]

Martin Henze[a,*], Lars Hermerschmidt[c], Daniel Kerpen[b], Roger Häußling[b], Bernhard Rumpe[c], Klaus Wehrle[a]

[a]*Communication and Distributed Systems, RWTH Aachen University, Germany*
[b]*Sociology of Technology and Organization, RWTH Aachen University, Germany*
[c]*Software Engineering, RWTH Aachen University, Germany*

Abstract

In the near future, the Internet of Things is expected to penetrate all aspects of the physical world, including homes and urban spaces. In order to handle the massive amount of data that becomes collectible and to offer services on top of this data, the most convincing solution is the federation of the Internet of Things and cloud computing. Yet, the wide adoption of this promising vision, especially for application areas such as pervasive health care, assisted living, and smart cities, is hindered by severe privacy concerns of the individual users. Hence, user acceptance is a critical factor to turn this vision into reality.

To address this critical factor and thus realize the cloud-based Internet of Things for a variety of different application areas, we present our comprehensive approach to privacy in this envisioned setting. We allow an individual user to enforce all her privacy requirements before any sensitive data is uploaded to the cloud, enable developers of cloud services to integrate privacy functionality already into the development process of cloud services, and offer users a transparent and adaptable interface for configuring their privacy requirements.

Keywords: Privacy, Cloud Computing, Internet of Things, Model-driven Development, User Acceptance

1. Introduction

The proliferation of the Internet of Things (IoT), which enables the world wide interconnection of an incredible large amount of smart things, allows to effectively realize systems that significantly improve everyday's life, ranging from

[☆]This is a significantly extended and rewritten paper based on the work *User-driven Privacy Enforcement for Cloud-based Services in the Internet of Things* by Henze et al. presented at the 2014 International Conference on Future Internet of Things and Cloud.
[*]Corresponding author. Tel.: +49-241-80-21425
Postal address: Informatik 4 (COMSYS), Ahornstr. 55, 52074 Aachen, Germany
Email address: `henze@comsys.rwth-aachen.de` (Martin Henze)

pervasive health care and assisted living to smart cities [1, 2]. However, these smart devices often suffer from extremely constrained processing and storage resources, and a limited energy budget, as they are often powered by battery. In order to overcome these limitations, one of the most promising approaches is to interconnect the IoT with the cloud and thus benefit from the elastically scalable and always available resources provided by the cloud computing paradigm [3–9]. The cloud-based Internet of Things simplifies storage and processing of collected data, allows using the same data in multiple services, eases the combination of data from several users, and supports user mobility. At the same time it prevents information fragmentation over several databases.

Although the necessary technologies for both, IoT and cloud computing, are readily available today, the interconnection of these two paradigms in application areas such as assisted living and public mobility assistance is gravely hindered by severe privacy concerns of individual users [2, 10–12]. In order to develop such a system that is accepted and hence employed by a wide range of users, it is of up-most concern to ensure users' control over their data. Notably, an individual user might still want to rate, e.g., in the context of pervasive health care, health higher than privacy in case of an emergency. For this, it is crucial to anchor privacy aspects of *individual* users within the system.

However, cloud services are typically not developed solely for one specific user, but instead target a large group of heterogeneous customers [13]. Hence, it is infeasible to decide on all privacy aspects already during the development of a cloud service as privacy by design might suggest. Instead, in order for users to accept such a service, privacy choices must be left to the user. This, however, significantly increases the complexity of designing and developing cloud services and at the same time puts a tremendous burden on the user who often has no awareness of the (technical) consequences of her privacy choices. Therefore, we strive to encompass both domains, end-user's as well as service provider's perspective with our comprehensive understanding of cloud-based IoT services.

In this paper, we present UPECSI, our solution for **U**ser-driven **P**rivacy **E**nforcement for **C**loud-based **S**ervices in the **I**oT. UPECSI takes a comprehensive approach to privacy for the cloud-based IoT by providing an integrated solution for privacy enforcements that focuses on individual end-users and developers of cloud services at the same time. UPECSI consists of several technical components and organizational processes. More specifically, with UPECSI, we present the following core contributions: i) individual, user-driven enforcement of privacy requirements already before any potentially sensitive data is handed over to the cloud, ii) a novel technique for designing and implementing cloud-based services that integrates privacy functionality into the development process, and iii) an easy to understand, flexible, and transparent approach for users of different privacy expertise to configure their individual privacy settings. These contributions of our comprehensive approach to privacy in the cloud-based IoT allow us to lay the foundation for bringing the IoT and cloud computing together in a user-accepted fashion.

This paper is structured as follows: In Section 2 we describe and discuss the application areas and network setting of our envisioned scenario. Based on this, we derive and present privacy concerns of end-users and privacy considerations of service providers that arise in a scenario like this in Section 3. We present important related work in Section 4. In Section 5, we formalize the challenges and requirements for realizing privacy-preserving cloud-based services for the IoT. To address these challenges and requirements, we present the main contribution of this paper, the design and implementation of UPECSI, in Section 6. We discuss in detail how this addresses and overcomes the identified challenges in Section 7. Finally, we conclude this paper in Section 8.

2. Scenario

The following section outlines our envisioned scenario. We address both the societal and, especially, technical point of view by discussing exemplary application areas from the contexts of assisted living and interactive mobility assistance in public spaces. Driven by these application areas, we derive and present an overview of the underlying network scenario of our system.

2.1. Application Areas

Our two subsequently presented application areas focus around a 75-years-old widowed female retiree, who, besides tending her close relationships to family members and friends, appreciates having a wide range of options to live independently. In the first application area, *assisted living*, we consider this lady controlling her room/building automation related systems, e.g., screening mechanical, lighting, heating/ventilating/air conditioning (HVAC), and security systems along with monitoring her vital signs by a number of unobtrusive sensors in her apartment. These sensors deliver information to the cloud offering fast access to, for instance, family members and third parties such as medical personnel (e.g., doctors), health care providers (HCPs), or technical staff (e.g., building service engineers).

In the second application area, *public mobility assistance*, we envision this lady being comfortable with seamlessly taking an assisted living service with her on her portable and/or wearable devices. By doing so, this service takes a proactive role as a "personalized companion" by providing seamless mobility chains for aged users. Such a service enables people with limited mobility traveling independently by combining technical assistance systems (e.g., via smartphones and devices such as Google Glass) and public transportation: The entire route is covered from the starting point to the destination, including not only the public transport services (timetables of buses, trams, taxi services etc.) but adding additional service value by including information on the local and regional conditions of the route because existing barriers (stairs, etc.) might be recorded, classified, and cataloged with their position data. Whereas today such services are already in development which aim primarily at assisting independent living of the elderly in their preferred environments [14, 15], we deem it necessary to add additional benefit to such concepts by i) integrating privacy enforcement (e.g., concerning mobility patterns, fingerprints etc.), ii) offering more functionality in terms of adaptable and transparent configurations (to the end-user as well as further stakeholders, e.g., including end-user's relatives and other affiliated persons, as well as other trusted third party actors), and, finally, iii) generating higher user acceptance.

Our envisioned scenario is influenced by social forces such as aging (western) societies and the therefore increasing role of technology in supporting formal and informal care/assistance. Focussing on older life as well as current and, especially, future needs for health and social care, we have to consider the changing social structure of older life [16]. Departing from general demographic developments (e.g., increasing share of people aged 65 years or above in the total population and longer span of "older" life phases due to rising life expectancy), aging societies have to face inextricably linked pressing forces. On the one hand, they have to cope with economic frictions (expectations about the quality of health care systems stemming from certain standards of living vs. financial constraints and shortage of skilled labor in the public health services sector) as well as eminent societal processes on the other hand. In this context, some of the most important societal changes to be highlighted might be singularization (increasing number of the elderly un-/deliberately living alone with their families/relatives distributed over large geographical areas due to increased mobility in society), differentiation (e.g., not one common understanding but many individualistic approaches of how health and social care should be provided), and changing gender roles (due to "feminization of aging" indicating at the increasing number of women in old and, especially, very old age groups).

Hence, our overall scenario with its application areas serves as adequate example to derive design requirements for our system to be user-accepted and provides us with a promising evaluation setting by keeping in mind the differentiation and variability of old/very old age, i.e., including different age cohorts with a plurality of norms, consumer/consumation habits, and technology experiences/knowledge. However, our technical foundations are applicable to much broader contexts ranging from building management systems over intelligent transportation systems to smart manufacturing.

2.2. Network Scenario

Based on our application areas, we derived the following network scenario for realizing cloud-based services for the IoT which is depicted in Figure 1. As we focus on the privacy aspects of the integration of the IoT with cloud computing, our network scenario follows a user-centric view. Each *user (U)* of our system owns and thus operates one or more *IoT devices (D)*, often also referred to as smart objects. Following the definition of Gubbi et al. [17], we consider IoT devices that sense information from the environment, interact with the physical world, and – most importantly – allow communication using traditional Internet standards. Typical examples for IoT devices in our application areas range from sensitive floors for movement monitoring and fall detection in home environments and smart textiles (e.g., shirts, wristbands, or shoes) monitoring various vital parameters and connected to emergency notification systems to

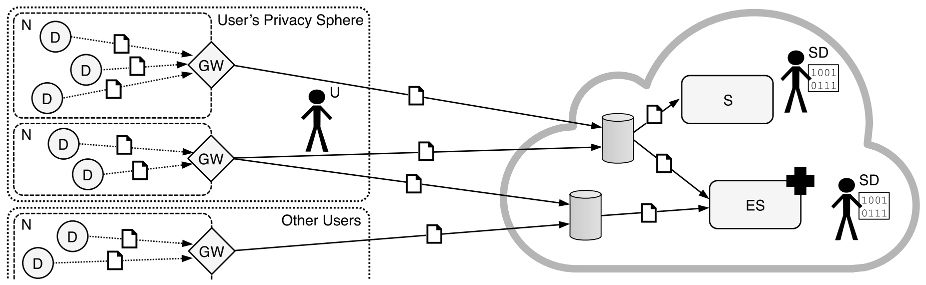


Figure 1: An individual user (U) operates one or more IoT networks (N). The IoT devices (D) in these networks send data to the cloud via a gateway (GW). The cloud stores data in different databases and provides it to services (S), as well as emergency services (ES), which are provided by service developers (SD) and were authorized to access data by the user.

advanced devices capable of monitoring specialized implants, such as artificial cardiac pacemakers (ACP).

These IoT devices typically build upon embedded platforms to reduce production costs and hence facilitate deployment. As such, they often suffer from limited storage and processing resources. Especially in mobile settings, they furthermore have to cope with limited connectivity and – as they are often battery driven – a limited energy budget. These limitations are addressed by the cloud-based IoT, where the IoT is interconnected with the cloud [3–9]. Given that, the core idea is to upload all sensed data to the cloud, where it is stored persistently. The user can then authorize specific cloud services to access and operate on her data and thus realize the wanted range of functions. Notably, all functionality that operates on IoT data is realized in the cloud, mainly because the design of the cloud-based IoT aims to maximize availability of data and services. However, in the case of a (temporary) connection disruption between an IoT device and the cloud, data cannot be pushed to the cloud immediately. To overcome this issue, the IoT device can cache the data locally and upload it to the cloud once the connection has been reestablished. Still, the cloud-based IoT allows to access all data that is already stored in the cloud and operate services on it even during a disrupted connection between IoT device and cloud. Hence, the cloud-based IoT significantly increases availability when compared to solutions that propose to store and process IoT data locally (and hence become completely unavailable in case of connection problems).

When realizing the cloud-based IoT, the IoT devices of one user are typically grouped into one or multiple logically or even physically separated IoT networks (N), e.g., a home network consisting of assisted living devices and a body area network connecting unobtrusive health care devices. All these networks are connected to the Internet and hence the cloud using a dedicated *gateway (GW)*. In the assisted living context, this would typically be a home router while for public mobility assistance the user's smartphone could act as the gateway. We define the combined network of all IoT devices and networks of a single user as her *privacy sphere*. The user trusts the devices and other network participants within this sphere but does not want any potentially sensitive information collected within this context to be available to unauthorized third parties. For this, the user has to employ standard network security measures for home and IoT networks, e.g., wireless channel encryption.

Data that is sensed in the networks of a user is forwarded to the *cloud* via the dedicated gateway. The cloud stores the data persistently and makes it accessible to *services (S)* and *emergency services (ES)*. These services are created by *service developers (SD)* who implement the functionality of a service. Additionally, they may implement supporting functionality, e.g., billing or elastically scaling with increasing load. These services are then operated by *service providers*. We assume that a large number of users, service developers, and service providers participate in this setting, which leads to a multi-tenancy scenario with all its privacy implications.

To be able to securely identify the individual technical entities in this network scenario, we assume that each technical entity (i.e., gateway, cloud, service, emergency service) is equipped with a public key certificate that certifies this entities' identity. This requires the existence of a public-key infrastructure with one or more trusted certificate authorities. These could, e.g., be operated by a trusted third party such as a non-profit organization or government agency.

The described network scenario understandably raises severe privacy concerns of users, as potentially sensitive data is outsourced to the cloud. It additionally requires privacy considerations by service providers, e.g., in order to meet data protection standards demanded by law. We will analyze and discuss these in more detail in the following section.

3. Privacy Concerns and Considerations

When realizing and implementing a scenario as described above, in which potentially sensitive data collected by IoT devices is outsourced to the cloud, different privacy aspects have to be considered. In the following, we present and discuss these aspects in more detail. For this, we further divide them into privacy concerns of end-users and privacy considerations of service providers.

3.1. Privacy Concerns of End-users

The data collected by IoT devices often consists of sensitive information that unauthorized third parties might be interested in [2]. For example, as one of our application areas is (public) mobility assistance, information collected by a car-based telematics system might be extremely valuable for insurance companies, as this knowledge could be used to increase a person's fee or even deny a new contract [18]. Additionally, not only the sensed data itself but also corresponding meta information might be considered sensitive [19], especially when thinking of location privacy [20], where location fixes and/or time stamps collected by GPS, wireless networks, or NFC tags are considered important meta information. Hence, users often do not want to reveal the data collected by their IoT devices to any third parties.

Even more privacy concerns and issues arise when outsourcing this data to the cloud. The major concern of end-users in this setting is the perceived loss of control over data when it is outsourced to the cloud [11, 12]. Hence, providers of cloud-based services must explicitly commit themselves to guarantee confidentiality and protection of the stored and processed data, because otherwise, an adoption barrier consequently is likely to appear for the data owner due to her individual concerns. Such individual concerns mainly result from the fact that there is no control or at least transparency over the access to this data and, hence, data might be handed over to third parties or misused for unintended purposes [12]. Due to these concerns, end-users ultimately tend to refrain from using cloud-based services for (highly) sensitive data such as health-related information, for instance stored and shared by cloud-based personal health records (PHRs) systems [21].

On a more general scale, people surveyed tend to declare their expectations about reasonable protection of their data, that legislation is adhered to, and that they should be informed about when and for what purpose their data is used [22]. This is one emphasize on the importance of including the user within privacy research, especially by providing transparency. However, almost consistently and further complicating implementation concepts of cloud-based systems, such interviewees rashly declare to handle their data on the Internet as sparingly as possible and just to rarely give away sensitive data such as bank account information or credit card numbers. In this context, a discrepancy between expressed attitudes and factual behavior might be observed similar to the one that accords to surveys concerning pro-environmental behavior: albeit people surveyed are clear about the fact that giving away sensitive data is always critical, the risk is quite often taken if there is at least a small but immediate advantage such as, e.g., faster order procession or a discount in the case of online shopping [23, 24]. Finally, such observations of empirical discrepancies between general cloud-related user attitudes and behavior, as well as users' willingness to trade-in privacy in health-related contexts, e.g., in case of emergencies or quality of life in a nursing home [25] clearly stresses awareness about privacy as not being a fixed norm. This furthermore emphasizes the importance of including sociological sound concepts of end-users within privacy research.

3.2. Privacy Considerations of Service Providers

Notably, considering end-user's privacy concerns is not only important for researchers, but at least equally relevant for service providers. If service providers do not address the above depicted concerns adequately, they have to expect severe undesired consequences, ranging from the complete nonacceptance of their service to extremely costly lawsuits [2, 26]. Additional to these lawsuits, which mainly target private law issues between individual end-users and service providers, service providers also have to follow several legal restrictions. This becomes especially important when the realization of service functionality is outsourced to the cloud [12, 26]. One legal restriction that is considered extremely challenging with respect to the cloud and that also raises severe concerns of end-users, is the transfer, storage, and processing of customer data across legislative boundaries (which often happens completely unnoticeable) [27–30]. For example, in many countries of the European Union it is forbidden by federal law to store tax and accounting records as well as corporate documents outside the respective jurisdiction of these countries [31]. Another legal restriction is a recently proposed new data protection regulation in the European Union known as "the right to be forgotten" [32]. This right states, in principal, that information about an individual person has to automatically be deleted after a certain period of time. Implementing the right to be forgotten in a technical system, especially in the context of cloud computing, is considered as an extreme challenging task [12, 32]. Especially for adhering to legal requirements, service providers depend on support of the cloud providers they make use of to realize their services [12, 33]. Hence, service providers do not only have to care about the privacy requirements of their customers but additionally become customers with privacy requirements themselves as soon as they utilize the cloud for their services.

As we already discussed earlier when considering end-users' privacy concerns, the service provider has to lay out all usage of customer data in order to satisfy the user demand for transparency. Whereas a service provider pursuing the development and marketing of an application usually decides at design time on the information processed by the application, there is no such solution capturing users privacy requirements at design time. Descriptions of such applications' privacy policies are developed by specialized lawyers or companies' legal departments, with such personnel having to interview the developers in order to get information about the processed data. In this regard, changing derived policies is not a common, or at least willingly accepted practice because such changes have to be signed off by legal experts and communicated to users again. Furthermore, as human communication is error prone [34], derived policy descriptions from developer interviews are often formulated very defensive, so that potential lawsuits filed by users might not be successful even if the data handling deviates from the developer-intended and described behavior [35]. This lack of precise, up-to-date information about the data processing again leads to uncertainty and non-transparency for the users.

Concluding our consideration of end-users' and service providers' privacy considerations, we sum up that the development of a service adaptable to users' individual needs is more complex then developing a service for a consistent group of predefined users with fixed privacy requirements. This even more stresses the importance of our approach which supports developers in building software products adjustable to users' privacy needs in order to bring the advantages of privacy by design to software products and the software development process.

4. Related Work

In order to address individual parts of the above presented privacy concerns of end-users and privacy considerations of service providers several approaches have been proposed by the research community. We group our description and discussion of this related work into four categories. First, we discuss approaches for securely outsourcing data to the cloud. Based on this, we present solutions for preserving privacy in the IoT and cloud computing. Then, we give an overview over the field of privacy policy languages. Finally, we introduce the model-driven development approach and its applications.

4.1. Securely Outsourcing Data to the Cloud

In recent years, the field of securely outsourcing data to the cloud, especially data collected by large numbers of comparably small devices, has spawn significant interest in the research community.

A large number of approaches aim at securing health data when it is outsourced to the cloud. One of the first approaches in this direction was proposed by Lounis et al. [36]. In their work, they particularly focus on guaranteeing confidentiality and integrity of outsourced medical data with minimum management and processing overheads. Thilakanathan et al. [37] proposed a platform that realizes mobile telecare by allowing doctors to remotely monitor patients. For this, they rely on the cloud as a central data storage which requires them to take special care of security, confidentiality, and access revocation. In a similar context, Li et al. [38] and Liu et al. [21] proposed approaches for realizing the scalable and secure sharing of personal health records (PHRs) using the cloud. In order to secure the health records in this setting, they make use of attribute-based encryption respectively signcryption.

On a more general scale, other researchers focus on securely outsourcing general-purpose sensor data to the cloud. The work of the SensorCloud project [3, 5, 7] aims at protecting the sensor data already within the sensor network, i.e., before it is uploaded to the cloud. The operator of the sensor network may then select specific cloud services and allow them fine-grained access to individual parts of her sensor data. As this requires the cloud services to realize and implement the necessary security functionalities to verify and decrypt sensor data, the sensor cloud security library has been proposed as an abstraction of these security functionalities [39]. Similarly to the approach taken in the SensorCloud project, Pooja et al. [40] also realize the protection of sensor data already within the sensor network. In order to further increase the security of outsourced data, they make use of two separate clouds for storing the encrypted sensor data respectively the keying material needed for decryption.

All these approaches consider security aspects when outsourcing data to the cloud. However, they mainly focus on providing confidentiality and access control. Hence, additional work is required in order to fully meet the privacy requirements identified in Section 3 when outsourcing data to the cloud.

4.2. Preserving Privacy in the IoT and Cloud Computing

Solutions for preserving privacy have been proposed in both domains, IoT and cloud computing, and mainly focus on individual application scenarios. In the context of the privacy-preserving utilization of medical data, e.g., by clinical researchers, Molina et al. [41] present a solution for the privacy-preserving computation of statistics on health care data in the context of the IoT. Similarly, Yang et al. [42] address the challenge of privacy-preserving medical data sharing in the context of cloud computing. In the context of smart home automation systems, risk driven approaches have been proposed to preserve privacy when utilizing the IoT [43] and the cloud [44]. In order to preserve privacy when outsourcing data to the cloud, a variety of solutions have been proposed. These range from annotating data with fine-grained privacy obligations that impose restrictions on where and for how long data is allowed to be stored [12] over utilizing information flow control for enforcing location requirements [45] to the utilization of tamper-proof hardware components in order to realize the privacy-aware storage and processing of data in the cloud [46].

The referenced approaches present very promising ideas for preserving privacy in the IoT and cloud computing. However, they mostly focus on specific application scenarios and selected aspects of privacy. To the best of our knowledge, there is no comprehensive effort for providing privacy in the combination of the IoT and cloud computing that supports users and service developers in using and designing services in a privacy-aware way by taking the various aspects of privacy as discussed in Section 3 into account.

4.3. Privacy Policy Languages

To express users' privacy requirements and privacy policies in a machine-readable way, several approaches have been proposed in the context of access control systems. Karjoth et al. [47] developed a privacy policy model as a basis for an access control system which respects user consent and obligations. P-RBAC by He, Ni et al. [48, 49] is an extension of the Role Based Access Control model with privacy policies based on data handling purposes and obligations. For expressing users' data distribution policies, Spillner et al. [50] proposed the textual language FlexDDPL.

Notably, the SPARCLE Workbench [51] transforms privacy policies from natural language to XACML [52], a standard for expressing access control policies, or P-RBAC and back. This is presented as useful for policy makers, auditors, and managers [53] but not for service developers, as they need a compact and comprehensive language, which adopts known concepts of their domain [54]. Trabelsi et al. [55] developed the PrimeLife Privacy Policy Language (PPL), which extends XACML to express users' and service providers' privacy policies. They also implemented a user agent, which compares these policies and in case of a mismatch, asks the user if she would like to adjust her policy. All these approaches are well suited to express privacy policies, but they are not integrated within the service development process.

From a software development perspective Colombo and Ferrari [56] utilized UML to develop the Privacy-aware Modeling Language (PaML), which allows modeling and analysis of systems with purpose full data usage control. For use in the early development phase Jutla et al. [57] extended UML Use Case Diagrams with privacy measures or services like `Notice and Agreement`, `Pseudunymization` and `Anonymization`. These approaches assume users' participation during the design process to enable choice, which is true for business applications, but not for our end-users centered scenario.

4.4. Model-driven Development

The model-driven development (MDD) approach [58] proposes to use models instead of general purpose programming language code, which are then used to generate parts of the software [59]. This allows to increase the productivity of the software development process by rising the layer of abstraction during programming and hence concentrate on concepts rather then technical details. A modeling language used in MDD and beyond that during software design is the Unified Modeling Language (UML) [60]. As UML is a graphical language its use as a language during development is limited. To overcome these shortcomings the UML/P [61, 62], an adoption of the UML for programmers has been developed.

To realize the MDD approach tools which work on models to process, analyze, and transform them are needed. MontiCore [63] and Xtext [64] are frameworks which support building such tools for textual modeling languages. On-top of MontiCore, tools for processing UML/P [65] and generating program code have been developed. We build upon these tools to realize the Privacy Development Language which we present in Section 6.1.1.

To use modeling techniques in specific problem domains, several domain specific languages (DSL) have been introduced [66]. Related to this work, extensive work on security modeling has been done [67] which primarily focuses on role based access control [68, 69]. Although access control is an impotent part of implementing privacy policies, these modeling languages do not support to fully model users' privacy.

5. Problem Statement

Motivated by the need and high demand for integrating the IoT and cloud computing [9], especially in application areas such as assisted living and public mobility assistance (see Section 2), we strive at realizing a tight interweaving between the two technical paradigms. Notably, besides an impressive amount of technical challenges in doing so [2, 16], the adoption of such a system, particularly in the above mentioned application areas, is severely hindered by prevalent privacy concerns [2, 10–12, 33]. As we have shown in Section 3, these concerns do not only involve end-users, i.e., natural persons, but also – at least equally important – are a huge deal breaker for providers of services that operate on IoT data in the cloud. In order to surmount these inherent concerns, we have to shift the choice of data usage from service providers and developers to the individual users and offer them a transparent guarantee that this choice will be accepted and adhered to [33]. Hence, a system that aims at successfully integrating the IoT and cloud computing in privacy-critical application areas such as assisted living and public mobility assistance has to fulfill the following core requirements:

Data security: The access to data has to be secured and only be controllable by the owner of this data. However, the security mechanisms have to be

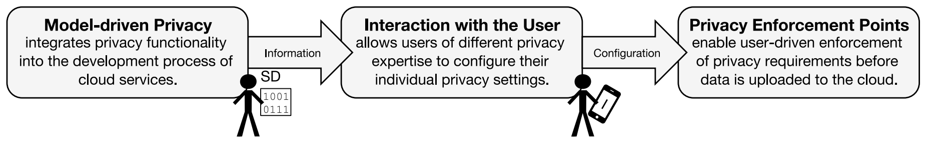

Model-driven Privacy
integrates privacy functionality into the development process of cloud services.
Information
SD
1001
0111
Interaction with the User
allows users of different privacy expertise to configure their individual privacy settings.
Configuration
Privacy Enforcement Points
enable user-driven enforcement of privacy requirements before data is uploaded to the cloud.

Figure 2: High-level overview of the interaction between the three core contributions of our solution for User-driven Privacy Enforcement for Cloud-based Services in the IoT (UPECSI).

flexible enough to account for spontaneous mind changes about privacy when users are in physical danger.

Transparency by design: Privacy has to be integrated into the development process of a cloud service. Most importantly, this includes the documentation of data usage already during the design and implementation of a cloud service in order to improve transparency of data usage for the user.

Efficient privacy-aware development: In order to reduce the additional effort needed for enhancing cloud services with privacy capabilities, we have to provide support for engineering privacy into the service and ease the review of this functionality.

User-controlled data usage and handling: The choice and control over the usage and handling of data has to be shifted from service providers and developers to the individual end-user.

Adaptable user-control: We have to account for different privacy expertises of the envisioned end-users and thus provide an adaptable user-control mechanism. This allows for simplification of complex privacy considerations for privacy novices, while at the same time enables a fine-grained control for privacy experts.

In order to meet these requirements when integrating cloud computing and the IoT, related work provides valuable insights and building blocks. However, in order to meet all these requirements, as well as to address the privacy concerns and considerations as discussed in Section 3 when combining IoT and cloud computing, additional effort is required. Thus, the design and realization of a system for user-driven privacy enforcement for cloud-based services in the IoT is still an unsolved research challenge. Most notably, it is crucial to focus on both the users *and* developers of an IoT/cloud service. In this paper, we address this research challenge by presenting a comprehensive – i.e., focusing on end-users and service developers – approach for privacy-enabled service development in the cloud-based Internet of Things.

6. User-driven Privacy Enforcement for Cloud-based Services in the IoT

In order to address the above mentioned privacy concerns and considerations, as well as to meet the core requirements for a system that integrates

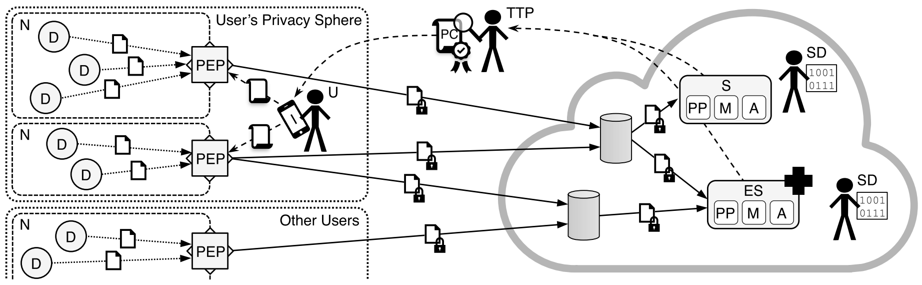


Figure 3: Privacy Enforcement Points (PEP) encrypt all data before it is send to the cloud. Each Service provides a Privacy Policy (PP), Data Usage Monitoring (M), and Audit (A) information to a trusted third party (TTP). The TTP reviews the information from all services and provides default Privacy Configurations (PC) to the user, who views these using her Interface (I) to finally configure her PEP to grant a specific service access to her data.

the IoT and cloud computing in privacy-critical application areas as formulated in our problem statement, we present the design and implementation of UPECSI, our approach for **U**ser-driven **P**rivacy **E**nforcement for **C**loud-based **S**ervices in the **I**oT. While designing and implementing UPECSI, we focus on the agreed requirements for privacy enforcement, which are notice, consent, self-determination, adequate security, and purposeful use [26]. By interacting with the user – which provides transparency – we realize notice, consent, and self-determination. In order to realize adequate security and purposeful use, we rely on technical mechanisms.

Our design and implementation of UPECSI consists of the following three core components as depicted in Figure 2: i) *Model-driven Privacy*, as a novel software development design technique which allows the easy integration of privacy functionality into the development of cloud service, ii) *Interaction with the User* in order to provide transparency for users of different privacy expertise, and iii) *Privacy Enforcement Points*, which reside on the IoT network gateways and enable the user to enforce her privacy and security requirements when her potentially sensitive data is outsourced to the cloud. On a high level, these core components interact as follows: Model-driven Privacy allows us to automatically retrieve information from the development process and generate an interactive user-configurable, service-specific privacy policy. This information is then consulted to interact with the user and hence derive an individual privacy configuration. Finally, the individual privacy configuration instructs the Privacy Enforcement Point on how to enforce this specific user's privacy.

In Figure 3 we provide a more detailed overview of UPECSI. We illustrate the core components and their interconnection. First, with the novel software development design technique Model-driven Privacy, we can automatically derive a service-specific *privacy policy (PP)*. A *trusted third-party (TTP)* audits the correct implementation of a cloud service and the *data usage monitoring (M)* based on *audit information (A)* which we provide based on the information given by the service developer during the development process. If the user authorizes a service access to the data collected by her IoT network, she can review the audited policy together with a default *privacy configuration (PC)* recommended by a trusted third party on her *interface (I)* (e.g., a smartphone or web browser). On her interface, the user takes the decision whether and under which conditions she allows a service to access her data. By this, we realize user consent. Finally, the *Privacy Enforcement Points (PEP)* enable the user to control the access to her potentially sensitive data based on the user's decision. This allows us to guarantee user-adequate security and consent.

In the remainder of this section, we present and explain the core components of UPECSI, model-driven privacy, interaction with the user, and privacy enforcement points, in more detail.

6.1. Model-driven Privacy

Considering privacy during service development requires additional effort from the service developers and potentially leads to more complexity. One approach to reduce complexity and ease the developer's job is to rise the abstraction of the programming language.

As part of UPECSI, we introduce the Privacy Development Language (PDL) in order to allow cloud service developers to describe their privacy considerations while at the same time having object oriented programming elements such as classes and methods. This way, we interconnect the service's data model and privacy description during service development. As stated earlier, it is necessary to enforce access control on users' data when handing it over to a service. A model written in PDL expresses this user data structure along with information about the data usage within the service. This information is used to provide the user with as much details on the usage of her data as possible to enable an informed decision about the privacy configuration she employs for her data. We defer the description of the end user's interface to Section 6.2 and focus now on the developer, who uses the PDL.

To realize the PDL, we extend UML/P [61, 62, 65], a developer-friendly version of UML for modeling data structures and methods in a Java-like syntax.

Following Pearson's "top six" for software engineers [26], we identified *Privacy Policy* as well as *Monitoring Data Usage and Auditing Data Processing* as mandatory for services to meet users' privacy demands. These measures are highly dependent on the data structures and methods used within the service. We use the information modeled in PDL to automatically derive the privacy policy and monitoring of data usage. Following, we describe the generation of these parts from the PDL and the course of auditing the processing of data.

6.1.1. Privacy Policy

Commonly, a privacy policy is composed of service specific information and general liability disclaimers, required for every service and provided by the legal division of the service provider. As noted before, the service-specific information is often very vague, due to the lack of more precise information from developers. With the help of the PDL, we make detailed information on the usage of user

```
package de.rwth.aal;
import java.util.List;

classdiagram AmbientHealthSystem {

  class Camera {
    Room location;
    List<Person> recognizedPersons;
    VideoData stream;
  }

  class Monitoring {
    <<use="Determine whether resident needs help.",
      condition="Sensors detect that resident needs help.",
      mandatory="Camera.recognizedPersons">>
    boolean detectCriticalHealth();

    <<condition="Testified health professional detects that
                 resident needs help.">>
    boolean healthProfessionalDetectCriticalHealth();

    <<use="Ad funded service", unused="If you do not want to get
           advertisements, the service costs $1 per month",
      optional="Camera.recognizedPersons">>
    Ad getPersonalizedAd();
    List<MedicalCase> medicalHistory;
  }

  class NotificationManager {
    <<use="Call an ambulance to my home.", optional="Camera.location">>
    void placeEmergencyCall();

    <<use="Call family in case of medical incident.",
      optional="Camera.stream">>
    VideoData openVideoCall();

    <<use="Display all medical information to the emergency doctor.",
      optional="Monitoring.medicalHistory">>
    void provideMedicalHistory();
  }
}
```

Listing 1: Example PDL model from ambient assisted living

data available, which allows us to automatically derive the service-specific part of the privacy policy. Commonly used privacy policies leave only one choice to the user: Accept the policy and use the service as a whole or do not use it at all. With UPECSI, we provide a more decent view on this decision as we enable users to use only parts of a service, provide only the necessary subset of data, and hence customize the privacy policy to their specific perception of privacy. In addition we empower users to enforce their individual privacy policy on their data by means of the PEP as discussed in Section 6.3.

We illustrate our PDL concept for achieving customizable privacy policies using two examples from our application areas. Listing 1 illustrates an example implementation of a cloud-based service in the area of assisted living, while Listing 2 depicts a cloud-based service for public mobility assistance. These PDL models are written by a service developer, who creates a health-care assisted living, respectively, public mobility assistance service. For each attribute, i.e., customer data originating from an IoT device, the service developer has to

```
package de.rwth.mobility;
import java.util.List;

classdiagram MobilityService {

  class RoutingService {
    <<use="Find people near me to help me on my trip.",
      optional="User.location">>
    void addPotentialHelpers();

    <<use="Respect the shopping list when calculating the route.",
      optional="User.shoppingList">>
    void addShoppingList();

    <<use="Plan your current route.", mandatory="User.location">>
    void planRoute();

    <<use="Inform someone else about where I am and where I am heading to.",
      optional="RoutingService.currentRoute">>
    Route retrieveMe();

    <<condition="A family member is searching for me.">>
    boolean familiyIsSearchingForMe();

    <<description="Next places to visit">>
    Route currentRoute;
  }

  class User {
    Location location;
    List<String> inabilities;
    List<String> shoppingList;
  }
}
```

Listing 2: Example PDL model from mobility scenario

specify the usage of customer data within this service respective method with the `use` stereotype (see, e.g., method `placeEmergencyCall` in Listing 1). This information allows to express data usage in a detailed, human-readable way. To describe users' data in the privacy policy, the attribute names from PDL are used. In case these attributes are of solely technical nature, e.g., a framework requires to implement a data structure with specific attribute names and hence users might have problems understanding these terms, the `description` stereotype may be used to describe the attribute from a non-technical perspective (see variable `currentRoute` in Listing 2 for an example). We defer the discussion on how this information can be perceived by inexperienced users to Section 6.2.

In order to allow the user to customize the functionality of a service to her individual privacy requirements, the service developer can declare parameters of the individual service methods, i.e., the user-data a method operates on, as `mandatory` or `optional` using the respective stereotype. Hence, each `optional` stereotype (see, e.g., method `openVideoCall` in Listing 1) indicates a user-choice in the privacy policy. In case the service functionality changes if a method is disabled by the user, the consequences have to be specified in human-readable form using the `unused` stereotype. For example, in Listing 1 the user has two options with respect to the method `getPersonalizedAd()`: i) allow the cloud service to use the list of recognized persons to display advertisements and by this fund the operation of the service, or ii) do *not* allow the cloud service to use this data to display advertisements and instead pay a monthly fee of $1 for using this cloud service. Hence, for each `optional` method, the user has to choose between the two mutually exclusive options `use` and `unused`.

As emergency services (see Section 2.2) are used in a strictly time-constrained manner, it is infeasible to present a privacy policy to the user when the data is actually needed (i.e., in case of an emergency). Instead, the user already decides about her privacy choices for an (emergency) service during the upfront setup of this service. An emergency event detected by a service or indicated by a trusted third party results in a method call within the service implementation, as the service needs to react upon this event. In order to enable the user to specify which condition should be treated as an emergency, we provide the PDL keyword `condition`. This keyword marks methods which detect such emergency conditions (see method `familiyIsSearchingForMe` in Listing 2 for an example) and give a description to the user. The generated flexible privacy policy then allows the user to specify which data should be provided to the emergency service if certain conditions indicate an emergency. In case a `condition` method does not rely on data from the IoT network of the user, the emergency condition may be determined externally, e.g., by a trusted third party. As such a method can function properly even in cases the IoT network of the user is not functional, e.g., in case of a car accident, the flexible privacy policy allows the user to choose in which situations she rates availability of a service higher than privacy enforcement.

6.1.2. Monitoring Data Usage and Auditing Data Processing

Using the interactive privacy policy, users define how their data *should* be used by a service. In order to give them transparency over who actually *did* access their data, we additionally introduce a database abstraction layer that monitors access to data. This layer, which is provided and operated by the provider of the cloud platform, logs every access of a cloud service to a user's data, i.e., each access to an attribute with a `use` stereotype. This allows the user at any time to retrieve a detailed statement on which method of a specific cloud service accessed which parts of her data, at which point in time, and for which specific purpose. As the collected log on data usage itself reveals potentially sensitive information, it has to be protected. For this purpose, we employ object security mechanisms (similar to those employed by the PEP which we describe in Section 6.3.1) to ensure that only the owner of the data that has been accessed can view the corresponding log file. In order to guarantee that no one can tamper with the log file, i.e., modify or delete lines, we can additionally apply a secure logging scheme, e.g., the one proposed by Ma and Tsudik [70].

While being aware of the services which access her data, the user needs to be guaranteed that the individual methods of a specific cloud service only use her data as stated by the service developer in the `use` stereotype. However, a typical user of our system (see Section 2) does not have the expertise to verify this by herself even if the source code would be freely available, which often is not the case because the service developer might want to protect his intellectual

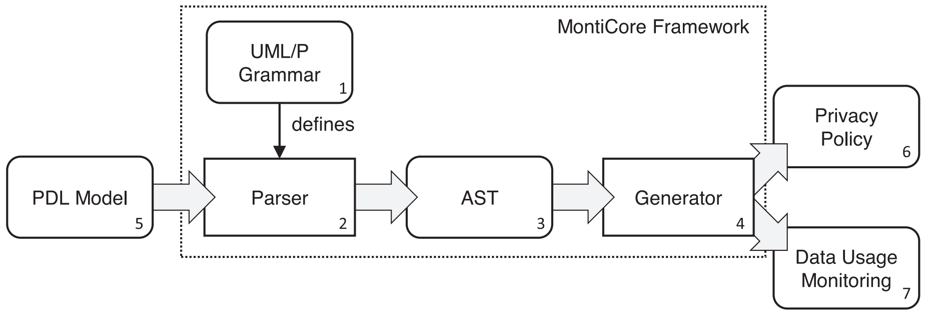


Figure 4: Interplay of components while generating code from a PDL model. Boxes with sharp edges are components which process artifacts that are depicted as boxes with round edges.

property. In order to offer the *audit of the processing of data*, we introduce a trusted auditor, e.g., the provider of the cloud platform or a certification authority, that has to verify the data usage in the service implementation. We designed the description of data usage such that it is directly connected to the method implementing the corresponding functionality. This enables the auditor to efficiently perform a detailed review, as he only has to review those parts of a cloud service implementation that operate on privacy-relevant data and is aware of their promised and thus expected behavior. By reviewing the application-specific authorization implementation part of flexible access control, the auditor checks that emergency conditions in deed rely on the trusted third party granting access. The provider of the cloud platform will only allow to operate and thus make available to users those cloud services, which have successfully passed a data processing audit. This allows us to guarantee the user that cloud-based services actually use her data only as specified in their corresponding privacy policy.

6.1.3. Generating Executable Code from PDL Models

Following the model-driven development approach, a PDL model needs to be executable [59] to be of value for developers. In order to realize this, we utilized the UML/P implementation, which is part of the MontiCore Framework (see Section 4.4), and implemented a generator which automatically derives the implementation specific part of a service's privacy policy and data usage monitoring from a PDL model. Figure 4 depicts an overview of the implemented generator within the MontiCore Framework. MontiCore provides the UML/P (1) which we extended to realize the PDL. From the UML/P language defining grammar, the MontiCore Framework derives the parser (2) as well as an Abstract Syntax Tree (AST) data structure (3) which is used in the core generator (4) to access the parsed information from the PDL model (5).

The process of transforming a PDL model to executable code (6, 7) by these components works as follows: First the parser (2) parses the PDL model from a file and creates an AST (3) which provides the essential information from a

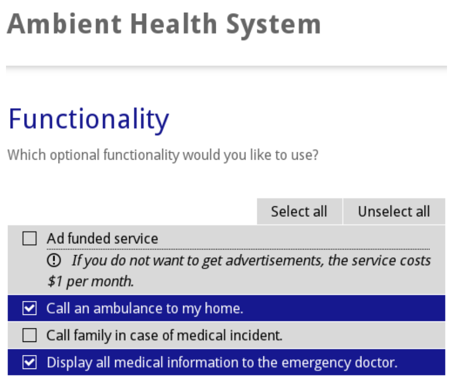
PDL model, e.g., by leaving out syntactical sugar such as brackets, whitespaces, and newlines. The generator (4) essentially consists of several Freemarker templates [71] which are called by MontiCore after parsing. A template, being a prototype of an output file, contains variables that need to be assigned a value during the generator run in order to produce a concrete output file. These variables are assigned based on data from the AST (3) which is provided to the generator. The Freemarker template language provides control sequences which allow to directly feed values from the AST into the generated file. However, more complex computations on the AST are written in Java and called from within the template when the calculated value is needed. Using this setup when processing a PDL model, the generator produces a web application (6) which presents the service's interactive privacy policy to the user. To support Data Usage Monitoring in services, the generator derives service-specific Data Access Objects (DAOs) (7) which log every access to user data from the PDL model. As discussed in Section 6.1.2, these DAOs are then used within the service as interface to the database.

6.2. Interaction with the User

In order to realize notice and consent, user interaction with UPECSI requires special consideration. When referring to the application areas of Section 2.1, we must consider that UPECSI's approach to privacy may be applicable to users of different levels of privacy expertise, ranging from privacy experts able to make informed decisions up to absolutely IoT-privacy novices – such as our 75-years-old widowed female retiree. In this section, we first present configurable privacy policies for privacy experts before we discuss an approach for default privacy settings for privacy novices. Finally, we argue on the social context of user interaction.

6.2.1. Configurable Privacy Policies for Privacy Experts

Technically speaking, a user considered an IoT-privacy expert may use her interface to read on the most detailed scale through the generated privacy policy of every service she uses. Figure 5 shows the generated privacy policy from the ambient assisted living example depicted in Listing 1. As shown in Figure 5a, the experienced user can choose which optional parts of the ambient assisted living service she wants to use. If the user decides to enable the advertisement funded service, the notification about the monthly fee disappears. Figure 5b shows the data fields (class attributes) to which the service needs access in order to realize the functionality selected by the user. Users are able to review the used data for each combination of service functionality and choose the one that fits their privacy needs. Similarly, Figure 6 shows the adaptable privacy policy that was generated for the interactive mobility assistance service depicted in Listing 2. In order to express flexible access control rules as discussed in Section 6.3.2, the user utilizes the condition editor shown in Figure 5c and Figure 6c. In this context, the decision which options to choose from the privacy policy or which data to submit to the service, respectively, is sent as the user's privacy configuration to the PEP which will enforce this privacy decision.



Ambient Health System

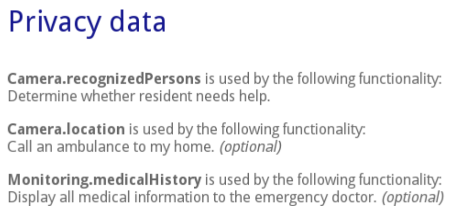
Functionality

Which optional functionality would you like to use?

Select all | Unselect all

- [ ] Ad funded service
  - ⓘ *If you do not want to get advertisements, the service costs $1 per month.*
- [✓] Call an ambulance to my home.
- [ ] Call family in case of medical incident.
- [✓] Display all medical information to the emergency doctor.

(a) Interactive selection of functionality.



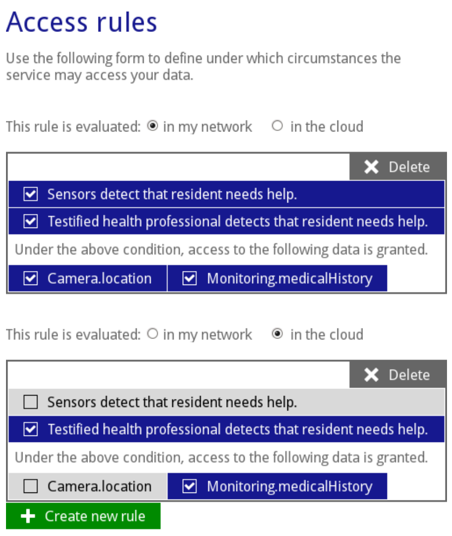
Privacy data

Camera.recognizedPersons is used by the following functionality:
Determine whether resident needs help.

Camera.location is used by the following functionality:
Call an ambulance to my home. *(optional)*

Monitoring.medicalHistory is used by the following functionality:
Display all medical information to the emergency doctor. *(optional)*

(b) Overview over resulting data usage.



Access rules

Use the following form to define under which circumstances the service may access your data.

This rule is evaluated: [✓ in my network] in the cloud

Delete

- [✓] Sensors detect that resident needs help.
- [✓] Testified health professional detects that resident needs help.

Under the above condition, access to the following data is granted.

- [✓] Camera.location
- [✓] Monitoring.medicalHistory

This rule is evaluated: in my network [✓ in the cloud]

Delete

- [ ] Sensors detect that resident needs help.
- [✓] Testified health professional detects that resident needs help.

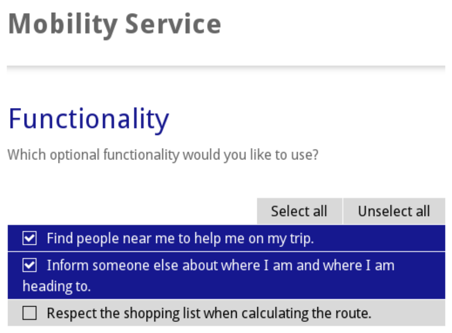
Under the above condition, access to the following data is granted.

- [ ] Camera.location
- [✓] Monitoring.medicalHistory

Create new rule

(c) Condition editor for realizing flexible access control.

Figure 5: Screenshots of the automatically generated adaptable privacy policy for the ambient health system.



Mobility Service

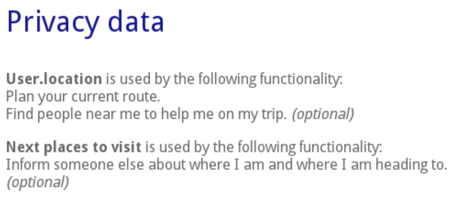
Functionality

Which optional functionality would you like to use?

Select all | Unselect all

- [✓] Find people near me to help me on my trip.
- [✓] Inform someone else about where I am and where I am heading to.
- [ ] Respect the shopping list when calculating the route.

(a) Interactive selection of functionality.

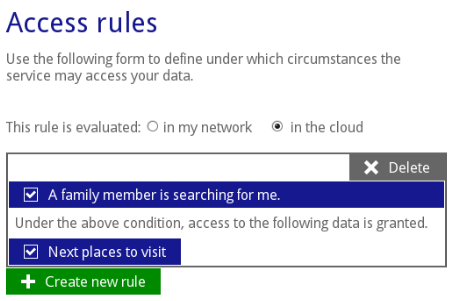


Privacy data

User.location is used by the following functionality:
Plan your current route.
Find people near me to help me on my trip. *(optional)*

Next places to visit is used by the following functionality:
Inform someone else about where I am and where I am heading to. *(optional)*

(b) Overview over resulting data usage.



Access rules

Use the following form to define under which circumstances the service may access your data.

This rule is evaluated: in my network [✓ in the cloud]

Delete

- [✓] A family member is searching for me.

Under the above condition, access to the following data is granted.

- [✓] Next places to visit

Create new rule

(c) Condition editor for realizing flexible access control.

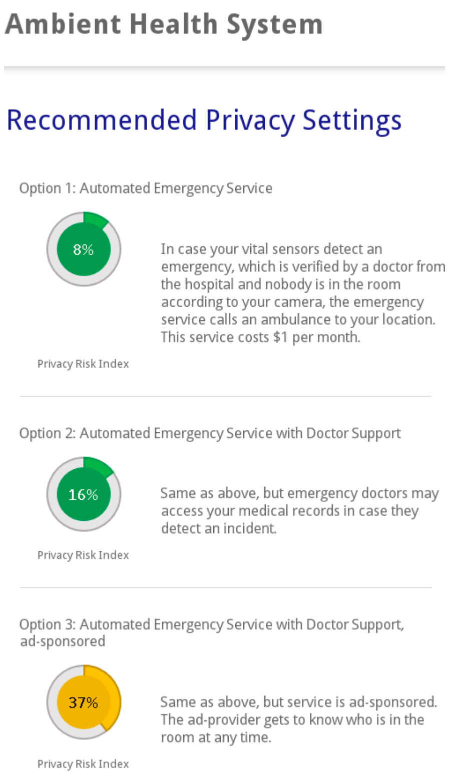
Figure 6: Screenshots of the automatically generated adaptable privacy policy for the mobility service.

6.2.2. Default Privacy Settings for Privacy Novices

In contrast to this very detailed privacy policy, default privacy settings at different layers of abstraction are provided by trusted third parties to support IoT-privacy novices in their decision about privacy configurations. Figure 7 shows a possible abstraction layer for privacy novices provided by a third party institution for the assisted living resp. mobility example. In Figure 7a, the user can choose between three default privacy configurations for her ambient health system. Here, the first option provides the most privacy protection, but also limits functionality and requires a fee. The second option represents the privacy configuration shown in Figure 5 and provides slightly less privacy in trade for more functionality. Finally, the third – ad-funded – option provides the least amount of privacy. Similarly, the default privacy policies in Figure 7b offer the user a choice between two options. Here, the second option (which resembles the privacy configuration from Figure 6) again offers a reduced level of privacy in order to realize more functionality. To create such various abstraction layers of privacy settings, third parties use their privacy expertise – e.g., such as metrics like the "Privacy Risk Index" depicted in this example – and the detailed descriptions from the adaptable privacy policy. Although using this recommended policy, the user optionally may still change individual settings in her privacy configuration referring to the detailed adaptable privacy policy shown in Figures 5 and 6. By offering such interaction possibilities, the user is continuously assured control and verifiability of her data regardless of her factual privacy-related technical expertise, which leads us to finally focus on the social aspects from this technical description of user interaction.

6.2.3. Social Context of User Interaction

UPECSI highlights not only a technical but also a social, i.e., situated and collective, context [72] by constraining what information is collected, as well as with whom and under which conditions information is shared [73]. Thus, the system remains adaptable to context-specific norms of different real-life application areas (ranging from our presented assisted living application area to the mobility assistance outlined in Section 2). Furthermore, this is an useful approach to maximize transparency [74] of the whole system which will foster users' trust in UPECSI. To illustrate this, reconsider the user interaction depicted in Figure 7 again: The introduced privacy-novice uses several services with recommended policies from trusted third parties which may range, e.g., from family members' recommendations to such of certified home care service providers (HCPs). Such third parties may be institutions or crowd-based voting systems analyzing the privacy policy of a service and judging about the privacy trade-off in place of the user (or, on a less rigid level, giving at least recommendations to those IoT-privacy novices). In any case, a user relies on the verdict of an auditor, and an institution guarantees granting access under clearly specified conditions. For example, she may decide in advance which information from which types of sensors should be revealed permanently or temporarily to her HCP or to any medical personnel in case of an actual emergency. But all recently discussed privacy concerns and accompanying efforts are worthless, if


Ambient Health System

Recommended Privacy Settings

Option 1: Automated Emergency Service

In case your vital sensors detect an emergency, which is verified by a doctor from the hospital and nobody is in the room according to your camera, the emergency service calls an ambulance to your location. This service costs $1 per month.

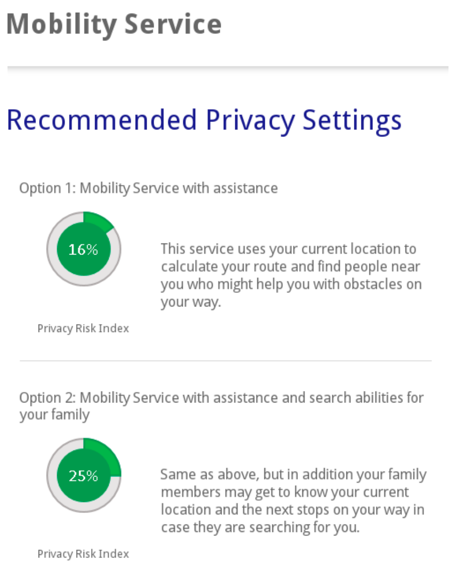
Option 2: Automated Emergency Service with Doctor Support

Same as above, but emergency doctors may access your medical records in case they detect an incident.

Option 3: Automated Emergency Service with Doctor Support, ad-sponsored

Same as above, but service is ad-sponsored. The ad-provider gets to know who is in the room at any time.

(a) Ambient Health System


Mobility Service

Recommended Privacy Settings

Option 1: Mobility Service with assistance

This service uses your current location to calculate your route and find people near you who might help you with obstacles on your way.

Option 2: Mobility Service with assistance and search abilities for your family

Same as above, but in addition your family members may get to know your current location and the next stops on your way in case they are searching for you.

(b) Mobility Service

Figure 7: Screenshots of default privacy settings for privacy novices provided by a trusted third party.

UPECSI's usability is flawed. Therefore, the condition editor shown in Figures 5c and 6c is used to present the privacy policy about which options to choose in an understandable and specifiable way. For instance, while it may absolutely not being necessary for the depicted non-experienced user to understand the technical details of UPECSI (i.e., data protection, annotation enforcement, flexible access control, dynamic privacy policies, data monitoring, and auditing of data protection), those users familiar with such techniques may choose their own configurations and type of personal information involved, i.e., characteristics of the data-gathering technique, the form of the data, spatial, and locational aspects. To sum up, as [30] puts it, trust in the information society is based not only on calculus and knowledge, but on social reasons as well. Here, the notion of trust is being defined as the user's certainty that the system is capable of providing the required services accurately and infallibly, i.e., sound operations, effective security mechanisms, abidance of regulations and laws, while at the same time, besides all, the notion of trust always contains the acknowledgement of a minimum risk factor. We will deepen the discussion on the user acceptance of UPECSI in Section 7.6.

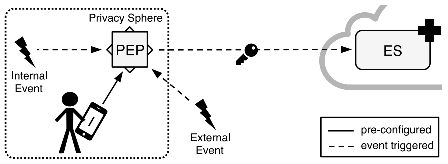
6.3. Privacy Enforcement Points

After a user has decided on her specific privacy configuration, this privacy configuration has to be enforced whenever IoT data leaves the control sphere of this user. In our application areas and network scenario, the connection of an IoT network to the cloud is realized using a gateway (see Section 2). Notably, a single user can operate an arbitrary amount of IoT networks and hence arbitrarily many gateways. We identify these gateways as the distinct boundaries at which potentially sensitive data leaves the control sphere of the user. Hence, the enforcement of a user's security and privacy requirements has to be realized here. For this, we introduce the *Privacy Enforcement Point (PEP)* as a new logical entity on each of these gateways. There, the PEP acts as the representative of the user and allows her to remain in control over her data, even if it leaves her secured IoT network and is outsourced to the potentially untrusted cloud. To achieve this goal, the PEP has to fulfill three tasks: i) data protection and access control, ii) flexible access control, and iii) enforcement of data handling requirements. In the following, we discuss these tasks of the PEP in more detail.

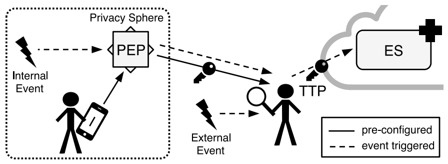
6.3.1. Data Protection and Access Control

In order to protect access to potentially sensitive data when it leaves the secured network of the user, we provide cryptographic access control which makes use of encryption. For this, we base the data protection and access control mechanisms of UPECSI on the SensorCloud security architecture [3, 5, 7, 39]. The underlying idea here is to encrypt all data using a symmetric data protection key before it is upload to the cloud. In order to allow a cloud service access to specific data, the PEP then releases the corresponding data protection keys for this cloud service. This way, we protect the confidentiality of potentially sensitive data and prevent unauthorized access to it. In order to allow more fine-grained access control, we periodically exchange the data protection keys, which allows to restrict access to certain periods of time.

More precisely, the PEP encrypts each piece of data only once, irrespective of the number of authorized cloud services, using a data protection key and leveraging a symmetric cipher (e.g., AES in CBC mode). When the user instructs the PEP to authorize a cloud service to access (specific parts of) her potentially sensitive information, the PEP has to release the corresponding data protection key for each authorized service. For this, the PEP encrypts the data protection key with the public key of each authorized cloud service (e.g., using the asymmetric cryptosystem RSA) and uploads it to the cloud. Only an authorized cloud service may then decrypt the encrypted data protection key using its private key and, subsequently, decrypt only those data it is authorized to access. Notably, nobody else, not even the operator of the cloud, can gain access to the protected data. Hence, using the PEP, we provide an effective measure to securely restrict access to potentially sensitive data and allow only authorized



(a) Flexible access control enforced by Privacy Enforcement Point.



(b) Flexible access control enforced by external trusted third party.

Figure 8: The user can leverage her Privacy Enforcement Point (a) or an external trusted third party (b) to realize flexible access control based on internal and external events.

entities to access this data. Furthermore, this access can be restricted to specific parts of a piece of data and limited to a specific time range.

6.3.2. Flexible Access Control

Using our basic data protection and access control as described above, only authorized cloud services can access a user's data. However, in certain situations we have to account for a trade-off between privacy and safety, e.g., health (see Section 3). In case of a medical emergency, the user would most likely be willing to completely give up her privacy if this could help to save her life. When using fixed and predefined cryptographically secured access control mechanisms (as presented in Section 6.3.1), this is per se not possible. Hence, UPECSI additionally supports *Flexible Access Control*. The underlying idea here is to automatically authorize a cloud service based on observed internal and external events. Here, internal events originate from one of the user's IoT networks and can, e.g., be triggered based on the measurements of an IoT device that indicates a critical health condition. Contrary, external events originate from outside the user's IoT networks and are typically triggered by a trusted third party, e.g., a health professional or emergency center observing a critical health condition. Notably, also the combination of internal and external events is possible. For example, an emergency doctor could only be allowed to access a user's data, if his statement that this user is unconscious is consistent with readings from the user's vital signs monitoring device. As these examples show, the choice between respectively combination of internal and external events directly influences the privacy of the individual user. We will deepen the impact of this choice on the user's privacy in our discussion in Section 7.

In order to realize flexible access control, the user has to define upfront who should get access to her data whenever a specific internal or external event is triggered. For example, she could choose that any emergency doctor gets access to her data in case of a medical emergency as testified by her personal vital signs monitoring device. This way the user can ensure that medical personal is always granted access to her medical data when needed to save her life. From a technical perspective, flexible access control requires to dynamically release the necessary data protection keys (see Section 6.3.1) to a previously not authorized cloud service based on observed events. For this, the user can decide to either leverage the PEP or an external trusted third party. We depict this two alternatives in

Figure 8. Using the PEP prevents potential misuse while an external trusted third party mitigates the negative effects of a single-point-of-failure. Again, we will deepen the impact of this trade-off between privacy and safety in our discussion in Section 7 and in the following focus on the technical realization of the two alternatives.

If the user decides to rely on the PEP for performing flexible access control (depicted in Figure 8a), the PEP generates a new public/private key pair upfront for each flexible access control policy and uses the corresponding public key to continuously release the necessary data protection keys encrypted to the cloud (see Section 6.3.1). Using a new public/private key pair here is necessary to prevent that an emergency service can access the data *before* the potential emergency took place. Based on the measurements from the IoT network and notifications over external events, the PEP constantly evaluates whether a flexible access control policy should be executed. In this case, the PEP encrypts the private key it created for this policy with the public key of the corresponding emergency service and releases it to this service. The emergency service can then decrypt the private key and thus also decrypt the data protection keys needed for accessing the actual data.

Should the user decide to rather use a trusted third party for managing flexible access control (shown in Figure 8b), the PEP again creates a priori a new public/private key pair for each flexible access control policy. Again, the required data protection keys are continuously released encrypted to the cloud using this public/private key pair. The PEP then, already when setting up a new flexible access control policy, encrypts the private key for this policy with the public key of the to-be-authorized emergency service and hands this encrypted key over to the external trusted third party. The trusted third party securely stores the encrypted key (which the trusted third party itself cannot access as the key is encrypted) and constantly checks for the (internal and external) events listed in the flexible access control policy. As soon as it observes the listed events, it releases the encrypted private key to the emergency service. Again, the emergency service can then decrypt this private key, subsequently decrypt the data protection keys, and finally access the data. We will further focus on the benefits of this approach and its implications on privacy and security in our discussion in Section 7.

6.3.3. Enforcement of Data Handling Requirements

As discussed in Section 3, privacy requirements do not only consist of access to data but also impose restrictions on how data is handled, e.g., where data is transferred to, stored, and processed. In order to allow the user to require the enforcement of these so-called data handling requirements [75], the PEP allows the user to add data handling annotations [12] to her data. For each IoT device, i.e., data source, the user can provide data handling annotations that should be attached to data that originates from this device. Using data handling annotations, a user can, e.g., specify that certain data is not allowed to leave her network at all, other data can leave the network but must only be forwarded to cloud offers in a certain jurisdiction, while a third group of data can be transferred to any cloud provider. First, the PEP decides, based on the annotation, whether data is allowed to leave the controlled network and if so, only forwards the data to eligible cloud offers. Before any data leaves the IoT network, the PEP attaches a digital data handling annotation to each piece of data [12]. These annotations contain obligations on how a users' data has to be handled in the cloud. Such annotations, e.g., can impose restrictions on where and how long data is allowed to be stored. In the cloud, these data handling annotations can be interpreted and enforced by a data handling requirements-aware cloud stack [12, 75]. We will discuss in the next section how this contributes to our goal of empowering users to keep control over their privacy when outsourcing data collected by IoT devices to the cloud.

7. Discussion

After having presented the cornerstones and individual components of UPECSI in detail, let us examine how our approach addresses and overcomes the development and privacy challenges and considerations presented in Section 3. We show for each of the challenges (data security, transparency by design, efficient privacy-aware development, user-controlled data usage and handling, and adaptable user-control) we derived in our problem statement (Section 5), how we approached and solved it. Whilst doing so, we particularly focus our discussion on the individual user, as achieving user acceptance is the most crucial driving force of our work. We close this section with a discussion on how the user acceptance of UPECSI can be evaluated and present preliminary results of this evaluation.

7.1. Data Security

As a core challenge of realizing UPECSI, we identified the securing of access to data such that only the owner of this data can grant data access, while at the same time allowing access to preselected emergency services in case of need. In order to tackle this challenge, we introduced Privacy Enforcement Points (PEP), which secure data before it is uploaded to the cloud (see Section 6.3). As all data objects are encrypted before they leave the secured network of the user, per se no one except the user can access it. Thus, without further action, all data remains private. Only if the user decides to grant a cloud service access to specific data, this selected cloud service gains insight to the data. Notably, UPECSI allows the user to restrict data access to only those parts of the data that are absolutely needed and at the same time provides an audit-based guarantee that this data is only used for the intended purpose. We will further focus on the benefits of this with respect to transparency and user-controlled data usage in the following sections and for now continue focusing on data security.

Notably, as both data and data protection keys are stored in the cloud, data can be accessed even if PEP is currently not reachable. More specifically, even if an attacker performs a denial-of-service (DoS) attack [76] against the IoT network of a user in order to block access to this network, both data and services operating on this data will not be impaired. However, an attacker can still block, e.g., using a DoS attack, the upload of *new* IoT data to the cloud. Still, as all cryptographic operations of UPECSI happen either based on a timely schedule or upon request of the user owning the device, the attacker cannot trigger (costly) cryptographic operations to significantly increase the impact of his DoS attack [77]. Hence, UPECSI improves the protection against DoS attacks on a user's infrastructure when compared to solutions that realize storage and processing of IoT data within the IoT network itself as the system's pre-attack state remains fully available with services still operational on this state. A full protection of the user's infrastructure against DoS attacks and the even more powerful distributed DoS attacks is orthogonal to our work and has already been widely studied in the literature [78].

Remarkably, data security gets significantly more challenging when considering emergency access to data. We address this challenge using our flexible access control policies (see Section 6.3.2). These allow a user to define upfront, who should get access to her data in case of an emergency, which is trigged by internal or external events. The actual access granting in case of a triggered event is then realized either internally by the user's PEP or externally by a trusted third party. As shown in Section 6.3.2, a user's decision between relying on internal or external events as well as internal or external access granting constitutes a trade-off between privacy and safety. In the following, we will detail our discussion on this trade-off.

The first trade-off the user has to make is the decision between internal (i.e., events originating from her own IoT devices) and external (i.e., events originating from outside the control of the user) events. The advantage of relying on internal events is that the user can be sure that the event is genuine and nobody tries to forge an event in order to get access to her data. This, however, comes at the cost of a single-point-of-failure, as the IoT network which triggers the internal event might temporarily be unreachable. Here, external events, which can be triggered by a multitude of entities, provide more reliability and thus safety. Notably, UPECSI also supports the combination of internal and external events. This allows the user to configure that normally internal and external events have to be consistent, but still allows to use only external events in case of network disruptions as a fail-over.

For the second trade-off the user has to decide whether her PEP or an external trusted third party should release the necessary data protection keys in case of an emergency. Both solutions cryptographically guarantee that the data protection keys are only released to an entity that the user authorized upfront. However, if realized using an external trusted third party, the user has to trust that events are monitored and evaluated properly and data protection keys are indeed only released in case of an emergency. Again, utilizing the PEP for this task provides stronger privacy but bears the risks of a single-point-of-failure.

These trade-offs between privacy and safety have to be decided by each user on her own. Hence, in UPECSI we empower the user to take this decision separately for each individual piece of data and application area. However, as we will discuss in Section 7.5, we additionally account for privacy inexperienced users by providing policy recommendations from trusted third parties. This allows us to establish data security as the core foundation to ground user trust in the privacy of UPECSI.

7.2. Transparency by Design

Beside data security concepts and reusable components, service developers need to consider privacy when designing and implementing the system's functionality and data structures. To support them in this activity, we developed the PDL as presented in Section 6.1. Using the PDL, developers can express the data structures and service functionalities during system development using UML class diagrams in a textual Java-like language, being familiar with from their daily development tasks. Using the PDL keywords as discussed in Section 6.1 they describe the usage of data within their service on a per-attribute and per-method level. This allows for a much higher level of detail than other approaches might offer, as presented in related work. Notably, the privacy description is integrated already into the design process at a point where system designers of conventional systems make the actual decision about users' privacy. Hence, our approach results in a much more straightforward way to create a privacy description as compared to the reverse engineering of this description after the implementation has already been finished.

The details provided by the service developer using our PDL are crucial for providing transparency to end-users, for informing them about privacy implications when using the service, and allowing them to make a profound decision about there privacy. As the service developer is forced to entrench a detailed description of the usage of data for each utilized attribute already at the time of development, the users gain full and fine-grained transparency by design over the potential usage of their data *before* they decide to entrust a specific service with their potentially sensitive data.

Furthermore, our approach for monitoring and auditing the usage of data as presented in Section 6.1.2, allows the user and trusted auditors to check at any point in time whether the stated data usage descriptions are indeed adhered to by the actual implementation of a cloud service's functionality. With UPECSI we enable the individual user to, e.g., verify which specific method of a cloud service has accessed which parts of her data at which point of time. Especially for privacy experts, this for the first time enables full transparency over the usage of data in the cloud.

7.3. Efficient Privacy-aware Development

Privacy – like any other quality attribute – does not arise spontaneously without any extra effort during development. Thus, one goal of UPECSI is to minimize the additional effort of the developer while maximizing the amount of privacy features of the resulting cloud service. As our PDL captures privacy information along with the data structures and functionality design, developers can describe privacy information during the process of designing and implementing the cloud service. This way developers do not have to describe the privacy information in a separate, later step, e.g., when preparing the cloud service for release, but privacy is built in during development.

To maximize the effect and utilize the strength of the model-driven approach when developing structurally identical systems, which differ only in the data model, we provide with UPECSI a generator which allows to derive a privacy policy and a database abstraction layer for data monitoring from a PDL model. When functional requirements for the service change, the data structure and functionality need to be changed and so do the data `use` descriptions. As UPECSI supports a model-driven development process, where the PDL model is a primary development artifact, all these changes are performed within the PDL and the privacy policy and data monitoring get automatically re-generated. Hence, during service evolution, the privacy information is kept up-to-date and consistent within the service and it's representation for the user.

As cloud services in UPECSI operate on unencrypted data, the user must trust the cloud service implementation. To establish this trust, we employ two mechanisms: i) We connect the privacy policy with the services' source code which allows us to make the data processing within the service transparent to the user. This is achieved by adding `use` statements to methods in the PDL. ii) We introduce an auditor who reviews the service's implementation and ensures, that the service handles the data exactly as described in the `use` statement. As the `use` corresponds to a statement within the actual source code, the audit is much simpler and provides more details compared with a state-of-the-art black box data privacy audit of a whole application.

With these measures, we significantly reduce the extra effort needed for engineering privacy functionality into the service and, at the same time, ease the review and audit of the service's usage of data.

7.4. User-controlled Data Usage and Handling

In order for users to accept and trust UPECSI, we shifted the choice and control over the usage and handling of data from the service providers and developers to the individual end-user. For this, UPECSI realizes *notice*, *consent*, and *self-determination* via the generated privacy policy as follows. As described in Section 6.2, *notice* is realized by providing the user with a detailed privacy policy which we generate using the information provided by the developer via the PDL. This way, we inform the user about the data a service uses and how it is used within the individual parts of the service. By this, we especially empower experienced users to get a detailed insight into the way the service processes their data. To provide choice and *self-determination* to the user, the privacy policy is customizable. This allows the user to choose which optional functionalities of a service, which require access to additional data, she wants to use. In case the user is willing to provide this data only in case of certain events, she can use the condition editor to specify flexible access control policies. The resulting data usage is then displayed in the privacy policy data usage listing. If the user finally agrees with the customized privacy settings, she accepts the policy. Thus, the PEP will grant this specific service access to the data specified by the user. This way, cloud services in UPECSI can only access the data explicitly specified by the user. Hence, we successfully realized user *consent*.

In order to meet a user's privacy requirements, it is not enough to only focus on access to and usage of data. Additionally, users may wish to impose restrictions on the handling of data, e.g., regarding the storage location of data (see Section 3). For this, in addition to the `use` statements provided by the service developer, we allow the user to add data handling annotations to potentially sensitive data when it is processed by the PEP. It is then the task of the cloud provider and PEP to interpret and adhere to these data handling annotations [12, 75]. This helps to increase a user's privacy in two ways. First, the user can configure that certain, extremely sensitive data should not leave her secured network at all. Using data handling annotations, she can instruct the PEP to never forward this data to the cloud. Thus, she has a strong guarantee that this data will stay in her network, even if she might accidentally grant a service access to this highly private information. Second, the user can precisely specify where and for how long her data is allowed to be stored. The cloud provider's adherence to the annotated data handling requirements can then be verified using UPECSI's monitoring of data usage (see Section 6.1.2). This gives the user a certain guarantee that data handling requirements are indeed being followed.

7.5. Adaptable User-control

As previously described, one important feature of UPECSI is to shift the choice of data usage and handling from service providers and developers to end-users. Although this puts the user back in control over her privacy, this also implies that she has to take a lot more responsibility for her privacy. However, due to the focus of our application areas, we have to consider dealing with privacy and technical inexperienced users. Therefore, we have to avoid that privacy novices like the 75-years retiree get overwhelmed by the various options provided by UPECSI. Due to the simple fact that all privacy functionality is worthless if the user cannot handle it, privacy novices need a simplification for complex privacy considerations, while at the same time privacy experts should be attracted by UPECSI offering full control to all privacy functionality.

Hence, it is one crucial point to offer accessibility by providing different levels of functionality for different user groups. However, offering more functionality in terms of adaptable and transparent configurations must additionally relate to different norms of usage. As shown in Section 3, privacy is not a fixed norm: in different contexts, users might trade-in privacy for, in their regard, more or less beneficial goods or services, e.g., unveiling personal health information in case of medical emergencies or quality of life in a nursing home. Finally, adaptability of user-control requires the inclusion of third party assistance, i.e., in our case having the option of being provided with policy recommendations from trusted third parties. These trusted third parties provide for a specific use case and privacy norm (e.g., nursing homes in a specific country) default privacy configurations for specific cloud services. Still, the user can review these default settings and overwrite (parts of) them should she want to do so. By this, we realize an adaptable control for the user over the usage and handling of her data while at the same time still allow for parameterizing a cloud service individually for each user.

7.6. Evaluating User Acceptance

So far, we have only outlined the social context of UPECSI, which now leads us to a tentative evaluation on user acceptance or user experience, respectively. As shown in previous work [23], not until recently have cloud-based services proliferating ambient assisted living solutions (or mobility services, respectively) been subject to explicit sociological analyses. Thus, studies aiming at encompassing insights into how private end-users adhere to such cloud-based services – their areas of application and what users consider as being problematic – often rely on open, exploratory methodological designs; e.g., qualitative, semi-structured interviews (as opposed to the rigorously quantitative survey methodology) which facilitate the creation of a comfortable conversational climate empowering the interviewees to weight different aspects of the respective topic according to their own views. Such a way of interviewing helps to identify central ideas without risking to ignore aspects due to too strongly pre-formed perceptions of the relevant matters.

In [23] we present the conception, realization, and results of such a cloud-related, qualitative methodological approach in detail. Here, we summarize these results as follows: In general, technological approaches such as Cloud Computing and Smart Home (e.g., as eminent building blocks of our ambient assisted living scenario) are appreciated by the interviewees and considered useful when regarding benefits like the supervision and control of health-related vital data or different functions and appliances in the house. As well, this holds true for the less technophiles among our interviewees (e.g., older ones). Addressing potential negative aspects, interviewees' concerns regard primarily the handling and functioning of the technology and its consequences, and thus tackle the question how a respective application might influence everyday life. It is likewise important to transparently communicate a system's features and procedures to potential users in order to support their decision-making and risk-assessment – e.g., prevention of loss of control on the user side by including mechanisms enabling the user to regulate a system's level of autonomous actions – as shown beforehand by discussing the introduction of the PDL and its consequences, as well as by illustrating how the user is represented at the interconnection of IoT and the cloud. To sum up, such a way of assessing user acceptance as presented for the time being here helps to stimulate the technical buildup of UPECSI with insights from its social context.

8. Conclusion

The federation of the Internet of Things and cloud computing with all its benefits is hindered by severe privacy concerns of end-users and privacy considerations of service providers. In order to overcome these severe concerns, especially for privacy-critical application areas such as assisted living and public mobility assistance, we presented our comprehensive approach to privacy in the cloud-based IoT that addresses end-users and service developers at the same time. As result of our work, we presented the UPECSI approach containing of a comprehensive set of technologies together with organizational measures to realize user-driven privacy enforcement for cloud-based services in the IoT.

With our contributions in UPECSI we i) allow individual end-users to protect their potentially sensitive data before it is transferred to the cloud, ii) empower cloud service developers to efficiently integrate privacy functionality into the development process of a cloud service, and iii) provide users an intuitive, adaptable, and transparent user interface which allows them to configure their privacy settings based on their individual privacy experience.

As our discussion shows, UPECSI effectively contributes to a significantly increased level of privacy when integrating the IoT and cloud computing and thus allows to profit from the benefits of cloud computing even in privacy-critical application areas such as assisted living and public mobility assistance.

In the future, we will further validate the user acceptance of our comprehensive approach. For this, UPECSI's further technical build-up will strongly be tied to feedback loops of a participatory design approach [6]. By doing so, we effectively incorporate different interest groups (e.g., end-users, service developers, and experts) into the further development process. This empowers stakeholders as active members of the future development of UPECSI and hence helps us to increase user acceptance.

Acknowledgments

We thank Michael Eggert, René Hummen, Alexander Roth, Kirsten Rüssmann, and Henrik Ziegeldorf for many fruitful discussions and valuable feedback. Additionally, we would like to thank Anna Feininger and Marie Ludwig for their research assistance and especially Lukas Koschmieder for the prototypical implementation of our approach. We also thank the anonymous reviewers and conference attendees of the 2014 International Conference on Future Internet of Things and Cloud for their constructive comments which encouraged us to improve our work. Finally, we would like to thank the anonymous reviewers of Future Generation Computer Systems for their suggestions and comments which helped to significantly improve this paper. This work has been funded by the Excellence Initiative of the German federal and state governments within the HumTec Project House at RWTH Aachen University.

References

[1] L. Atzori, A. Iera, G. Morabit, The Internet of Things: A survey, Computer Networks 54 (15) (2010) 2787–2805. doi:10.1016/j.comnet.2010.05.010.

[2] J. H. Ziegeldorf, O. Garcia Morchon, K. Wehrle, Privacy in the Internet of Things: Threats and Challenges, Security and Communication Networks 7 (12) (2014) 2728–2742. `doi:10.1002/sec.795`.

[3] R. Hummen, M. Henze, D. Catrein, K. Wehrle, A Cloud Design for User-controlled Storage and Processing of Sensor Data, in: 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2012, pp. 232–240. `doi:10.1109/CloudCom.2012.6427523`.

[4] F. Li, M. Voegler, M. Claessens, S. Dustdar, Efficient and Scalable IoT Service Delivery on Cloud, in: 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), IEEE, 2013, pp. 740–747. `doi:10.1109/CLOUD.2013.64`.

[5] M. Henze, R. Hummen, R. Matzutt, D. Catrein, K. Wehrle, Maintaining User Control While Storing and Processing Sensor Data in the Cloud, International Journal of Grid and High Performance Computing (IJGHPC) 5 (4) (2013) 97–112. `doi:10.4018/ijghpc.2013100107`.

[6] M. Eggert, R. Häußling, M. Henze, L. Hermerschmidt, R. Hummen, D. Kerpen, A. Navarro Pérez, B. Rumpe, D. Thißen, K. Wehrle, SensorCloud: Towards the Interdisciplinary Development of a Trustworthy Platform for Globally Interconnected Sensors and Actuators, in: H. Krcmar, R. Reussner, B. Rumpe (Eds.), Trusted Cloud Computing, Springer, 2014, pp. 203–218. `doi:10.1007/978-3-319-12718-7_13`.

[7] M. Henze, R. Hummen, R. Matzutt, K. Wehrle, A Trust Point-based Security Architecture for Sensor Data in the Cloud, in: H. Krcmar, R. Reussner, B. Rumpe (Eds.), Trusted Cloud Computing, Springer, 2014, pp. 77–106. `doi:10.1007/978-3-319-12718-7_6`.

[8] L. Hermerschmidt, A. Navarro Pérez, B. Rumpe, A Model-based Software Development Kit for the SensorCloud Platform, in: H. Krcmar, R. Reussner, B. Rumpe (Eds.), Trusted Cloud Computing, Springer, 2014, pp. 125–140. `doi:10.1007/978-3-319-12718-7_8`.

[9] A. Botta, W. de Donato, V. Persico, A. Pescapé, On the Integration of Cloud Computing and Internet of Things, in: 2014 International Conference on Future Internet of Things and Cloud, IEEE, 2014, pp. 23–28. `doi:10.1109/FiCloud.2014.14`.

[10] I. Ion, N. Sachdeva, P. Kumaraguru, S. Capkun, Home is Safer than the Cloud! Privacy Concerns for Consumer Cloud Storage, in: Proceedings of the Seventh Symposium on Usable Privacy and Security, ACM, 2011, pp. 13:1–13:20. `doi:10.1145/2078827.2078845`.

[11] H. Takabi, J. Joshi, G. Ahn, Security and Privacy Challenges in Cloud Computing Environments, IEEE Security Privacy 8 (6) (2010) 24–31. `doi:10.1109/MSP.2010.186`.

[12] M. Henze, R. Hummen, K. Wehrle, The Cloud Needs Cross-Layer Data Handling Annotations, in: 2013 IEEE Security and Privacy Workshops (SPW), IEEE, 2013, pp. 18–22. doi:10.1109/SPW.2013.31.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A View of Cloud Computing, Commun. ACM 53 (4) (2010) 50–58. doi:10.1145/1721654.1721672.

[14] M. Memon, S. Rahr Wagner, C. Pedersen Fischer, F. H. Aysha Beevi, F. Overgaard Hansen, Ambient Assisted Living Healthcare Frameworks, Platforms, Standards, and Quality Attributes, Sensors 14 (2014) 4312–4341. doi:10.3390/s140304312.

[15] A. Queirós, A. Silva, J. Alvarelhao, N. P. Rocha, A. Teixeira, Usability, accessibility and ambient-assisted living: a systematic literature review, Universal Access in the Information Society 14 (1) (2015) 57–66. doi:10.1007/s10209-013-0328-x.

[16] S. Koch, Healthy Ageing Supported by Technology: A Cross-Disciplinary Research Challenge, Informatics for Health and Social Care 35 (3-4) (2010) 81–91. doi:10.3109/17538157.2010.528646.

[17] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of Things (IoT): A vision, architectural elements, and future directions, Future Generation Computer Systems 29 (7) (2013) 1645–1660. doi:10.1016/j.future.2013.01.010.

[18] M. Courtney, Premium binds, Engineering & Technology 8 (6) (2013) . doi:10.1049/et.2013.0611.

[19] D. Christin, A. Reinhardt, S. S. Kanhere, M. Hollick, A survey on privacy in mobile participatory sensing applications, Journal of Systems and Software 84 (11) (2011) 1928–1946. doi:10.1016/j.jss.2011.06.073.

[20] J. H. Ziegeldorf, N. Viol, M. Henze, K. Wehrle, POSTER: Privacy-preserving Indoor Localization, in: 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2014, pp. 1–2. doi:10.13140/2.1.2847.4886.

[21] J. Liu, X. Huang, J. K. Liu, Secure sharing of Personal Health Records in cloud computing: Ciphertext-Policy Attribute-Based Signcryption, Future Generation Computer Systems (2014) . doi:10.1016/j.future.2014.10.014.

[22] I. G. Smith (Ed.), The Internet of Things 2012 – New Horizons, IERC, 2012.

[23] M. Eggert, R. Häußling, D. Kerpen, K. Rüssmann, SensorCloud: Sociological Contextualization of an Innovative Cloud Platform, in: H. Krcmar, R. Reussner, B. Rumpe (Eds.), Trusted Cloud Computing, Springer, 2014, pp. 295–313. doi:10.1007/978-3-319-12718-7_18.

[24] M. Lesk, The Price of Privacy, IEEE Security Privacy 10 (5) (2012) 79–81. doi:10.1109/MSP.2012.133.

[25] R. Beckwith, Designing for Ubiquity: The Perception of Privacy, IEEE Pervasive Computing 2 (2) (2003) 40–46. doi:10.1109/MPRV.2003.1203752.

[26] S. Pearson, Taking Account of Privacy when Designing Cloud Computing Services, in: 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, IEEE, 2009, pp. 44–52. doi:10.1109/CLOUD.2009.5071532.

[27] S. Pearson, A. Benameur, Privacy, Security and Trust Issues Arising from Cloud Computing, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2010, pp. 693–702. doi:10.1109/CloudCom.2010.66.

[28] M. Zhou, R. Zhang, W. Xie, W. Qian, A. Zhou, Security and Privacy in Cloud Computing: A Survey, in: 2010 Sixth International Conference on Semantics Knowledge and Grid (SKG), IEEE, 2010, pp. 105–112. doi:10.1109/SKG.2010.19.

[29] J. Heiser, M. Nicolett, Assessing the Security Risks of Cloud Computing, Tech. Rep. G00157782, Gartner (2008).

[30] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation Computer Systems 28 (3) (2012) 583–592. doi:10.1016/j.future.2010.12.006.

[31] De Brauw Blackstone Westbroek N.V., EU Country Guide Data Location & Access Restriction (2013).

[32] J. Rosen, The Right to Be Forgotten, Stanford Law Review Online 64 (2012) 88–92.

[33] M. Henze, L. Hermerschmidt, D. Kerpen, R. Häußling, B. Rumpe, K. Wehrle, User-driven Privacy Enforcement for Cloud-based Services in the Internet of Things, in: 2014 International Conference on Future Internet of Things and Cloud, IEEE, 2014, pp. 191–196. doi:10.1109/FiCloud.2014.38.

[34] M. Glinz, S. Fricker, On shared understanding in software engineering: an essay, Computer Science - Research and Development (2014) . doi:10.1007/s00450-014-0256-x.

[35] C. Powers, P. Ashley, M. Schunter, Privacy Promises, Access Control, and Privacy Management – Enforcing Privacy Throughout an Enterprise By Extending Access Control, in: Third International Symposium on Electronic Commerce, IEEE, 2002, pp. 13–21. doi:10.1109/ISEC.2002.1166906.

[36] A. Lounis, A. Hadjidj, A. Bouabdallah, Y. Challal, Secure and Scalable Cloud-Based Architecture for e-Health Wireless Sensor Networks, in: 2012 21st International Conference on Computer Communications and Networks (ICCCN), IEEE, 2012, pp. 1–7. doi:10.1109/ICCCN.2012.6289252.

[37] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo, L. Alem, A platform for secure monitoring and sharing of generic health data in the Cloud, Future Generation Computer Systems 35 (2014) 102–113. doi:10.1016/j.future.2013.09.011.

[38] M. Li, S. Yu, Y. Zheng, K. Ren, W. Lou, Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption, IEEE Transactions on Parallel and Distributed Systems 24 (1) (2013) 131–143. doi:10.1109/TPDS.2012.97.

[39] M. Henze, S. Bereda, R. Hummen, K. Wehrle, SCSlib: Transparently Accessing Protected Sensor Data in the Cloud, in: The 6th International Symposium on Applications of Ad hoc and Sensor Networks (AASNET'14), Vol. 37 of Procedia Computer Science, Elsevier, 2014, pp. 370–375. doi:10.1016/j.procs.2014.08.055.

[40] B. Pooja, M. Manohara Pai, M. Radhika, A Dual Cloud Based Secure Environmental Parameter Monitoring System: A WSN Approach, in: 4th International Conference on Cloud Computing (CloudComp 2013), Springer, 2014, pp. 189–198. doi:10.1007/978-3-319-05506-0_18.

[41] A. D. Molina, M. Salajegheh, K. Fu, HICCUPS: Health Information Collaborative Collection Using Privacy and Security, in: ACM workshop on Security and privacy in medical and home-care systems (SPIMACS), ACM, 2009, pp. 21–30. doi:10.1145/1655084.1655089.

[42] J.-J. Yang, J.-Q. Li, Y. Niu, A hybrid solution for privacy preserving medical data sharing in the cloud environment, Future Generation Computer Systems 43-44 (2015) 74–86. doi:10.1016/j.future.2014.06.004.

[43] A. Jacobsson, M. Boldt, B. Carlsson, On the Risk Exposure of Smart Home Automation Systems, in: 2014 International Conference on Future Internet of Things and Cloud, IEEE, 2014, pp. 183–190. doi:10.1109/FiCloud.2014.37.

[44] T. Kirkham, D. Armstrong, K. Djemame, M. Jiang, Risk driven Smart Home resource management using cloud services, Future Generation Computer Systems 38 (2014) 13–22. doi:10.1016/j.future.2013.08.006.

[45] T.-M. Pasquier, J. Powles, Expressing and Enforcing Location Requirements in the Cloud Using Information Flow Control, in: 2015 IEEE International Conference on Cloud Engineering (IC2E), IEEE, 2015, pp. 410–415. doi:10.1109/IC2E.2015.71.

[46] W. Itani, A. Kayssi, A. Chehab, Privacy as a Service: Privacy-Aware Data Storage and Processing in Cloud Computing Architectures, in: Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), IEEE, 2009, pp. 711–716. doi:10.1109/DASC.2009.139.

[47] G. Karjoth, M. Schunter, A Privacy Policy Model for Enterprises, in: 15th IEEE Computer Security Foundations Workshop (CSFW), IEEE, 2002, pp. 271–281. doi:10.1109/CSFW.2002.1021821.

[48] Q. He, Privacy Enforcement with an Extended Role-Based Access Control Model, Tech. Rep. TR-2003-09, Department of Computer Science, North Carolina State University (2003).

[49] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, J. Karat, A. Trombeta, Privacy-Aware Role-Based Access Control, ACM Transactions on Information and System Security 13 (3) (2010) 24:1–24:31. doi:10.1145/1805974.1805980.

[50] J. Spillner, A. Schill, Flexible Data Distribution Policy Language and Gateway Architecture, in: 2012 IEEE Latin America Conference on Cloud Computing and Communications (LATINCLOUD), IEEE, 2012, pp. 1–6. doi:10.1109/LatinCloud.2012.6508149.

[51] C. Brodie, C.-M. Karat, J. Karat, J. Feng, Usable Security and Privacy: A Case Study of Developing Privacy Management Tools, in: Proceedings of the 2005 Symposium on Usable Privacy and Security, ACM, 2005, pp. 35–43. doi:10.1145/1073001.1073005.

[52] eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS Standard (2013).

[53] C. A. Brodie, C.-M. Karat, J. Karat, An Empirical Study of Natural Language Parsing of Privacy Policy Rules Using the SPARCLE Policy Workbench, in: Proceedings of the Second Symposium on Usable Privacy and Security, ACM, 2006, pp. 8–19. doi:10.1145/1143120.1143123.

[54] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, S. Völkel, Design Guidelines for Domain Specific Languages, in: Proceedings of the 9th OOPSLA Workshop on Domain-Specific Modeling, 2009, pp. 7–13.

[55] S. Trabelsi, J. Sendor, S. Reinicke, PPL: PrimeLife Privacy Policy Engine, in: 2011 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY), IEEE, 2011, pp. 184–185. doi:10.1109/POLICY.2011.24.

[56] P. Colombo, E. Ferrari, Towards a Modeling and Analysis Framework for Privacy-Aware Systems, in: 2012 International Conference on Privacy, Security, Risk and Trust (PASSAT) and 2012 International Confernece on Social Computing (SocialCom), IEEE, 2012, pp. 81–90. doi:10.1109/SocialCom-PASSAT.2012.12.

[57] D. Jutla, P. Bodorik, S. Ali, Engineering Privacy for Big Data Apps with the Unified Modeling Language, in: 2013 IEEE International Congress on Big Data (BigData Congress), IEEE, 2013, pp. 38–45. doi:10.1109/BigData.Congress.2013.15.

[58] C. Atkinson, T. Kuhne, Model-Driven Development: A Metamodeling Foundation, IEEE Software 20 (5) (2003) 36–41. doi:10.1109/MS.2003.1231149.

[59] B. Rumpe, Agile modeling with the UML, in: Radical Innovations of Software and Systems Engineering in the Future, Springer, 2004, pp. 297–309. doi:10.1007/978-3-540-24626-8_21.

[60] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, Addison-Wesley, 1999.

[61] B. Rumpe, Agile Modellierung mit UML: Codegenerierung, Testfälle, Refactoring, 2nd Edition, Springer, 2012. doi:10.1007/978-3-642-22430-0.

[62] B. Rumpe, Modellierung mit UML, 2nd Edition, Springer, 2011. doi:10.1007/978-3-642-22413-3.

[63] H. Grönniger, H. Krahn, B. Rumpe, M. Schindler, S. Völkel, MontiCore 1.0 – Ein Framework zur Erstellung und Verarbeitung domänenspezifischer Sprachen, Tech. Rep. 2006-04, Institute for Software Systems Engineering, Braunschweig University of Technology (2006).

[64] M. Eysholdt, H. Behrens, Xtext: Implement Your Language Faster Than the Quick and Dirty Way, in: Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion (OOPSLA), ACM, 2010, pp. 307–309. doi:10.1145/1869542.1869625.

[65] M. Schindler, Eine Werkzeuginfrastruktur zur agilen Entwicklung mit der UML/P, Ph.D. thesis, RWTH Aachen University (2012).

[66] A. van Deursen, P. Klint, J. Visser, Domain-specific Languages: An Annotated Bibliography, ACM SIGPLAN Notices 35 (6) (2000) 26–36. doi:10.1145/352029.352035.

[67] J. Jensen, M. G. Jaatun, Security in Model Driven Development: A Survey, in: 2011 Sixth International Conference on Availability, Reliability and Security (ARES), IEEE, 2011, pp. 704–709. doi:10.1109/ARES.2011.110.

[68] T. Lodderstedt, D. Basin, J. Doser, SecureUML: A UML-Based Modeling Language for Model-Driven Security, in: UML 2002 – The Unified Modeling Language, Springer, 2002, pp. 426–441. doi:10.1007/3-540-45800-X_33.

[69] D. Basin, M. Clavel, M. Egea, A Decade of Model-Driven Security, in: Proceedings of the 16th ACM Symposium on Access Control Models and Technologies (SACMAT), ACM, 2011, pp. 1–10. doi:10.1145/1998441.1998443.

[70] D. Ma, G. Tsudik, A New Approach to Secure Logging, ACM Transactions on Storage 5 (1) (2009) 2:1–2:21. doi:10.1145/1502777.1502779.

[71] Freemarker project, Freemarker (2015).
URL http://freemarker.org/

[72] P. Dourish, K. Anderson, Collective Information Practice: Emploring Privacy and Security as Social and Cultural Phenomena, Human-Computer Interaction 21 (3) (2006) 319–342. doi:10.1207/s15327051hci2103_2.

[73] H. Nissenbaum, A Contextual Approach to Privacy Online, Daedalus 140 (4) (2011) 32–48. doi:10.1162/DAED_a_00113.

[74] G. T. Marx, G. W. Muschert, Personal Information, Borders, and the New Surveillance Studies, Annual Review of Law and Social Science 3 (2007) 375–395. doi:10.1146/annurev.lawsocsci.3.081806.112824.

[75] M. Henze, M. Großfengels, M. Koprowski, K. Wehrle, Towards Data Handling Requirements-aware Cloud Computing, in: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, 2013, pp. 266–269. doi:10.1109/CloudCom.2013.145.

[76] R. M. Needham, Denial of Service, in: Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS), ACM, 1993, pp. 151–153. doi:10.1145/168588.168607.

[77] D. Dean, A. Stubblefield, Using Client Puzzles to Protect TLS, in: Proceedings of the 10th USENIX Security Symposium, USENIX, 2001, pp. 1–8.

[78] J. Mirkovic, P. Reiher, A Taxonomy of DDoS Attack and DDoS Defense Mechanisms, SIGCOMM Comput. Commun. Rev. 34 (2) (2004) 39–53. doi:10.1145/997150.997156.