Institute of Parallel and Distributed Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Privacy for the MBP IoT Platform

Jakob Benz

Course of Study:	Softwaretechnik
Examiner:	PD Dr. rer. nat. Holger Schwarz
Supervisor:	Dr. rer. nat. Pascal Hirmer
Commenced:	April 15, 2020
Completed:	October 31, 2020

Abstract

The popularity of the Internet of Things is rapidly increasing and so is the range of IoT application scenarios. The prominent examples are Smart Homes, Industry 4.0 and Car2X. Undoubtedly, there are huge benefits for end-users in these scenarios. However, due to the pervasiveness of the Internet of Things, the amount of private and sensitive data captured in IoT environments is of great magnitude. A central IoT platform is often used to manage this data. That is, privacy concerns arise which have to be dealt with, e.g., by the vendors of IoT platforms.
To address the privacy related issues in IoT environments, we propose a concept for privacy and access control particularly tailored for IoT environments. In a first step, an exhaustive evaluation of related work in the field of access control is conducted. We then present a generic end-to-end reference architecture for IoT environments. Based on this reference architecture, we examine the special requirements towards privacy and access control in IoT environments.
Following that, we present a generic concept for IoT environments, that allows different ways of integration, provides flexible yet expressive specification of privacy settings, is highly customizable, and introduces minimal overhead for access control enforcement. The concept provides details about the integration into existing IoT environments, the access control workflow and components as well as the suggested domain model and evaluation algorithm. A concrete implementation is illustrated with the Multi-purpose Binding and Provisioning Platform (MBP) developed at the University of Stuttgart.

Kurzfassung

Das Internet of Things wird immer beliebter und ist immer weiter verbreitet. Die bekanntesten Anwendungsszenarien umfassen Smart Homes, Industrie 4.0 und Car2X. Ohne Zweifel entstehen dadurch viele Vorteile für Endanwender. Da diese Anwendungsfälle aber mit der Erhebung und Speicherung großer Mengen an privaten Daten einhergehen, welche oftmals in zentralen IoT Plattformen gespeichert werden, müssen z.B. Hersteller solcher Plattformen besondere Aufmerksamkeit auf den Datenschutz legen.
In dieser Arbeit wird ein Konzept vorgestellt, das benutzt werden kann, um Datenschutz und Zugangskontrolle in IoT-Umgebungen umzusetzen. In einem ersten Schritt werden daher bereits existierende Ansätze und Arbeiten im Bereich der Zugangskontrolle evaluiert. Eine generische Ende-zu-Ende Referenzarchitektur für IoT-Umgebungen wird vorgestellt, basierend auf welcher die speziellen Anforderungen an ein Zugangskontrollsystem für IoT-Umgebungen erhoben werden.
Als Hauptbestandteil dieser Arbeit wird das generische Konzept für Zugangskontrolle in IoT-Umgebungen vorgestellt. Dieses Konzept erlaubt verschiedene Arten der Integration, flexible und sehr ausdrucksstarke Spezifikation von Datenschutz-Einstellungen, ist hochgradig anpassbar und erweiterbar, und unterstützt eine effiziente Umsetzung der Zugangskontrolle. Das Konzept beinhaltet Details zur Integration, zum Ablauf der Zugangskontrolle und den involvierten Komponenten, sowie ein Domänenmodell und einen Algorithmus zur Evaluierung von Zugangsanfragen. Eine konkrete Implementierung wird für die an der Universität Stuttgart entwickelte Multi-purpose Binding and Provisioning Platform (MBP) vorgestellt.

Contents

List of Figures

List of Tables

Listings

List of Algorithms

1 Introduction

Imagine your coffee machine, washing machine, and car have their own social media accounts and you can communicate or even talk to them like it was the most normal thing on earth? This is becoming reality with the Internet of Things (IoT), bridging the gap between the real and the digital world [OMEO17]. Nowadays, more and more objects in our everyday life are fitted with sensors or even actuators, turning them into smart devices. Fueled by recent power-efficient and low-cost communication technologies, these smart devices can be interconnected very easily [SM19]. There are a lot of already well-established application scenarios for the IoT, commonly referred to as SMART environments, e.g., SMART Homes [BLM+11]. Due to sensor technology becoming more sophisticated, versatile, and affordable, the number of smart devices with sensors and actuators is increasing significantly. However, the complexity of (initially) setting up an IoT environment with a large number of heterogeneous devices increases as well. The Multi-Purpose Binding and Provisioning Platform (MBP)[1] has been developed at the University of Stuttgart to facilitate creating, modeling, and deploying IoT environments and to automate the process of connecting devices since doing so manually can be very cumbersome and costly.

Once an IoT environment has been set up, the devices and respective sensors capture a wide range of data which requires quite a bit of data storage. Due to the resource-constrained and short-range-communication nature of such devices the data is hardly ever stored on the device itself, but rather outsourced to the cloud [ZCDV17] through a layer of communication middleware. Considering that most of the data collected in IoT environments should be labeled personal or sensitive – especially revisiting the SMART Home scenario and also enterprise-level use cases, where data may be confidential or even classified – security and privacy are inevitable. Unfortunately, both security and privacy are often treated negligently when it comes to IoT application scenarios [OMA15]. Although there is a wide variety of approaches to security and privacy, these are hardly ever applicable in the IoT context, due to its dynamic and ever-changing nature, completely different communication stacks, and scalability requirements necessitated by the immense amount of devices and sensors [SRGC15]. In addition to these architectural challenges, the great variety of opportunities that arise with the IoT at the same time calls for security and privacy measures that are adaptable and customizable to an extremely high degree while keeping the required expertise to a minimum.

Although there has been a lot of work done in the field of security and privacy in the context of a modern computing environment, including but not limited to the IoT, existing approaches do not (completely) meet the special requirements of IoT environments. Therefore, the main focus of this work is on the development of a generic concept for privacy and access control in IoT environments.

[1] MBP GitHub Repository: https://github.com/IPVS-AS/MBP

As a first step, the basics related to the Internet of Things and privacy are briefly explained in Chapter 2. Then, an extensive survey of existing approaches towards privacy and access control, especially focusing on IoT platforms like the MBP, is discussed along with an assessment of their suitability in Chapter 3. Based on a generic end-to-end reference architecture for IoT environments, Chapter 4 contains the result of the requirements elicitation. Chapter 5 presents the core of this work, the concept for privacy and access control in IoT environements. The prototypical implemenation for the MBP is shown in Chapter 6 before evaluating the concept and discussing its limitations in Chapter 7. Finally, Chapter 8 provides a conclusion and an outlook on future work.

2 Basics

This section provides some fundamental information relevant in the context of this work. An introduction to the Internet of Things is given in Section 2.1, including IoT platforms, a corresponding reference architecture, and a brief presentation of the MBP. Section 2.2 introduces and distinguishes the terms privacy and security in the context of this work.

2.1 The Internet of Things

The essential idea behind the Internet of Things (IoT) is to connect heterogeneous, physical devices using the internet, creating a pervasive presence of Things [AIM10] in our everyday life, which is also referred to as "stringent connectedness" [Ray18] between the physical world and its digital twin. These Things can be Radio-Frequency Identification (RFID) tags, sensors and actuators, or also existing devices fitted with them, e.g., smartphones, household devices, cars, etc. The goal on the one hand is to enable users to communicate with these Things, on the other hand to enable the communication between them, paving the way for well-advanced application scenarios, which unquestionably will have a big impact on the way we experience our daily life. Additionally, the extent and scale of these use cases is virtually unlimited, e.g., Industry 4.0 and smart factories are already in place, there is even the term Factory of Things [Zue10].

Following up on the larger scale IoT application scenarios mentioned in the previous section, these especially demand for the corresponding infrastructure and comprehensive administration tools to manage the millions of devices, the connection and communication between them, as well as the data captured by sensors of these devices. This is exactly what IoT platforms have been developed for. With the IoT gaining more and more attention, the number of already available IoT platform solutions increases alike. Although the offered functionality is often similar [GBF+16], due to the heterogeneity of the Things and many different (communication) standards, the technology stacks used in available IoT platforms are very heterogeneous themselves. The range of state-of-the-art IoT platforms includes both open-source solutions, such as ThingsBoard[1], FIWARE[2], and of course the MBP, as well as proprietary ones, e.g., Amazon Web Services IoT[3] [GBF+16]. When developing an IoT platform or frameworks for or on top of one, such as the privacy concept developed in this work, the downside of the heterogeneity among existing IoT platforms is noticeable, as there is no commonly accepted standard for IoT platform architectures. Guth et al. propose an abstract and generic reference architecture for IoT platforms [GBF+16] which is depicted in Figure 2.1. It comprises three layers: the device layer, the middleware layer, and the application layer. On the bottom most level there are the devices, to which sensors and actuators can be connected using

[1] Thingsboard Homepage: https://thingsboard.io
[2] FIWARE Homepage: https://www.fiware.org
[3] Amazon Web Services IoT Homepage: https://aws.amazon.com/en/iot/

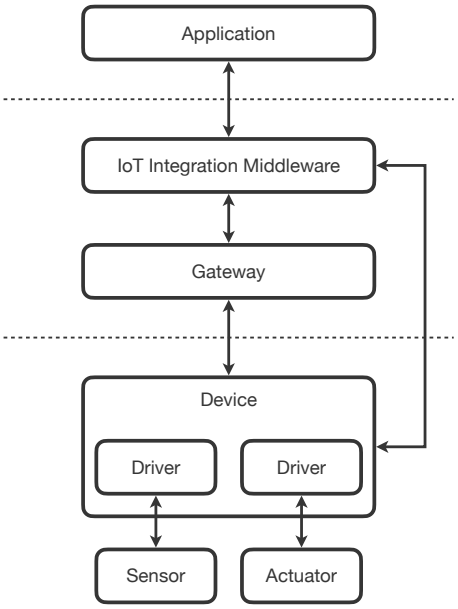

Figure 2.1: IoT Platform Reference Architecture according to [GBF+16]

respective drivers. Sensors measure environmental information, e.g., the current room temperature or the location of a smartphone. Actuators on the other hand are able to manipulate their environment by performing, e.g., mechanical, optical, or acoustical actions.
In order to access the data gathered by sensors and to send commands to actuators, the integration middleware layer provides the means that the application layer needs to communicate with the devices. Direct communication between the integration middleware and a device is possible if the device is compatible in terms of the transport protocol, such as HTTP or MQTT, and data format, such as JSON, or XML. In case a device is not capable of directly communicating with the integration middleware, a gateway can be used as an adapter, which transforms and forwards the data exchanged between devices and the integration middleware. The functionality of the integration middleware is not limited to the above mentioned use cases, but can also encompass data (pre-) processing logic or even a rules engine [GBF+16].
The application layer consists of software artifacts that implement IoT application scenarios using sensors and actuators. Furthermore, it can provide insights by aggregating and processing various sensor data which are then presented to users through different kinds of dashboards. For administrative purposes, the application layer usually provides a user interface for managing devices, sensors, and actuators as well as the way they're interconnected.

We use this reference architecture throughout this work, since the abstract design enables it to be used as the architectural basis for the concept for privacy and access control in IoT environments, which in turn increases the concept's degree of reusability.

2.1.1 The Multi-purpose Binding and Provisioning Platform (MBP)

The Multi-purpose Binding and Provisioning Platform (MBP) has been developed at the University of Stuttgart as a working prototype for an open-source IoT platform [FHS+20]. The primary target is to ease the management of IoT environments along all lifecycle phases. At first, a new IoT

environment has to be designed and modelled, which is faciliated by the MBPs own environment modeling tool (cf. Figure 2.2). Once a new IoT environment has been modelled, it has to be deployed, i.e., the application layer (cf. Figure 2.1) needs to be enabled to access the sensors and actuators via the respective devices, which is called binding [HWBM16b]. This can be done using either the MBP's web frontend or the RESTful API, since both of them offer the functionality to register and manage devices, sensors, and actuators [HBS+16]. The MBP employs a set of small scripts to control the devices, i.e., to start or stop them, and to extract sensor data and control actuators. These adapters scripts don't have to be device-specific and can be implemented generically, allowing to re-use them among multiple devices [HWBM16a] . Several ready-to-use adapter scripts can be found in the MBP Operators Repository[4], nevertheless it is possible to add custom adapter scripts. The middleware layer is implemented using a Mosquitto MQTT Broker[5], to which the adapter scripts publish the extracted sensor data [FHS+20]. The rule modeling tool enables users to create automated workflows in an MBP-managed IoT environment by defining rules using the event-condition-action pattern, where events are based on Complex Event Processing (CEP) [CM12] queries on sensor data and actions specified for actuators [FHPM18]. That way, also non-expert users can, e.g., easily add some automation to their homes.

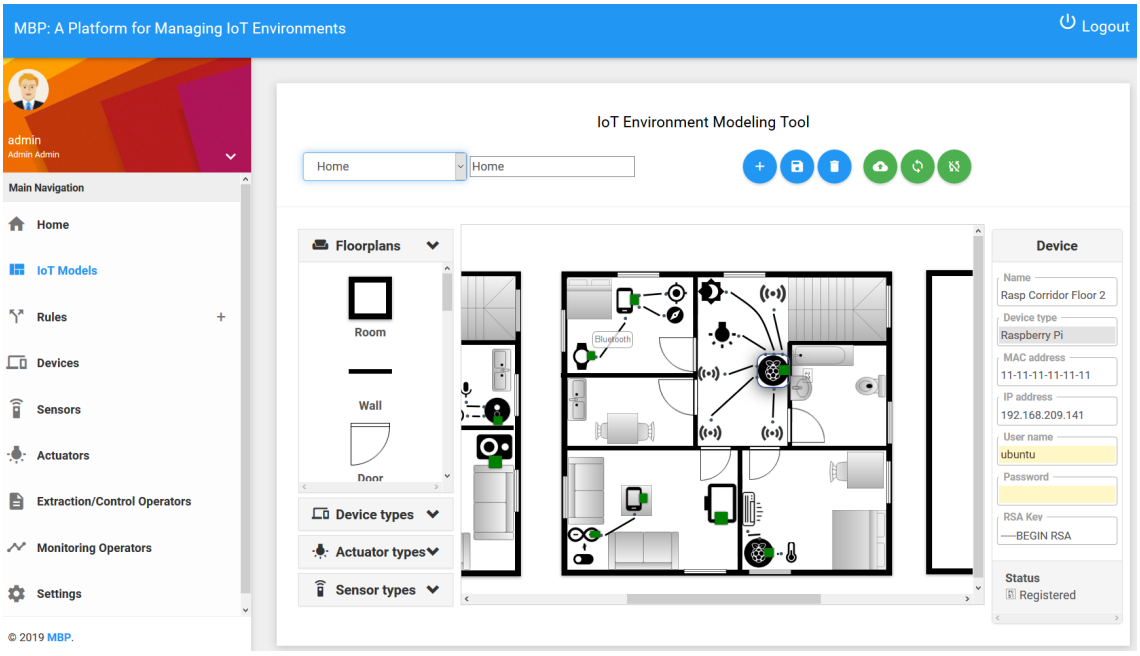

Figure 2.2: The MBP IoT Graphical Environment Modeling Tool [FHS+20]

[4]MBP Operators Repository: https://github.com/IPVS-AS/MBP/tree/master/resources/operators
[5]Eclipse Mosquitto Homepage: https://mosquitto.org

The technology stack of the MBP further comprises two databases, a MongoDB[6] for device (meta) data and user data, and an InfluxDB[7] as a time series database for sensor data, allowing the presentation of both current and historical data in the web frontend. The backend is hosted on an Apache Tomcat application server[8]. Finally, an Android smartphone app is available as an alternative frontend.

2.2 Security & Privacy

As sensors are becoming increasingly ubiquitous and sophisticated [DTS+16], awareness of privacy and security risks among users increases likewise [ZACF18]. Undoubtedly, privacy and security are of vital importance. In the following, the terms security and privacy are briefly explained.

Security in the context of information technology is built on three main aspects, namely confidentiality, integrity, and availability, which together form the so-called CIA-Triangle [WM11]. While confidentiality measures protect secure entities from unauthorized access in general, integrity measures are designed to prevent the unauthorized manipulation of said entities, e.g., the unauthorized update of database records, to ensure that reliable and correct data can be provided to authorized users. Countermeasures to protect confidentialiy and integriy include, among others, authentication, access-control lists, and data encryption. The purpose behind the third aspect, availability, is to guarantee that authorized users actually are able to access the entities they have been granted access for [MYAZ15].

While security is focused more on protecting personal data against unwanted access in general, such as malicious attacks, privacy relates to the right of users to control the extent to which they want to share their personal data, e.g., when and with whom to share their camera roll. With modern computing paradigms such as Cloud Computing and the Internet of Things, where privacy is reduced due to the omnipresence of intelligent devices [SNCC18], sophisticated privacy-preserving techniques are inalienable.

Since the data captured in IoT environments is mostly privacy-sensitive, they become attractive targets for attacks. In order to preservce the privacy of (IoT) users and also to protect their data from unwanted access, a widely used mechanism are access control systems. These systems manage the process of determining whether some requesting entity, e.g., a user or an application, should be granted access to some resource. Thereby, the decision-making process might be based on certain conditions, that have to hold in order to grant access. Moreover, the access control system could decrease the level of detail of the data that is presented to the requesting entity, e.g., by blurring images or reducing the accuracy of a location. For instance, hospital staff needs to have access to personal information of patients for administrative purposes, but should not be able to access the patients medical records. On another note, the data recorded in smart buildings might result in the disclosure of information of people and their habits that they don't want to share, such as when and where they went and what they did [PDY+17]. Going beyond the domain of personal and home IoT

[6] MongoDB Hompage: https://www.mongodb.com
[7] InfluxDB Homepage: https://www.influxdata.com
[8] Apache Tomcat Homepage: https://tomcat.apache.org/download-80.cgi

application scenarios towards enterprise and government use cases, legal issues associated with privacy and access control have to be considered, such as legal requirements regarding the logging of production steps [SWW15].

While the idea behind and the purpose of different access control models is more or less the same, the concept and implementation differ substantially. The next chapter presents various categories of access control models and related work within these categories, respectively.

3 Related Work

Many privacy-preserving approaches, models, frameworks, and mechanisms have been proposed in the context of the IoT. This chapter presents the most relevant work grouped by the following categories:

- Role-based access control models
- Attribute-based access control models
- UCON-based access control models
- Capability-based access control models
- Other models

The concepts of each category is briefly explained before a set of representative approaches is presented. Figure 3.1 shows an overview of the work done in the aforementioned categories of access control.

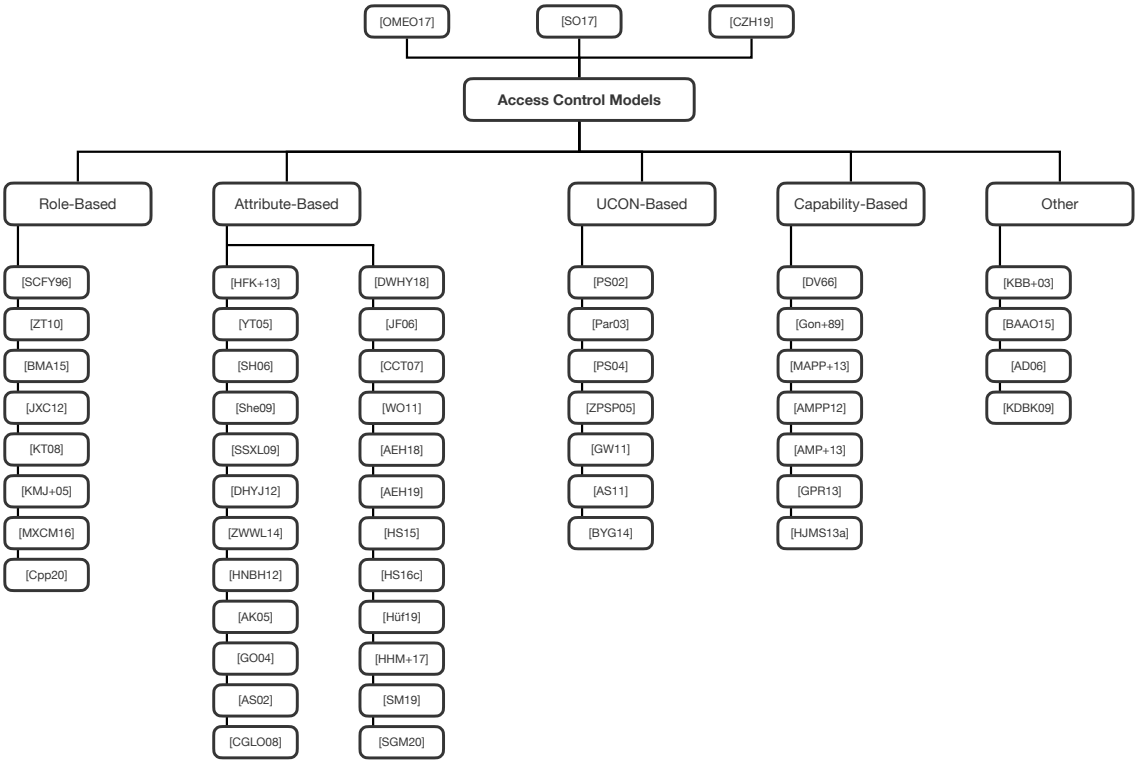

Figure 3.1: Existing access control models

3.1 Role-Based Access Control (RBAC) Models

The role-based access control model (RBAC) is the most widely accepted access control method [TJA10]. The core concept of RBAC uses roles with a set of associated permissions. These roles are then assigned to users, which are able to use the permissions associated with the respective roles [FSG+01]. Thereby, system administrators can easily manage users and their respective permissions, since roles can be predefined and reused, e.g., when creating new users [CW13]. Core RBAC can be extended by introducing role hierarchies allowing roles to extend other roles, and constraints, allowing for separation of duties, which guarantees that one individual cannot be assigned mutually exclusive roles, e.g., *purchasing manager* and *accounts payable manager* [SCFY96]. However, traditional RBAC is not able to cope with the high dynamic, heterogeneity, and scalability requirements of the IoT [OMEO17]. A major problem with RBAC in IoT environments is role-permission explosion due to the specification of permissions in terms of object identifiers, i.e., individual objects [RJK15]. In the following, several approaches that adopt traditional RBAC towards access control solutions for the IoT are presented.

Zhang and Tian [ZT10] propose a service-based approach to facilitate integration and management of physical objects, i.e., the functionality of IoT devices can be accessed through a set of standard services. They extend RBAC by introducing contextual constraints to allow the consideration of time, location, and other environmental variables in the evaluation of an access request. Such access requests are targeted at services offered by IoT devices. Prior to invoking a service, authorization of the requesting user is verified. Although the service-based approach greatly benefits the integration of heterogeneous IoT devices, this approach has been critized (e.g., by Ouaddah et al. [OMEO17]) for a number of shortcomings, including role-permission explosion, the fact that roles are only assigned to users excluding other devices and services, and missing description of the way the contextual information is captured from the devices' environments. Additionally, their approach has only been evaluated in a case study, omitting a prototypical implementation and implementation details.

Barka et al. [BMA15] present another approach based on RBAC that is focused on the Web of Things (WoT). Although they establish a careful mapping between the entities in RBAC and the WoT, such as *WoT-participant/RBAC-subject* and *WoT-Thing/RBAC-object*, the set of mappings does not include any information on how roles are mapped to the WoT entities and components. The proposed architecture is based on the concept of a modified version of the ISO reference monitor [ISO96] [MASM13] and employs two facilities: an *Access Control Enforcement Facility (AEF)* and an *Access Decision Facility (ADF)*. The AEF intercepts every request to access an object and forwards it to the ADF, which evaluates the request using the role-based policies and returns the decision to the AEF. This decision is then enforced at the AEF. Due to the completely centralized access control architecture, the WoT Things (abstractions of physical or virtual entities in the WoT [W3C20]) do not have any impact on the decision-making process. The problem of role proliferation is addressed by using role parameterization [FKC03]. Even though RBAC is capable of incorporating constraints, the authors do not describe if and how constraints are included in their approach. Furthermore, the decision whether access is granted or not, does not comprise any contextual information. Additionally, access is either granted fully or not at all, without allowing constraint access to resources.

Another approach targeted at the WoT and based on RBAC is presented by Jindou et al. [JXC12]. They propose an extension to Sandhu's standard RBAC model [SCFY96], named SARBAC, integrating Social Network Services (SNS), e.g., Facebook, and attributes, mainly focusing on the usability aspect of access control in the WoT. Thereby, a provider, which acts as the device owner, is able to configure user-role assignments based on the data obtained from user profiles (e.g., age, gender, occupation) and relationships between several user profiles, called social links (e.g., family, colleague). Likewise, role-permission assignments and policy definitions can be done using device attributes, e.g., a set of tags or the location. However, device-to-device communication is not addressed. Also, this approach is tightly coupled and highly dependent on the respective SNS provider, which makes the SNS provider an implicit trusted third party [OMEO17].

Kulkarni et al. [KT08] dynamically integrate context into RBAC by adding a context management layer on top of the access control layer, creating the Context-Aware RBAC Model (CA-RBAC). They aim at securing user data while simultaenously enabling third parties to access user data in case of, e.g., an emergency. This is achieved by establishing a modular context structure which is divided into three context conditions: normal, critical, and emergency. Although this structure is very easy to understand and user-friendly, it is very limited due to the non-generic and pre-defined context conditions.

Kim et al. introduce CWAC [KMJ+05], which employs a state checking matrix (SCM), a state checking agent, and a context-aware agent to monitor the context of users and dynamically adjust the role of the user, respectively. However, only user-related contextual information is incorporated into the access decision-making process. A major issue with both CA-RBAC and CWAC is the limited scalability and inefficiency, that can be caused by role-proliferation and/or improper configuration. In addition to that, the previously discussed disadvantages can be identified as well.

Several other role-based access control approaches have been proposed, such as Break-the-Glass Access Control (BTG-AC) by [MXCM16] [MXCM14] based on the Break-the-Glass Role-Based Access Control BTG-RBAC) model by Ferreira et al. [FCF+09], which offers the possibility to execute break-the-glass actions (e.g., override access policy) to authorized users, or the framework proposed by Chatterjee et al. [CPP20] which enables on-chain access control based on smart contracts in the context of Distributed Ledger Technology (DLT) with a clear separation of business logic and access control logic. However, these and other role-based approaches focus on different target environments.

3.2 Attribute-Based Access Control (ABAC) Models

Although the role-based approach is well suited for use cases where the set of permissions and the set of roles associated with users change infrequently, environments like the IoT demand for highly dynamic and flexible access control schemes. A widely used method is attribute-based access control (ABAC). We use the definition of Hu et al. [HFK+13]:

> *A logical access control methodology where authorization to perform a set of operations is determined by evaluating attributes associated with the subject, object, requested operations, and, in some cases, environment conditions against policy, rules, or relationships that describe the allowable operations for a given set of attributes.*

Using attributes, a more comprehensive and detailed input into the decision-making process can be achieved while simulateneously eliminating the need to define explicit user-permission mappings due to the attribute-based policies that can be applied generically to every requesting entity [AEH18]. However, challenges arise with large sets of policies or also very complex policies, which is very likely in enterprise IT landscapes [HKFV15], since evaluating (many) complex policies can be very expensive in terms of computing power. The eXtensible Access Control Markup Language (XACML) [Sta05] provides a comprehensive, but complicated policy specification and evaluation mechanism with quite a bit of performance overhead [KAPI18]. Considering the rather resource-constrained nature of the IoT, an efficient, yet expressive and easy-to-understand policy specification and evaluation technique is desired. In the following, several attribute-based access control models are discussed.

An early attribute-based approach to access control for web services has been proposed by Yuan and Tongin [YT05]. They evaluate challenges and requirements regarding access control and propose a new ABAC approach based on attributes of subjects, objects, and the environment. Their model includes an authorization architecture, which is based on Attribute Authorities (AA) creating and managing the attributes of subjects, a Policy Enforcement Point (PEP) requesting access decisions and enforcing them, a Policy Decision Point (PDP) evaluating policies and making the access decision requested by the PEP, and a Policy Authority (PA) managing the access control policies. These policies are defined as Boolean functions but evaluated using XACML. A sample policy (rule) is shown in Listing 3.2, where a subject may access the resource with the name `ApprovePurchase`, if it has the value `Manager` for the attribute `Role`.

```
1  R: can_access(s, r, e) ←
2       (Roles(s) = ''Manager'') ∧
3       (Name(r) = ''ApprovePurchase'') ∧
```

Listing 3.1: Example policy according to [YT05]

However, the authors only give an overview of the relation between the attributes and policies and focus more on policy fomulation without providing any implementation details with regard to policy evaluation. Furthermore, the decision results only encompass fully granting or fully denying access.

Similar to the previous approach, authors in [SH06] propose WS-ABAC, an attribute-based model for web services. They defined policies as a triple `<S, srv, C>`, where `S` identifies the subjects the policy includes, `srv` is the respective web service, and `C` is the attribute constraint, which is a set of attribute conditions of the form `<attribute_name><operator><value>`. A sample attribute constraint according to the WS-ABAC model is shown in Listing 3.2, where access to the web service is limited to a manager, that requests the service from his/her office between 9am and 5pm.

```
1  C: Identity = ''manager'' ∧ Time ≥ 09:00 ∧ Time ≤ 17:00 ∧ Location = ''office''
```

Listing 3.2: Example according to [SH06]

The attribute constraints can be combined using the logical AND or OR operator. However, the set of allowed operators inside the constraint is limited to $<, >, \leq, \geq, =, \neq$. An access request has the form `<U, srv, RA>`, where `U` identifies the requesting user, `srv` identifies the requested web service, and `RA` contains the runtime attributes, which are used to evaluate the attribute constraints of the matching policies. They present an XACML-based authorization architecture similar to the one by Yuan and Tong [YT05], and in addition to that a trust negotiation architecture, which allows

regulation over when and how secure information, such as user credentials, is transferred from the requestor to the service provider, e.g., if further proof is required. However, their approach has been critized (e.g., by Servos and Osborn [SO17]) for being minimalistic and the lack of a complete and foundational attribute-based model for access control. According to Servos and Osborn [SO17], several approaches related to the previous two models have been proposed [DHYJ12] [SSXL09] [She09] [ZWWL14], all of which basically have the same shortcomings.

Several hybrid models that combine role-based elements with attribute-based access control have been proposed [SO17]. The most basic solution for the combining the two models, is to use roles as simply another attribute. Huang et al. [HNBH12] propose a two-layer-architecture with a role-based frontend and a backend, which controls access based on attribute-based policies. This separation enables different levels of access control managament. Daily reviews of role assignments can be done using the frontend while rather complex administration operations, such as defining policies, can be performed in the backend. However, this advantage only maintains as long as all of the attribute-based policies can be mapped to the purely role-based model of the frontend, i.e., identity-less policies will cause issues [SO17]. The concept of parameterized roles has been addressed by Abdallah and Khayat [AK05], as they simply add a condition that is evaluated before a certain permission is granted. In contrast to that, Ge and Osborn [GO04] incorporate the attributes of subjects as well as the content of objects through locigal expressions similar to XPath [CD+99], which are evaluated at runtime based on subject attributes obtained from sessions. Another way to combine RBAC and ABAC is to assign roles based on attributes. Al-Kahtani and Sandhu [AS02] propose Rule-Based RBAC (RB-RBAC) which dynamically assigns roles to users at runtime based on certain attribute expressions, whereas attributes of objects are not considered. A similar model is presented in [CGLO08], where roles are assigned likewise, but with the additional option to define constraints for roles.

Moreover, Semantic Web Technologies [HKR09] have been used in several attribute-based approaches. Dong et al. [DWHY18] propose the Context-States-Aware Access Control (CSAAC) Model which combines ABAC with both role-based elements as well as context-states-awareness. They provide an abstract logical architecture, which extends the basic XACML reference architecture [Sta05] with state-tracking support. Policies are defined using the Sematic Web Rule Language[1] (SWRL). However, even the authors themselves state that implementing an access control system using Sematic Web Technology such as SWRL, is very difficult and extremely complex, which probably is the reason why they did not provide any implementation details or prototypical implemention.
Servos and Osborn [SO17] provide a list of other approaches that also use Semantic Web Technologies, including [JF06], [CCT07], and [WO11]. However, all of these models aim at assigning roles through the evaluation of attributes, which is not the goal of this work. Furthermore, none of the mentioned models uses attributes of objects and the environment, besides the approach of Jin and Fang-chun [JF06], which only encompasses temporal environment information [SO17].

A different approach is based on automatic policy specification [AEH18]. This automatic policy specification mechanism uses two types of contexts, a *guard context*, which is defined by an administrator or device owner and can be seen as a set of preconditions (application dependent variables and thresholds), that have to be met in order to access the respective resource, and an *operational context*, which contains real-time measurements of attributes of the subject, object, and

[1] W3C SWRL Proposal: https://www.w3.org/Submission/SWRL/

resource/operation. These contexts are defined using a set of predicates, so-called *primitive facts*, which basically are tuples of key-value pairs. However, so-called resource profiles, desribing the respective attribute domains through a range of possible values, have to be defined first. Access control policies are configured through a user-friendly web interface, and afterwards converted into *primitive facts*, which represents the notion of automatic policy specifciation, although the administrator is required to approve the automatically generated policy. Flexibility, maintainability, and reusability are constrainted due to the non-generic nature of resource profiles and the fact that access decision results are limited to full access or no access at all.

The Continuous Access Policy Enforcement (CAPE) framework also proposed by Alkhresheh et al. [AEH19] addresses the aspect of continuity in access control systems, which is necessary since the context of entities in IoT environments changes frequently. Automatic policy specification [AEH18] and continuous policy update and enforcement is provided using two main components: a *Session Registry* (SR) maintaining information regarding active (access) sessions, and a *Context Monitor* (CM) continuously observing all *operational contexts* and checking for policy updates, i.e., changed *guard contexts*. If necessary, access decisions are re-evaluted, e.g., because a subject's *operational context* does not match a corresponding *guard context* anymore. Despite the optimized adaptability of access control policies in highly dynamic IoT environments due to continuous re-definition of primitive facts and re-evaluation of access decisions at runtime, it has to be mentioned, that this approach produces performance overhead. The authors present a short performance evaluation of the access response time and the re-enforcement time. However, they state that CAPE is suitable for large-scale IoT environments, while limiting the number of attributes to only 13 and the number of *primitive facts* to only 1000 when investigating the performance of re-enforcement operations. Additionally, implementation details or a prototypical implementation are not provided.

Hüffmeyer et al. propose RestACL [HS15] [HS16c] [Hüf19], an access control language for RESTful services, which provides a resource-oriented high-level architecture and policy specification model for integrating access control into RESTful environments. The proposed architecture is shown in Figure 3.2. A RESTful server receives requests from RESTful clients and forwards requests to

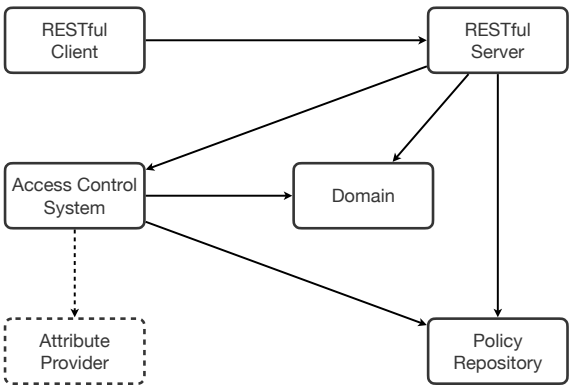

Figure 3.2: RestACL architecture according to [HS16c]

secured resources to the Access Control System, which contains the decision-making logic. For computing an access decision, it uses the domain (model), which provides a mapping between resources and policies, the Policy Repository holding all policies, and an optional Attribute Provider that can provide additional attributes that cannot be obtained from the request itself. RestACL policies comprise an identifier, a priority in case more than one policy is defined for a resource,

a (composite) condition, which holds a function to compare arguments, and an effect indicating the action to perform, e.g., grant access, if the condition holds. Furthermore, this model allows parameterized access, i.e., access to only parts of resources, such as filtered lists. In addition, it is possible to use URI templates when defining the set of policies for a (sub-) resource, which facilitates the administration of resource-policy mappings. A formal comparison with XACML is presented by Hüffmeyer and Schreier in [HS16b] and [HS16a]. The results indicate significant performance advantages regarding processing time and memory consumption compared to already optimized versions of XACML. The only downside of RestACL is the lack of partial or gradual access to resources. However, this can be achieved by optimizing the expressiveness of the effect parameter in policies.

SitAC, a system for situation-aware access control, is proposed by Hüffmeyer et al. [HHM+17]. This system is built on top of RestACL and the situation recognition system SitOPT [WSBL15] [HWS+15] [SHWM16]. Access decisions are based on attribute-based policies and real-time situation recognition using sensor values. However, the authors only provide a high-level architecture and describe concepts but no (prototypical) implementation. Furthermore, devices and sensors are managed by the SitOPT Resource Management Platform (RMP), which requires manual device, sensor, and service registration and administration, which can be cumbersone in large-scale IoT environments.

Stach and Mitschang introduce ACCESSORS [SM19], a data-centric approach to modeling permissions in IoT privacy systems. They reason that humans think in a data-centric way, which is why existing permission models are not suitable for the IoT. Therefore, data abstraction is used to eliminate the need to define a separate rule for each data-source, e.g., a particular sensor, if a user wants to protect a certain type of personal data, possibly originating from multiple data sources. Figure 3.3 illustrates the basic structure of an ACCESSORS permission. Besides the data

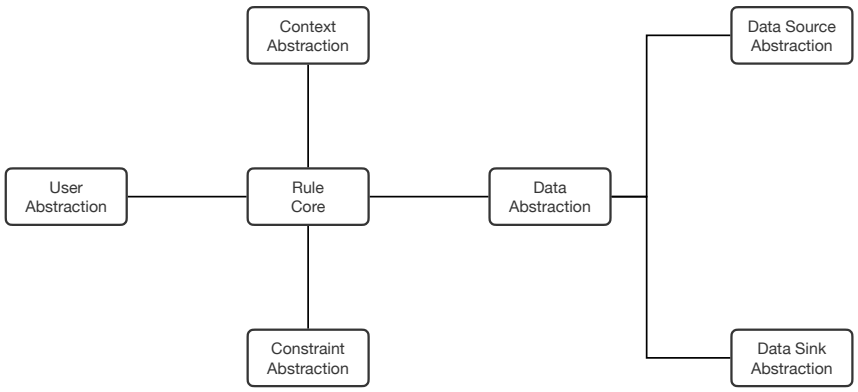

Figure 3.3: Abstract permission model according to [SM19]

abstraction element, ACCESSORS provides abstractions for users, contexts, and constraints. The user abstraction can be used to distinguish between different requesting entities, such as applications, devices, or users. The context abstraction enables the selective activation of rules, e.g., only if a certain situation occurs, while the constraint abstraction provides means to grant or deny access to data on the one hand, and to limit the access to data, e.g., to a certain type of data, on the other hand. The authors illustrate application scenarios of ACCESSORS for both the back-end layer and the smartphone layer, yet they do not provide any implementation details.

A simplified version of ACCESSORS is used in DISPEL [SGM20], a distributed privacy management platform for the IoT. In contrast to most existing privacy systems, DISPEL introduces data protection at the source level, i.e., access to sensor data is granted individually to authorized applications, rather than storing sensitive data in a central database and controlling access to it. Once access has been granted to an application, a gateway extracts data from the respective device. Afterwards, several privacy plugins can be used to transform sensitive data, e.g., to reduce the accuracy of a location prior to forwarding the data to the inquiring application. ACCESSORS follows the privacy-by-default approach, that is, applications initially do not have any permission. Permissions to access data have to be granted by users for every application and every application purpose individually, which causes substantial overhead in terms of permission administration.

3.3 Access Control Models based on UCON

The usage control (UCON) model has been proposed by [PS02] [Par03] [PS04] as a conceptual model for session-based access control introducing support for the mutability of attributes and decision sustainability, both of which are very important in the IoT context. Sustainable decision

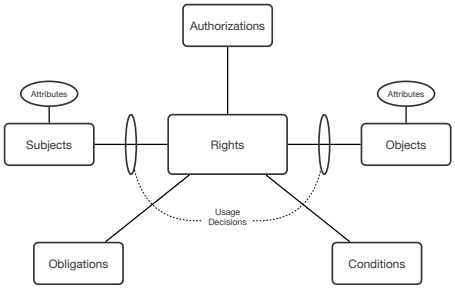

Figure 3.4: UCON core components model according to [PS04]

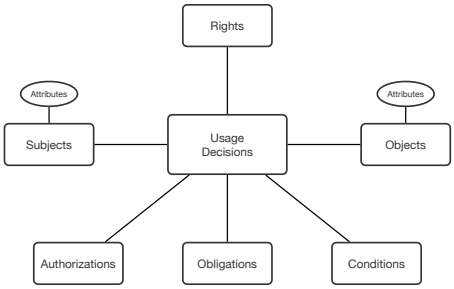

Figure 3.5: Alternative model according to [ZPSP05]

making is achieved by continuous permission evaluation and enforcement, i.e., authorization is enforced before, during, and after the access execution. Thereby, access might be revoked if certain conditions are not satisfied anymore due to changed attribute values. The core elements of the UCON model are shown in Figure 3.4. Subjects are entities that can aquire rights to access objects. Thereby, a right contains a set of usage functions that defines to what extent a subject can access or use an object. In contrast to other access control models, rights are not predefined, e.g., in an access control list, but rather determined exactly when an access request is made. Park et al. describe authorizations as functional predicates, which are evaluated by access decision routines in order to determine whether certain rights should be granted. Environmental information is considered through the incorporation of conditions in the decision making process, e.g., a certain right should only be granted during daytime. Lastly, using obligations, subjects can be forced to fulfill, or at least to agree on the fulfillment, of some mandatory actions either before the access is granted or during ongoing access. Note that both the authorizations and the obligations have to be fulfilled by the respective subject, while the conditions must be met by the environment. An alternative interpretation of the UCON model has been proposed in [ZPSP05], which emphasizes the usage

decisions, while the original version focuses on the relationships between the core components. It has to be noted that the aspect of administration and configuration regarding authorizations has not been addressed yet.

Guoping and Wentao propose an abstration of the UCON model in the IoT [GW11]. They provide a mapping between entities in UCON and the IoT and an assessment model based on fuzzy theory. Additionally, they present several small (theoretical) experiments to prove the expressiveness of the UCON model in the IoT context. However, the feasibility for implementation in real access control systems is not addressed [OMEO17].

Bai et al. present ConUCON [BYG+14], which incorporates context information into the usage control model aiming at enhancing data and resource protection in the context of the WoT. They provide the design and implementation for two specific use cases, a Smart Home platform and a remote appliance management prototype. However, the process of policy administration is quite cumbersome as the XML-based policies in ConUCON are complex, since the authors state that their policy model is derived from languages like XACML and SAML without explicitly defining which parts they use from which language. Furthermore, their approach is mainly focused on the WoT, which makes its suitability for IoT platforms questionable.

Similarly, Almutairi and Siewe [AS11] propose CA-UCON, a context-aware usage control model, which aims at improving the quality of service through access continuity, since in the original UCON access might be revoked if the environment changes, even if the subject still meets the requirements of the authorizations and obligations. In this case, CA-UCON tries to adapt to the new environmental situation by triggering certain actions without disrupting ongoing access sessions. However, details about which types of rules of changes in the envrionment their approach is applicable to are not provided.

3.4 Capability-Based Access Control (CBAC) Models

Dennis and Van Horn [DV66] first introduced the concept of a capability. Thereby, a capability is a token, ticket, or key that enables the entity holding it to access certain resources. Although the concept of capability-based access control faces two major downsides, namely the problem of capability propagation and revocation [Gon+89], it has been adopted in a wide range of IoT related systems [OMEO17]. Gong proposed the Secure Identity-based Capability System (ICAP) [Gon+89] to address the aforementioned drawbacks. The difference between the traditional capability-based approach resides in storing the identity of the subject along with the operation, aiming at facilitating capability propagation and improving the efficiency of capability storage [OMEO17]. A number of extensions to ICAP have been proposed [MAPP+13] [AMPP12] [AMP+13].

Gusmeroli et al. propose CapBAC [GPR13], a capability-based access control model for the IoT, which employs a central policy decision point for all authorization-related procedures and a separate capability revocation service. Similar to this approach, authors in [HJMS13b] [HJMS13a] propose DCapBAC, a distributed capability-based access control model, which implements the authorization functionality on IoT devices rather than a central platform. DCapBAC has been extended by authors in [BHS16] to create a trust-aware model based on fuzzy logic. However,

all of the discussed capability-based approaches lack at least one the following requirements: context-awareness, fine-grained policy definition, dynamic and continuous policy enforcement, or lightweight computation mechanisms.

3.5 Other Models

For the purpose of completeness, this section provides a brief summary regarding other access control models.

Organization-based access control (OrBAC) [KBB+03] extends the classic role-based access control model by adding an abstract level in order to enable administrators to specify policies independent of the actual use case/implementation. Approaches based on OrBAC include SmartOrBAC [BAAO15], Multi-OrBAC [AD06], and PolyOrBAC [KDBK09]. However, since these approaches ar based on OrBAC, which extends RBAC, the disadvantages dicussed in Section 3.1 apply as well.

Other approaches include task-based models, cryptography-based models, modifications and/or simplifications of XACML, and combinations of the previously mentioned appraoches. These are not be discussed here, since they are beyond the scope of this work. The interested reader is referred to survey papers by Ouaddah et al. [OMEO17], Servos and Osborn [SO17], and Cai et al. [CZH+19].

4 Requirements

This chapter presents the result of the requirements elicitation regarding general privacy in the context of the IoT. At first, a generic end-to-end IoT architecture is created which acts as a reference model for the requirements and conceptual model of the privacy and access control framework developed in this work. Along with this architecture we present a representative use case to help deduce the functional and non-functional requirements, presented later in this chapter. The next step is to conceptualize the design of the privacy framework based on these requirements, which is discussed in the next chapter.

4.1 End-to-End IoT Architecture

The end-to-end IoT architecture depicted in Figure 4.1, which is based on the architecture in [SGM20] and the reference architecture for IoT platforms by Guth et al. [GBF+16], provides insight into the dataflow between the components and entities in an IoT environment. This architecture

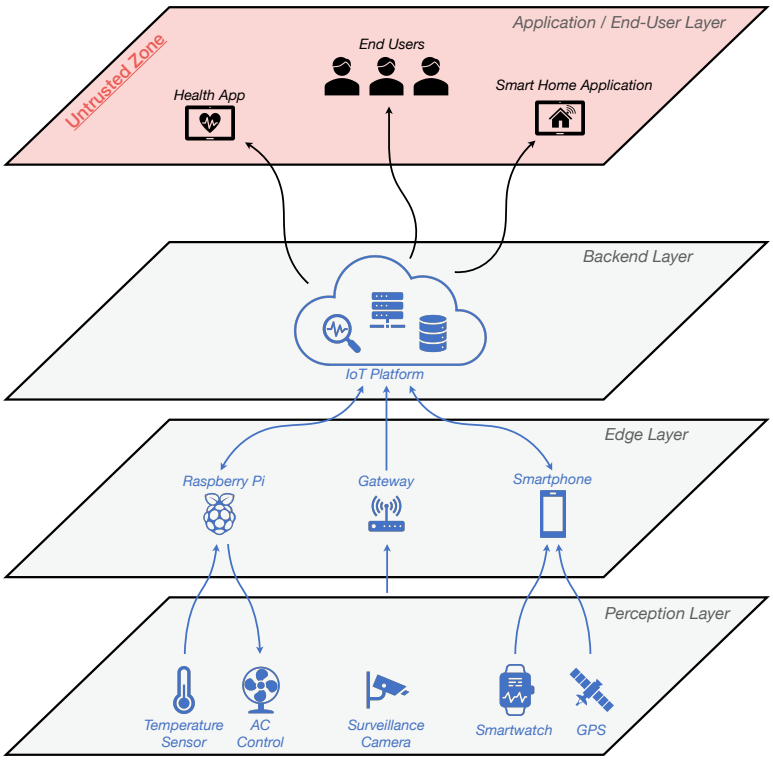

Figure 4.1: End-to-End IoT Architecture adapted from [SGM20] [GBF+16]

comprises four layers and thereby slightly deviates from the standard three-layer IoT architecture. The reason for this lies in connectivity-constrainted devices that cannot directly communicate with other devices and need a gateway, such as a smartphone connected via Bluetooth [SGM20]. Therefore, things like sensors, cameras, smart watches, or actuators reside in the perception layer are connected to, e.g., a Raspberry Pi, that is assigned to the edge layer, which is new compared to the standard three-layer architecture. In order to interconnect all these devices, the backend layer typically hosts a central IoT platform, which enables communication between the devices as well as data accumulation, storage, and analysis. Furthermore the backend layer provides the interface to the application and end-user layer, which covers (third-party) applications, such as health care apps or Smart Home solutions accessing data through an exposed API, and end users that access data through a graphical interface, e.g, an IoT platform web frontend. The top most layer is classified as an untrusted zone, since it encompasses third-party software and other users, for which no reliable assumptions can be made in terms of trustworthiness [SGM20]. We refrain from listing an exhaustive set of use cases and rather present an in-depth discussion of one representative use case which serves both as an illustration of the presented architecture and a foundation for the requirements eliciation.

The health care domain is perfectly suited for demonstrating both the emerging opportunities as well as the privacy concerns that come along with them. Figure 4.2 illustrates an example application scenario for a health care IoT platform. The patient weares a smartwatch that continuously monitores

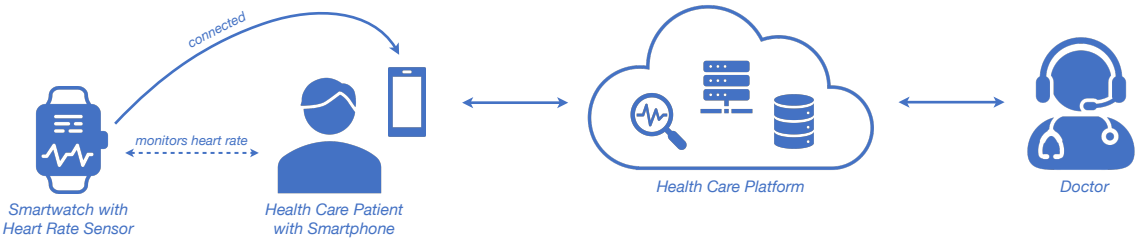

Figure 4.2: Health Care Scenario

its heart rate. Although the smartwatch might be able to detect an emergency in case of a very low or high heart rate, more complex diagnosis requires both a broader range of medical data and more computing power than the smartwatch can provide on its own. Therefore, the measured data is made available to the health care platform via the connected smartphone, which then is capable to perform a more sophisticated evaluation of the current situation, e.g., by including additional data, such as the patient's current location, into the analysis. Based on the result of this evaluation, the appropriate actions can be triggered, such as informing a doctor or even calling an ambulance. Thinking one step further, the data of multiple patients could be used for research purposes, e.g., to study the effectiveness of a newly developed drug. On this note, the arising privacy concerns regarding the patients, which act as representatives for any end-user in an IoT application scenario, have to be addressed with a privacy-preserving concept. Therefore, the next sections cover the elicitation of the functional and non-functional requirements towards this concept.

4.2 Functional Requirements

Personal (medical) data is highly sensitive, therefore it should be shared only if absolutely required (e.g., in an emergency situation), since medical concerns supersede privacy concerns. To guarantee this, there are two aspects, that have to be covered by a privacy-preserving concept in an IoT environment:

1. Configuration, and
2. Enforcement.

The first aspect, configuration, has to enable users (patients) to control, who can access which (type of) data under which conditions and to what extent. E.g., a patient might want to share its heart rate and blood sugar measurements at all time while granting access to its current location or its audio-visual peripheral equipment (e.g., camera and microphone) only in emergency situations. The second aspect, enforcement, has to make sure, that the users' privacy configuration is enforced, i.e., that entities in the untrusted zone (cf. Figure 4.1) are granted access only if permitted by the respective user's privacy configuration.

Based on these aspects, we can now formally define the functional requirements shown in Table 4.1:

FR1	The system should allow a 1:n owner-mapping from a user to an entity in the edge or perception layer.
FR2	The system should implement a privacy-by-default approach, i.e., initially no access is granted to any requesting entity from the untrusted zone except for the owner.
FR3	The system should allow entity owners to specify conditions under which certain entities are granted certain access.
FR3.1	These conditions must comprise different contextual information of the requesting entity as well as the requested entity, and
FR3.2	The possibility to restrict the data presented to the requesting entity in terms of, e.g., completeness or accuracy.
FR4	The system should provide a user interface for privacy configuration purposes.
FR5	The system should enforce the users' privacy configuration, respectively.

Table 4.1: Functional Requirements

FR1 is the most basic but also most essential requirement, as it guarantees the respective user's right to self-determined privacy configuration. Any device, sensor, actuator or other entity from the perception or edge layer is mapped to exactly one owner, while every user may be the owner of one or more entities. In the health care use case this could mean, that a patient is the owner of a wearable metering device. FR2 ensures that each and every access to, e.g., sensor data or an action performed by an actuator, is explicitly granted by the respective entitiy owner. Compared to a blacklist-based appraoch, where the owner has to restrict certain entities from certain access types, this approach provides a more secure and user-friendly way of privacy configuration. The third functional requirement, FR3, introduces context-awareness into the decision-making process as well as gradual / partial access permissions. FR3.1 enables the owner to specify conditions, that must be met to be granted access. These conditions evaluate contextual information of both the requesting entity and the requested entity. Revisiting the health care scenario, an example would

be for doctors to be granted access to medical data only in case some threshold, such as blood pressure, is exceeded. Using contextual information reduces the effort of privacy configuration, since conditions can be defined in an identity-less way. FR3.2 allows very fine-grained access decision results, due to the possibility to grant access to only parts of sensor data. For instance, a patient in the health care scenario might want to share its location with only a certain accuracy when not in an emergency situation. With FR4, it is ensured, that users can view and configure their privacy settings easily through a graphical interface. Lastly, FR5 guarantees, that the privacy configuration is actually evaluated and subsequently enforced whenever an entity requests access to a resource, for which it is not registered as the owner. This is important, since configuration without enforcement is worth nothing.
In addition to the previously discussed requirements, the health care scenario as well as the context of the IoT pose more requirements. However, these requirements relate to challenges, that cannot be addressed with purely functional requirements. Therefore, the next Section provides the elicitation of the non-functional requirements based on our representative health care scenario on the one hand and a survey of related literature [BAAO15] [OMEO17] [SM19] [SGM20] on the other hand.

4.3 Non-Functional Requirements

In this section, the results of the analysis of non-functional requirements is discussed. The following high-level non-functional requirements have been identified:

1. Expressiveness
2. Continuity
3. Transparency
4. Extendability
5. Efficiency
6. Usability

Each of these non-functional requirements is explained in detail. Furthermore, several aspects (sub-requirements) are presented, respectively. An overview of the non-functional requirements is shown in Table 4.2.

4.3.1 Expressiveness

Due to the heterogeneous nature of entities participating in IoT application scenarios and the resources provided by them, a privacy framework for the IoT has to stand out in terms of expressiveness. In order to achieve an expressive solution, the following aspects have to be addressed: *a)* context-awareness, *b)* granularity, *c)* a data-centric permission model [SM19], and *d)* data minimization [SGM20]. The first aspect, context-awareness, is also part of FR3.1 (cf. Table 4.1) and means, that the decision-making process should be based not only on static rules but rather incorporate the current state of the entities involved in the respective access request. The second aspect, granularity, on the one hand increases expressiveness by enabling a more fine-grained selection of entities, such as a specific doctor instead of any (available) doctor or a specific device rather than all devices connected via Bluetooth. On the other hand, a fine-grained

privacy configuration allows sophisticated permissions compared to the all-or-nothing approach, which either completely grants or denies access. To realize this fine granularity of the privacy framework, the following two aspects are essential. A data-centric permission model introduces an abstract layer below the perception layer in our reference architecture (cf. 4.1), which encompasses the data procuded by sensors and the actions provided by actuators. The aspect of data minimization goes one step further as it aims at modifying the data presented by means of, e.g., adding noise to or decreasing the accurary of the data.

4.3.2 Continuity

In addition to the heterogeneity of the entities in IoT application scenarios, another key aspect is the constantly changing state of these entites (just think of the location of smartphone somebody carries around). Therefore, the evaluation and enforcement of privacy configurations should be performed in a continuous way, instead of a one-time or session-oriented manner. Moreover, when a set of permissions granted to an entity is altered, it is important, that these changes come into effect immediately. The following aspects have to considered: *a)* real-time evaluation & enforcement, and *b)* continuous evaluation & enforcement based on up-to-date (contextual) data. The process of evaluating and enforcing should be done in real-time to avoid the lost-update problem, which could occurr if changes are made to the privacy configuration between the start of the evaluation process and the permission grant, which then would be based on outdated data. Likewise, changes to the privacy configuration could be made right between two consecutive access requests where the requesting and requested entities are identical, respectively. If evaluation is performed only for the first request, the changes to the privacy configuration won't be reflected in the second access request. This is why each access request should be evaluated indepedently, which is what the second aspect aims at.

4.3.3 Transparency

An essential requirement closely related to privacy is the aspect of transparency. To achieve transparency in an IoT privacy framework, the following aspects have to be dealt with: *a)* access purpose transparency [SGM20], and *b)* derivation transparency [SM19]. In order to decide whether to grant access to a (potentially unknown) third-party application or person, users should be able to see what their data is used for, which is referred to as the access purpose [SGM20]. E.g., a patient in the health care example might agree to share its medical records with a health care service for the purpose of medical advice from a doctor but not for marketing purposes of pharmaceutical companies. Furthermore, data might be correlated with regard to derivation, e.g., X could be derived from Y. Now, if the data owner decides not to share X, but at some point wants to share Y, the privacy framework should at least generate a warning, which indicates that by sharing Y, entities with access to Y might be able to derive X.

4.3.4 Extendability

The ever-changing nature cannot only be observed with the entites in IoT application scenarios, but also with the concept of the IoT itself. New (communication) technologies, platforms, and concepts are constantly emerging. Therefore, a very important characteristic of a good privacy framework for the IoT is extendability. The key factor to achieve an extensible solution is generic design, which should be applied to the following artifacts of the privacy framework: *a)* the domain

model for configuration, and *b)* the evaluation and enforcement facilities. The domain model should be as flexible as possible in order to support a wide range of (future) use cases. Additionally, a generic domain model is important to decrease the effort of privacy configuration, e.g., by providing identity-less and attribute-oriented configuration mechanisms and enabling reusable privacy settings. The components and algorithms used for evaluation and enforcement should be generic in terms of where they can be deployed and executed, in order to support both centralized approaches as well as distributed approaches. The latter approach becomes increasingly relevant, since we can observe a „smartness shift from the center to the edge“ [OMA15].

4.3.5 Efficiency

Following up on the distributed architecture approach discussed in the last section, efficiency is a crucial requirement, since the IoT is resource-constrained by design. Two aspects have to kept in mind when designing and implementing an IoT privacy framework: *a)* lightweight computation, and *b)* scalability. The importance of lightweight computation is obvious if the edge layer is involved in the evaluation and enforcement of privacy settings. However, it cannot be disregarded even if all evaluation and enforcement functionality is implemented on a central platform, which has access to sufficient resources. The number of entities participating in IoT application scenarios tends to grow exponentially and might lead to substantial scalability issues, if evaluation and enforcement requires huge amounts of computational resources. Scalability is immediately linked with the continuity aspect (cf. Section 4.3.2), since a non-scalable solution can affect real-time decisions. Revisiting the health care scenario, this might add some delays in an emergency situation, which in turn might have fatal consequences for some patients.

4.3.6 Usability

Last but not certainly not least, a privacy framework for the IoT should address the following aspects in order to achieve a high degree of usability: *a)* efficient privacy configuration mechanisms, and *b)* an easy-to-understand permission model. The interface and mechanisms provided to users for the configuration of privacy settings should support easy and efficient configuration workflows. If users have to navigate through complicated and extensive configuration strucutures, they are likely to configure their privacy settings either too restrictive, due to coarse-grained access restriction to avoid having to create different fine-grained permissions, or too loose, due to no configuration at all. While this correlates to the accuracy of permissions, in order for them to also be defined correctly, the permission model should be easy-to-understand for the user. This aspect correlates to the first non-functional requirement, expressiveness (cf. 4.3.1), as the permission model should be highly-expressive yet easy-to-understand at the same time. Otherwise, the benefits of an expressive permission model could be annihilated.

NFR1	Expressiveness
	Context-Awareness
	Granularity
	Data-Centric Permission Model
	Data Minimization
NFR2	Continuity
	Real-Time Evaluation & Enforcement
	Continuous Evaluation & Enforcement based on up-to-date (contextual) data
NFR3	Transparency
	Access Purpose Transparency
	Derivation Transparency
NFR4	Extendability
	Domain Model for Configuration
	Evaluation and Enforcement Facilities
NFR5	Efficiency
	Lightweight Computation
	Scalability
NFR6	Usability
	Efficient Privacy Configuration Mechanisms
	Easy-to-understand Permission Model

Table 4.2: Non-Functional Requirements

5 Conceptual Design

This chapter presents the conceptual design of the access control system developed in this work. The concept is presented top-down, beginning with a high-level dataflow model and gradually zooming in to the access control domain model and syntax. We start with the architecture and integration of the access control system which are discussed with respect to a dataflow model for IoT environments. After that, the access control workflow is described based on which the elicitation of the respective components is presented. Finally, the core of the access control system – its domain model – is discussed.

5.1 Architecture and Integration

In this section the architecture is presented along with the rationale for the most important aspects. As a first step, we have to determine the dataflow in an IoT environment. Figure 5.1 shows an abstract dataflow model based on the reference architecture by Guth et al. [GBF+16]. The interaction

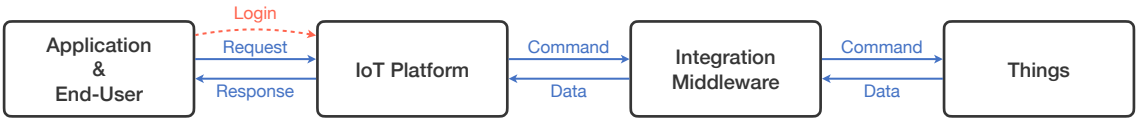

Figure 5.1: Dataflow in IoT environments

between users (including users in the form of applications) and the IoT environment is always handled by the central IoT platform. In the case of human users, this is mostly done through a web interface, whereas applications can communicate with the IoT environment through an exposed API, e.g., a REST API. The communication between the Things, such as devices and sensors, and the IoT platform is enabled by several middleware components, e.g., a message broker. This way, commands can be sent to the Things, e.g., to trigger certain actions, and data from sensors can be sent to the IoT platform.
Based on the established dataflow model, the following sections discusses several key architecture and integration issues and presents the solution that has been determined most suitable.

5.1.1 Privacy vs. Security

The purpose of an access control system is to restrict access to certain resources to only those users, that are authorized to access them, e.g., if the respective resource is owned by the user or particular access has been granted by the owning user to the requesting user. When designing the architecture of the access control system, it is important to thoroughly consider which functionality is related to privacy or access control and which functionality is already covered by security mechanisms in a respresentative target environment. For our concept, we assume that the following security measures are established in the target environment:

- Authentication of end-users and applications: every request from an end-user or an application to the IoT platform is authenticated, e.g., using HTTP basic authentication or OAuth2. Likewise, requests originating from the web frontend of an IoT platform require previous user login, i.e., authentication.
- Secure communication: the communication between all entities shown in Figure 5.1 is encrypted, e.g., using TLS [1] or lightweight protocols like the Constrained Application Protocol (CoAP) [2] considering resource-contrained devices.

Consequently, the scope of the access control system only has to encompass the authorization of requests and does not have to deal with authentication. This is also important when dealing with the domain model and syntax, since the design and scope of the domain model are directly related to the functional scope. Now that we differentiated privacy and security and determined the scope of the access control system, the next step is to determine how to integrate the access control system into the target environment.

5.1.2 Centralized vs. Distributed Architecture

There are several options for the architecture and integration of the access control system, which are discussed in the following. The central question is where and at which point in the dataflow (cf. Figure 5.1) the authorization process is performed.

Centralized Architecture The most obvious solution is to implement the access control system directly in the central IoT platform. This way, each request to the IoT platform is forwarded to the access control system and evaluated. Furthermore, the privacy settings for each user and entity can be stored in a central database within the IoT platform. The web frontend can be used for administration purposes by end-users. However, contextual information plays a very important role in the decision-making process, therefore, some overhead might be introduced when constantly having to retrieve contextual information, e.g., the state of a device.

Distributed Architecture Another solution is to outsource the access control logic to the Things themselves, which then can act autonomously. This way, the required contextual information is always available in real-time. However, the big disadvantage is the overhead that is caused by the access control processes. In addition to that, forwarding the request and the contextual information of the requesting user has to be dealt with. In contrast to the IoT platform, a Thing is usually resource-constrained, which makes this solution inappropriate for most IoT application scenarios. Moreover, with this distributed architecture, the delays attributed to access control processes are most definitely higher than the delays caused by access control processes in a centralized architecture, since in both cases contextual information has to be transferred and the IoT platform is certainly able to evaluate the access requests significantly faster.

Hybrid Architecture A third solution is to use a hybrid approach which is based on the centralized architecture. In addition to that, a lightweight access control system is added to the Things, which then are able to configure the data sent to the IoT platform and trigger actions to be executed based

[1] RFC-5246: The Transport Layer Security (TLS) Protocol
[2] RFC-7252: The Constrainer Application Protocol (CoAP)

on the current contextual information, completely detached from the current requests related to a particular Thing. With this approach, a 2-way access control system can be achieved, where some general conditions are created, that define when to send data to the IoT platform and when not to, and the access control system within the IoT platform handles the evaluation of each request. This way, it is possible to prevent sensitive or personal data to be sent to the IoT platform in the first place or at least to be anonymized, e.g., someone's current location during a certain time in the day.
Considering these three architectural approaches, the advantages of the hybrid architecture clearly prevail. Therefore, we implement the access control system based on the hybrid approach. The flexibility of this approach, i.e., the fact that the (additional) access control logic implemented on the Things can be seen as optional, makes it the most suitable option for most target environments.

5.1.3 Access Control Workflow

The access control workflow is heavily influenced by the way the entities in the IoT environment (cf. Figure 5.1) are communicating. The most important aspect is the request-response type of communication between the IoT platform and end-users or applications. In combination with the requirements from Chapter 4, especially context-awareness and continuiuty in the enforcement of privacy settings, a request-based access control workflow is most appropriate. Figure 5.2 illustrates

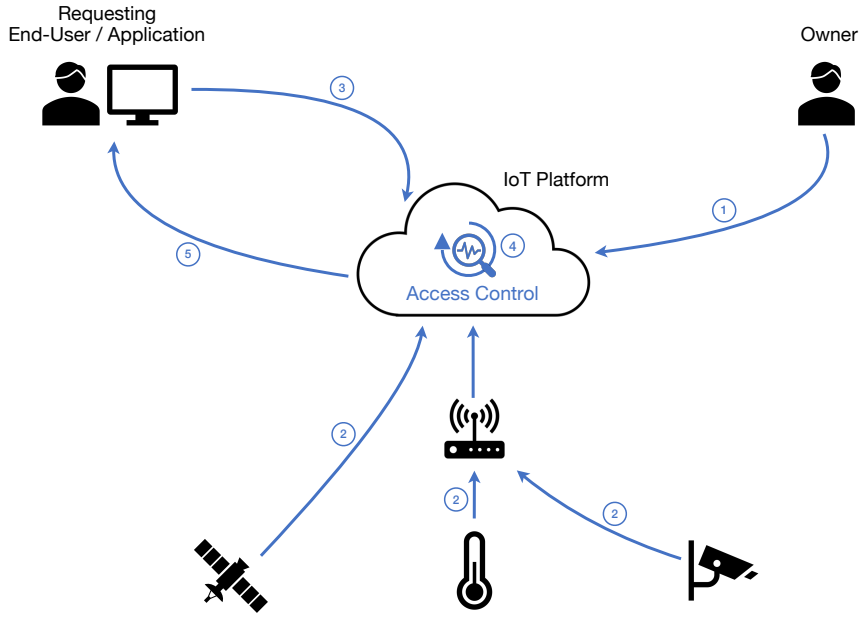

Figure 5.2: Access Control Workflow

the generic, request-based access control workflow, which is described in the following.

1. In a first step, a user registers a new Thing, e.g., a smart camera, and creates several privacy settings accordingly.

2. Once the camera has been registered, it sends data to the IoT platform depending on the privacy settings. For now, let's assume that the camera is sending (raw) data.

3. Now another user can request the data from the camera.

4. The IoT platform receives the request and immediately can determine the requested entity and the corresponding privacy settings defined by the owner. The request is forwarded to the integrated access control system, which evaluates the request and applies the appropriate constrains to the data, e.g., reducing the resolution of the images obtained from the camera.

5. The requesting user receives the response which contains the result of the evaluation of the access request. In case access is granted, the response contains the requested data.

Using this request-based access control workflow, continuous enforcement of access requests is covered, since each request is processed and evaluated individually in contrast to session-based or token-based models. Now we have to consider the aspect of context-awareness. The challenge is to provide the contextual information of both the requesting entity (user) and the requested entity to the access control system. This is addressed in the next section.

5.1.4 Access Control Components

Until now, the access control system has been described from a black-box point of view. The next step is to zoom in and discuss the component-level design of the access control system. Therefore, this section presents the various components of the access control system as well as several key design decisions. In order to determine the required components for the access control system, we have to look at the required functionality and how we can achieve an optimal cut of the functionality into components. Figure 5.3 provides an overview of the components in the access control system as well as the interaction between them. The interface to the IoT platform is the component that receives the requests sent to the IoT platform (by end-users or applications), here called the *Request Receiver*, which for instance could be a controller for REST endpoints. The numbers on the connections

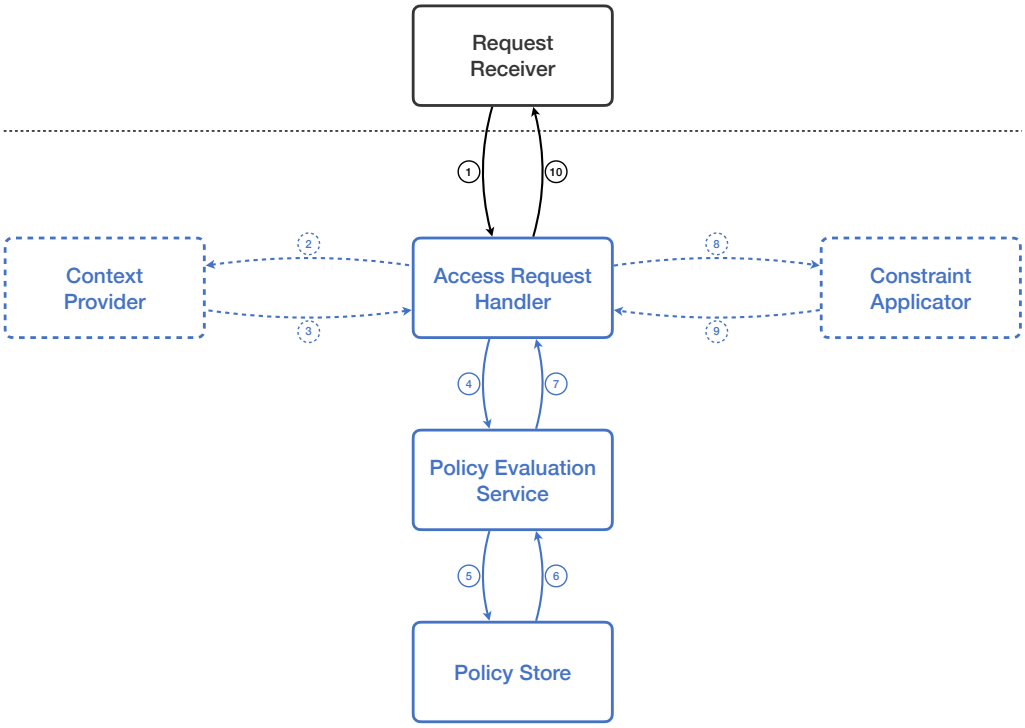

Figure 5.3: Access Control Components

between the components illustrate the workflow inside the access control system. Dashed lines indicate optional components. Due to the modular architecture with optional components, a high degree of flexibility is achieved which allows to tailor the access control system individually to different target environments. For instance, the full-blown version of the access control system could be implemented in the IoT platform, while employing a lightweight version on the Things themselves.
The main component in the access control system is the *Access Request Handler*. It acts as a coordinator for the access control workflow and receives the access request from the IoT platform's request receiver along with information about the requesting and requested entity (1). The requesting entity may include contextual information in the request. However, it is hard to verify the validity of this information, which is why contextual information sent by the requesting entity should only be considered if the requesting entity is trusted, e.g., if the request originates from the IoT platform's web frontend. The *Context Provider* is responsible for determining and providing (additional) contextual information with regard to both the requesting and requested entity based on trusted sources (cf. [HS16c]). There are many conceivable options for implementing more sophisticated context providers, e.g., based on situation recognition technology. Furthermore, the number of context providers is not limited and the access control system could just as well connect to an external attribute provider. If a *Context Provider* is included in an access control system implementation, the *Access Request Handler* requests (additional) contextual information from the *Context Provider* (2), which responds with the information that is currently available on the requesting and requested entity, e.g., the current state of a device (3). From now on, we use the term *policy* when refering to privacy configuration or privacy settings. Once all information required for the evaluation of an access request has been retrieved, the request is forwarded to the *Policy Evaluation Service* (4). Based on the requested entity, the *Policy Evaluation Service* retrieves the corresponding policies from the *Policy Store* (5) (6). The *Policy Store* could be a separate database within the access control system or could also be integrated into an existing database of the IoT platform. The *Policy Evaluation Service* is then able to evaluate the given policies based on the contextual information. The result of the policy evaluation process is returned to the *Access Request Handler* (7). In case access is denied, the decision is immediately returned to the IoT platform's request handler. If access is granted, it has to be checked whether the policy, that grants access, specifies one or more constraints for the access. These constraints may filter a given set of data or also modify data, e.g., for anonymization purposes. The functionality to apply constraints is implemented in the *Constraint Applicator*, which is an optional component and can be ommitted for simple use-cases that don't require the application of constraints. If a *Constraint Applicator* is available in the access control system, the *Access Request Handler* sends the data to constrain to the *Constraint Applicator* (8) and receives the constrained data (9). The access control workflow is now complete, i.e., the decision and (constrained) data can be returned to the IoT platform's request handler.

5.1.5 Key Design Decisions

Based on the access control workflow presented in the previous section, there are several implications on key design decisions regarding the underlying access control model.
In Chapter 3, various approaches to access control models have been presented. Since we have to provide context-aware access decisions, the model we choose is crucial. Role-based access control models lack the required degree of expressiveness. This is often remedied by overengineering roles, however, it is not a recommendable way of achieving fine-grained permissions in highly dynamic environments. The remaining two types of models discussed earlier, UCON-based and

capability-based access control models, both fall short in terms of context-awareness, lightweight evaluation of policies, and extendability (cf. Section 3.3 and Section 3.4). Furthermore, the amount of research done in these resepective areas is limited, which results in fewer proposed approaches and extensions, especially in the context of IoT environments. With attribute-based access control there is a very powerful solution to access control in IoT environments, which is perfectly suited to support context-aware access control, since attributes can be used to easily capture contextual information. Moreover, the attribute-based approach, due to its generic nature, can be extended and customized very easily and quite extensively.
Another very important aspect is whether to use a whitelist-based or blacklist-based approach for the definition of policies. The whitelist-based approach would grant access as soon as there is a policy that grants that access, whereas with the blacklist-based approach, every policy has to be evaluated in order to determine whether there is a policy that denies that access. Therefore, we choose the whitelist-based approach, since it reduces the overhead caused by the evaluation policies on the one hand, and simplifies the process of access configuration, i.e., defining policies, for the users on the other hand.

5.2 Access Control Domain Model

Now that the component-level architecture and integration has been established, we can zoom in even more and have a look at the elements of the access control model. In this section, the domain model and the syntax of the access control system are presented. Figure 5.4 shows the complete domain model in the form of a entity-relationship diagram. In the following, each element in the model is described in detail.

A *requesting entity* can either be an end-user, a third-party application or a Thing in the IoT environment itself. These entities can request *access* to certain other entities, called *requested entities*. A requested entity can be something real, e.g., a physical device, but it can also be seen as an abstraction for data [SGM20]. Using this data abstraction, users can specify fine-grained policies (cf. Section 4.3.1), e.g., a user could specify a policy only for location data and a separate policy for other data instead of one policy for the entire device or sensor the data originates from.
Access to entities is further differentiated based on the *Access Type* and the *Access Purpose*. The access type indicates whether, e.g., reading or writing access is requested, whereas the access purpose specifies, what the requesting entity intends to use the requested access for. Revisiting the health care scenario from Chapter 4 (cf. Figure 4.2), the doctor could specify „perform diagnosis“ as the access purpose. Access type and purpose are also reflected in the specification of policies, which are discussed later.
In the previous section, we already mentioned the importance of contextual information with regard to context-aware access control decision-making. Since we are using an attribute-based access control model, *context* is simply a set of attributes. Both requesting and requested entities are characterized by one or more attributes. An *attribute* is defined as a key-value pair and could hold any information that comes to mind. Even a legacy role-based access control system could be incorporated into our attribute-based system by simply adding an attribute „role“.
The core of the domain model are *policies*. A policy specifies whether access should be granted or denied based on one or more *conditions*. In addition to that, policies are prioritized for two reasons: 1) to allow one policy to overrule another policy, and 2) to enable fast access request evaluation. Policy conditions can be of one of three forms: *simple condition*, *composite condition*, or *situation condition* [HS16c]. A simple condition basically is an equation with two arguments and a function

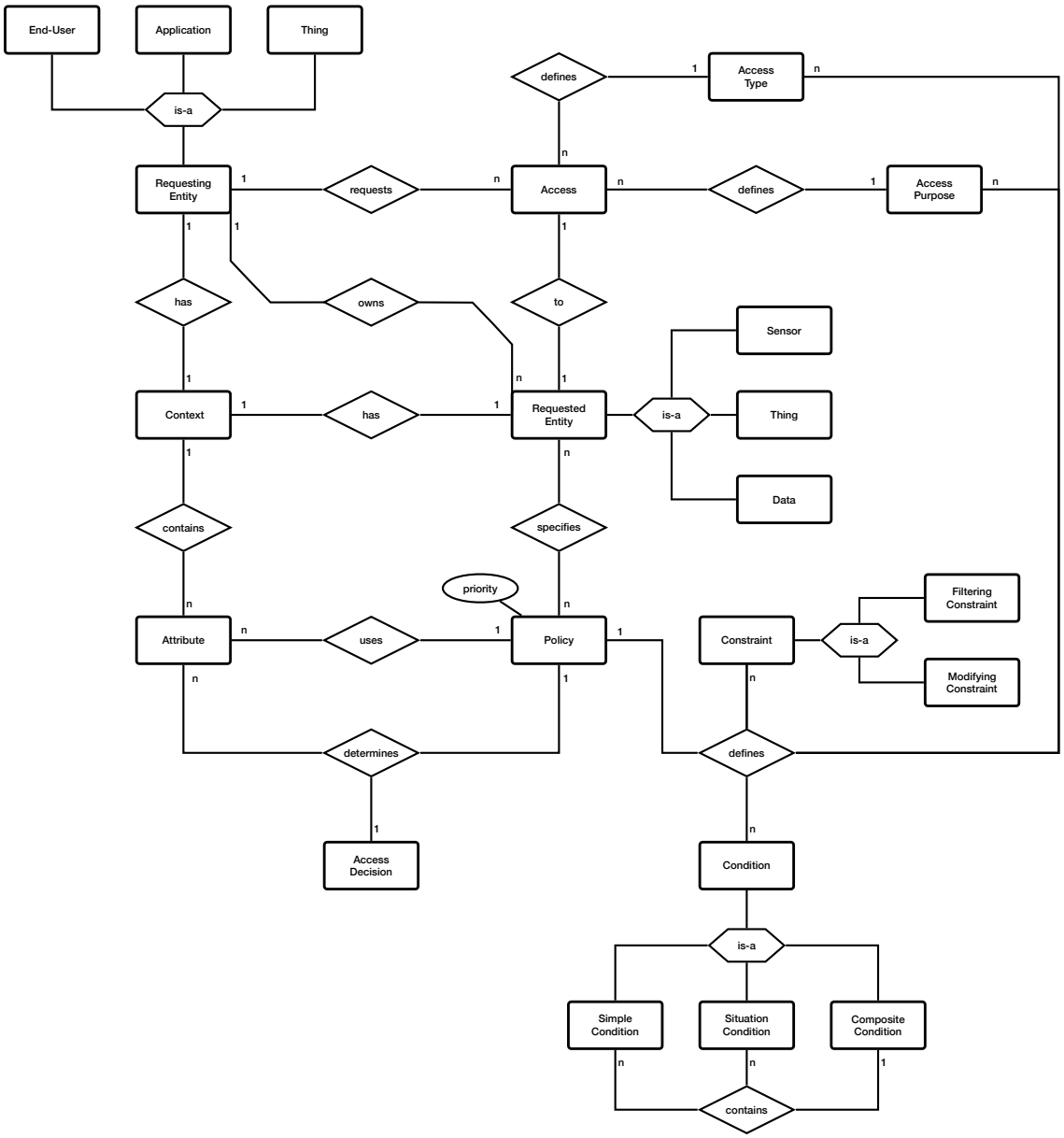

Figure 5.4: Access Control Domain Model

to compare the two arguments. The left side of the equation always is an attribute value that has to be evaluated. The right side of the equation can either be a fixed value or an attribute value. By using attribute values, conditions can be evaluated based on dynamic values, i.e., based on up-to-date data (cf. Section 4.3.2). With composite conditions, there is a way to specify more complex conditions by building boolean expressions based on simple or also nested composite conditions. Situation conditions take the degree of context-awareness to another level by allowing policies to be based on situations determined by situation recognition and complex event processing technology. In addition to this expressive set of conditions, *constraints* allow to further restrict the access granted to the requesting entity if the conditions hold. Two types of constraints are available – *filtering constraints* and *modifying constraints*. An application for filtering constraints could be to

filter a range of values based on an upper and lower bound. An example for a modifying constraints would be to reduce the accuracy of the returned data, which is of particular interest when dealing with location data. It would further be possible to completely pseudomyize or even anonymize the data presented to the requesting entity.
The *access decision* is finally determined by the policies based on the conditions and constraints. Possible access decisions are: access denied, access granted, and access granted with constraints.

5.2.1 Syntax

To give an idea of how the domain model presented previously can be implemented from a data format viewpoint, Listing 5.1 provides an example for a policy written in JSON. The policy specification is done independent of particular requested entities, which allows to re-use policies for several requested entities. Moreover, the process of assigning or removing a policy to or from a certain requested entity can be implemented by simply updating the list of policy ids associated with a requested entity. Another advantage of this solution is that the set of policies to evaluate for an access request can be directly obtained with the requested entity itself.
The policy in Listing 5.1 contains a composite condition with two simple conditions that are linked with the logical AND, i.e., both of the simple conditions have to evaluate to „true“ in order for the composite condition to hold. The policy grants access to a fixed group of users and to all employees that are on or above the senior level. If access is granted based on the condition, the constraint of the policy restricts the accuracy of the returned data (e.g., some financial figures) to an accuracy of ten, i.e., 87.5 would be returned as 90.
It has to be mentioned, that the presented syntax should only be regarded as an example for a concrete implementation of the domain model. The actual syntax and implementation should be designed individually and with focus on the respective target environment.

5.2.2 Evaluation

The last part of the access control system to discuss is the algorithm that is used to evaluate access requests. This functionality is implemented in the *Policy Evaluation Service* (cf. Section 5.1.4).

```
Input: Requested entity e, Context of requesting and requesting entity c, access type, access purpose
Output: Access decision

policies ← getPolicies(e, access_type, access_purpose)
policies ← sort(policies, priority)
for Policy p in policies do
       if evaluate(p, c) == True then
              if p.constraint then
                     e ← applyConstraint(e, p.constraint)
              end if
              return AccessDecision(GRANTED, e)
       end if
end for
return AccessDecision(DENIED)
```

Algorithm 5.1: Evaluation algorithm for access requests

Listing 5.1 shows the evaluation algorithm for access requests. It has four input parameters: the requested entity, the context (attributes) of both the requesting and requested entity, the access type, and the access purpose. The generated output is the access decision (access granted or denied)

```
{
    "id": "5f589ddff6b51b0e096b09c0",
    "accessTypes": [ "READ" ],
    "priority": 1,
    "conditions": [
        {
            "id" : "5f5899cd76c03a3b43488579",
            "operator" : "AND",
            "conditions": [
                {
                    "id": "5f5899cd76c03a3b43488577",
                    "function": "IN",
                    "left" : {
                        "entityType" : "REQUESTING_ENTITY",
                        "key" : "REQUESTING_ENTITY_USERNAME"
                    },
                    "right": {
                        "value": [
                            "user-1",
                            "user-2",
                            "user-3"
                        ]
                    }
                },
                {
                    "id": "5f79a2df427c03058980d52b",
                    "function": "GREATHER_THAN_OR_EQUAL_TO",
                    "left" : {
                        "entityType" : "REQUESTING_ENTITY",
                        "key" : "REQUESTING_ENTITY_EMPLOYEE_LEVEL"
                    },
                    "right": {
                        "value": "SENIOR"
                    }
                }
            ]
        }
    ],
    "constraints": [
        {
            "id": "5f79a491881a1566315b91dd",
            "parameters": {
                "precision": "0",
                "accuracy": "10"
            },
            "type": "NUMERIC_ACCURACY_MODIFICATION"
        }
    ]
}
```

Listing 5.1: Policy with condition and constraint in JSON format

which contains the possibly constrained requested entity if access should be granted.
The algorithm first retrieves all policies for the requested entity based on the access type and access purpose. This way, non-applicable policies are filtered before even processing them. The set of applicable policies is then sorted by priority, before iterating over each policy and evaluating the policy condition. As soon as the first policy (condition) evaluates to „true“, the requested entity is returned either directly or after applying the respective constraint. If no condition evaluates to „true“, access is denied.

6 PoC Implementation for the MBP

The last chapter presented the conceptual design of an access control system for generic IoT environments. Now, we want to prove that the concept actually works. Therefore, the concept has been implemented for the MBP IoT platform (cf. Section 2.1.1). As in the previous chapter, we present the implementation of the access control system by starting on the architecture level and gradually zooming in to the concrete implementation details.

6.1 Architecture of the MBP IoT Platform

Figure 6.1 shows the architecture of the MBP IoT platform, from now on simply referred to as MBP, including the used technologies. The core of the MBP is its backend service which is implemented in Java and builds on top of Spring Boot. It is the central coordinating facility in the MBP IoT environment. The MBP supports three kinds of Things: devices, sensors, and actuators. The latter two usually are connected to a device, since they do not provide appropriate communication capabilities. Communication between the backend service and the devices is then enabled by a Mosquitto MQTT Broker, which allows retrieving sensor data and sending commands using the publish-subscribe communication pattern. The MBP employs two separate data stores. Data from sensors is stored in an InfluxDB time-series database, all other data resides in a MongoDB document-based database. Users can interact with the MBP through the web frontend or the Android app, both of which use the REST API exposed by the backend service, which can as well be used directly by other applications.

6.2 Scope of the MBP Access Control System

When using the concept developed in this work to implement an access control system for an IoT platform, the first step is to analyze the functionality implemented in the IoT platform and to determine, which of these functionalities have to be covered by access control and which parts of the concept should be implemented.

6.2.1 Architectural Scope

An important aspect is the integration of the access control system into the IoT platform. As discussed in Section 5.1.2, there are different ways of doing that. The central question is whether we can support access control directly in the edge or even perception layer (cf. FIgure 4.1). To make a sophisticated decision, we need to have a closer look at how devices are managed (e.g., started and stopped) and how sensor data is actually extracted and sent to the IoT platform. The MBP employs a solution based on scripts, which have to be provided by the user, ususally this is the user that creates the device, sensor, or actuator. These scripts are deployed onto devices and are in charge of the actual communication with the Mosquitto MQTT broker. Therefore any access control logic on the device layer would have to be implemented in these scripts. Since they have to be provided by

end-users, which usually have limited technical skills, we limit the access control system to the backend service, thereby creating a centralized architecture as described in Section 5.1.2. More details on the implementation of the access control system is given later on in this chapter.
Another important question from an architectural point-of-view is where to store the policies for the access control system. Information about devices, sensors, and other entities is stored in the MongoDB database. One solution for storing policies would be to add another database to the MBP architecture, e.g., a graph database, which is suitable for the highly interconnected entities and policies. However, focusing on simplicity and the proof-of-concept style of implementation, the policies are stored in the same database as the entities they refer to in order to not further increase the complexity of the system. Policies are referenced by entities, which takes care of easy and lightweight retrieval of policies for certain entities.
The final question we have to address is how the attributes of the requesting and requested entity are made available to the access control system, respectively. The concept provides two solutions for that – using the attributes sent by the requesting entity along with the actual request (only for attributes of the requesting entity), and using an attribute provider, which can either be an internal component or an external component. We implement both solutions for the MBP. Requesting entities can send attributes using a special HTTP header (`X-MBP-Access-Request`). An internal attribute provider is used to provide additional attributes on the one hand, and to act as a verification facility for the attributes sent by requesting entities on the other hand. The subject of attribute verification is crucial in the context of access control, because without a verification step before the actual evaluation, requesting entities might be able to access restricted entities by manipulating certain attribute values.

6.2.2 Functional Scope

Having determined the architectural scope and design, the next step is to define the use-cases, that are covered by the access control system. Firstly, the following list describes the use-cases of the MBP:

1. Creating, retrieving, deleting, and sharing devices, sensors, and actuators.
2. Creating, retrieving, deleting, and sharing adapters to monitor devices.
3. Creating, retrieving, deleting, and sharing adapters to extract and monitor sensor data.
4. Creating, retrieving, deleting, and sharing operators to execute actuator actions.
5. Creating, retrieving, deleting, and sharing rules with triggers that can execute certain actions.
6. Modeling an IoT environemnt with a graphical sketching tool.
7. Testing rules with an integrated testing tool.

Access control is implemented for the first four use-cases. Rules are ommited since they are based on entities, that are already covered by access control for other use-cases. The environment modeling tool and the testing tool are still in development and therefore won't be included. In the following, we discuss the features of the concept (cf. Section 5.2 that are included in the access control system for the MBP.
The access to a requested entity is characterized by an access type, which can be either one of the following values: `READ`, `READ_VALUE_LOGS`, `UPDATE`, `DELETE`, `START`, `STOP`, `DEPLOY`, `UNDEPLOY`, and

MONITOR. Access purposes are not implemented as a dedicated element in the access request since they are not required for the use-cases currently implemented in the MBP and would introduce unnecesary complexity for the end-user. Regarding the types of requested entities, the access control system supports the following entity types: DEVICE, SENSOR, ACTUATOR, and ADAPTER. Data as a requested entity is not supported since the MBP currently does not support differentiation beyond sensors, i.e., a sensor can be seen as an atomic unit wihtin the MBP. For the protoypical implementation of the access control system, the context of requested entities is not incorporated in the decision making process. For specifying the conditions of a policy, the MBP access control system implements two types of conditions – a simple condition to compare two (attribute) values, and a composite condition that allows to create logical expressions based on simple conditions. For constraining the data presented to the requested entity, the MBP access control system offers a modifying constraint for sensor values. Using this modifying constraint, the requested sensor values can be altered in terms of accuracy and precision, e.g., by rounding values to the left of the decimal point to tens.

6.3 Conceptual Aspects and Implementation

The MBP access control system employs the attribute-based access control model as well as the whitelist-based evaluation approach and the request-based architecture as suggested by the concept presented in the previous chapter. In the following, the integration of the access control system into the MBP backend is discussed. Moreover, the approach for policy and condition specification and evaluation is presented.

6.3.1 Access Control Integration and Workflow

As mentioned in the previous section, the MBP employs a central access control system. Figure 6.1 illustrates the workflow of an access request in the context of the MBP and provides insights into the MBP access control system. The workflow on an architectural level is basically identical to the workflow presented in Section 5.1.3. Note that the step of registering an entity with the MBP is ommited in Figure 6.1. In the following, the design of the MBP access control system is presented from a component-level point-of-view, since the concrete implementation differs from the generic design presented in Section 5.1.4. The MBP access control system encompasses steps seven to ten as shown in Figure 6.1. Before introducing the different components of the MBP access control system and the integration into the MBP backend, it has to be mentioned, that the MBP employs an abstraction for all entities that can be requested by users, called `UserEntity`. Since the design of the MBP backend is based completely on this abstraction, the MBP access control system builds on top of that, i.e., every requested entity is a `UserEntity`. Furthermore, REST controllers are available for each `UserEntity`, which enables easy identification of the request entity as well as the requested access type, e.g., a request to `/MBP/api/sensor/98fb7d5/deploy` indicates access to deploy the sensor with id `98fb7d5`, where deploy is the access type. The MBP domain model is extended in a way that adds a policy property to each `UserEntity`. Consequently, policies (and conditions and effects) are stored in the MongoDB. With the identification of the access request and storage of policies taken care of, the following list provides the additional components required to implement the access control system.

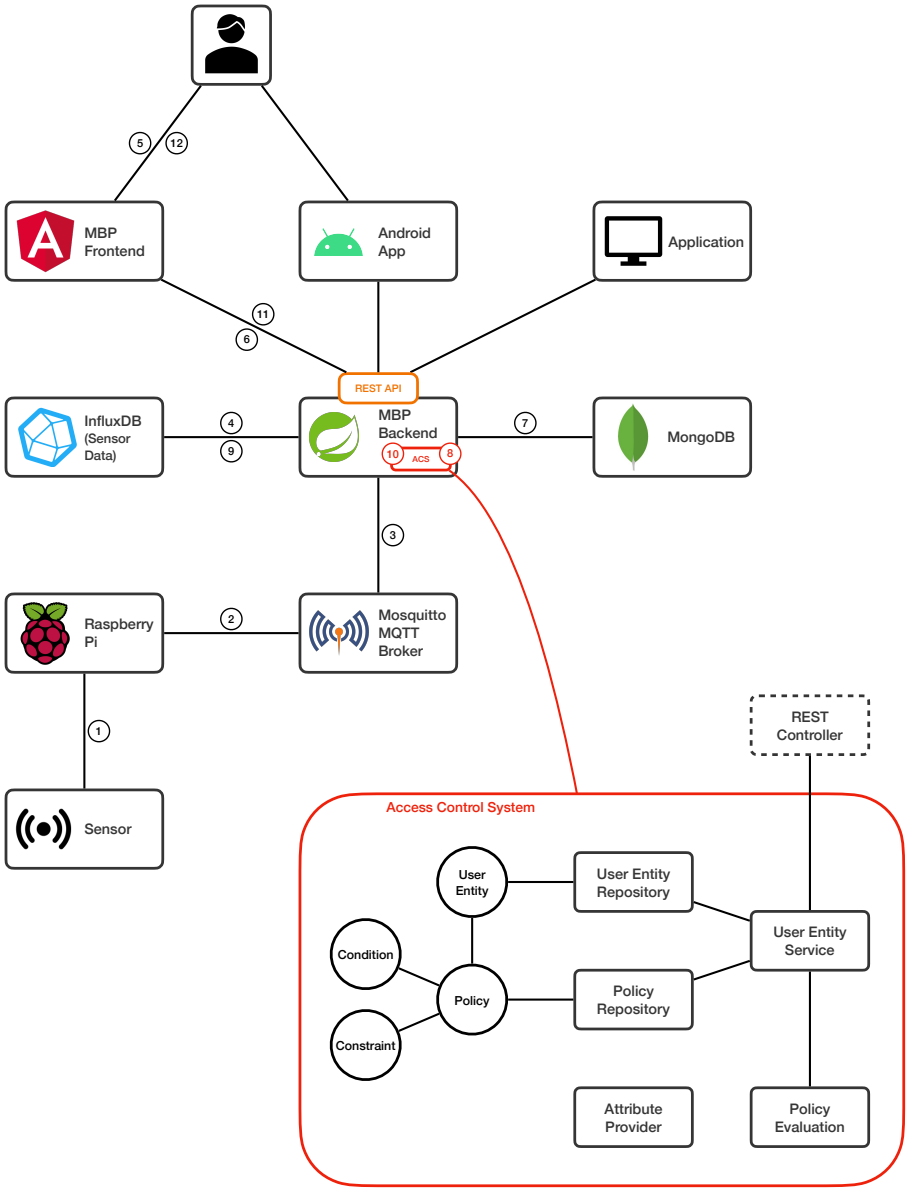

Figure 6.1: MBP Architecture and Access Control Integration

- A central `UserEntityService` is responsible for handling an access request. As a first step, the `UserEntityService` evaluates whether the requesting entity has admin privileges or is the owner of the requested entity. If not, the applicable policies for the requested entity and access type are retrieved from the MongoDB.
- An `AttributeProvider` provides attributes for both the requesting and requested entity. Within the scope of the prototypical implementation, this is limited to static attributes of entities, i.e., mostly metadata of users.

- The `PolicyEvaluationService` receives the requested entity, access type, and applicable policies from the `UserEntityService`. Since the policies are not prioritized, they are simply sorted by the number of conditions they contain. This way, the probability of having to evaluate less policies on average is increased. The policies are then evaluated based on the attributes provided by the `AttributeProvider` and the attributes obtained from the access request.
- `Policy`, `Condition`, and `Constraint` domain objects represent policies that include access type(s), condition(s) and (optional) constraint(s).
- A user interface provides means to manage policies, conditions, constraints and the mapping to `UserEntities`, which is built into the MBP web frontend. The specification of policies, conditions, and constraints is discussed in the next section.

6.3.2 Policy and Condition Specification

End-users can specify policies, conditions, and constraints through the MBP web frontend. Policies, conditions, and constraints are created individually, thereby enabling them to be reused and to be combined in any possible way. Furthermore, policies are defined as standalone entities and can be applied to, e.g., devices or sensors later on. This way, one policy can be used for several entities which saves a lot of configuration overhead for the end-user. In addition to that, policies can be specified for one or more access types which allows fine-grained access control and also enables filtering the policies specified for an entity based on the requested access type, which reduces the overhead for handling an access request introduced by access control. Note that policies are not prioritized in order to keep the policies as simple as possible for the end-users. For the specification of conditions, the MBP web frontend provides a graphical query editor that enables easy condition building with a clear-cut structure. Figure 6.2 shows an example of a condition built with the

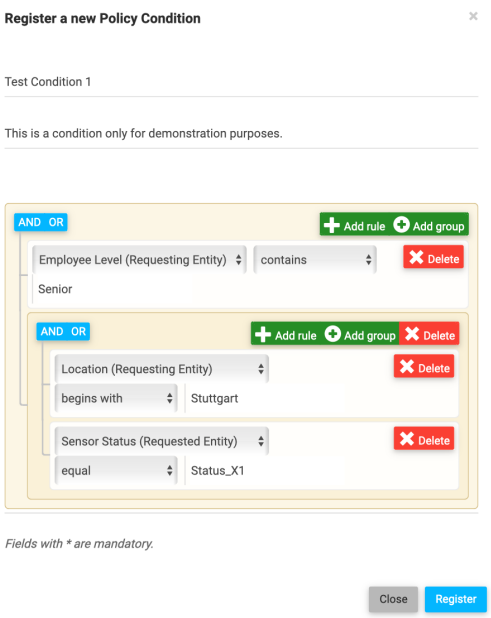

Figure 6.2: MBP Condition Builder

MBP condition builder. The condition builder is based on the jQuery QueryBuilder [1]. Users or applications that interact with the MBP backend via the exposed REST API can either use the format of the jQuery QueryBuilder or use the domain model of the MBP access control system.

6.3.3 Evaluation Approach

Since in the MBP every entity is associated with an owning entity, the generic evaluation algorithm (cf. Listing 5.1) is slightly modified in order to incorporate the owner of a requested entity into the evaluation process. Furthermore, the MBP distinguishes between admin and non-admin users. Therefore, the admin status is used in the evaluation algorithm. The actual evaluation functionality for conditions is implemented directly with each condition in order to easily introduce new types of conditions and decrease the amount of refactorings required to do so.

```
Input: Requested entity e, Requesting entity r, Context of requesting entity context, access type
Output: Access decision

if r.isAdmin == True then
       return AccessDecision(GRANTED, e)
else if e.owner == r then
       return AccessDecision(GRANTED, e)
else
       policies ← getPolicies(e, access_type)
       policies ← sort(policies, #conditions)
       for Policy p in policies as p_loop do
              for Condition c in p.conditions do
                     if not c.evaluate(context) then
                            continue p_loop
                     end if
              end for
              if p.constraint then
                     e ← applyConstraint(e, p.constraint)
              end if
              return AccessDecision(GRANTED, e)
       end for
end if
return AccessDecision(DENIED)
```

Algorithm 6.1: Evaluation algorithm for access requests

Listing 6.1 shows the modified evaluation algorithm for the MBP access control system. In contrast to the generic version from the concept, it checks whether the requesting entity has admin privileges or is the owner of the requested entity. If so, access is granted immediately regardless of any policy. Otherwise, the policies are sorted by the number of conditions and evaluated. Since a policy might contain more than one condition, these top-level conditions all have to hold in order for the policy to hold, which is implemented by the fail-fast approach in the inner condition loop.

[1] jQuery QueryBuilder Homepage

7 Evaluation

In this chapter, we evaluate the access control concept for IoT environments presented in Chapter 5, from now on simply referred to as the concept. Therefore, we evaluate whether the requirements specified in Chapter 4 have been fulfilled. Finally, the limitations of the concept are discussed.

7.1 Requirements Verification

This section evaluates whether the concept meets each functional and non-functional requirement. Therefore, the prototypical implementation of the access control concept for the MBP presented in Chapter 6 is also used to prove that certain requirements have been fulfilled.

7.1.1 Verification of Functional Requirements

We cover each functional requirement in the order they were presented in Table 4.1. For a detailed explanation and the rationale behind the functional requirements see Section 4.2. FR1 relates to the owner-relationship between users and requested entites from the edge or perception layer (cf. Figure 4.1). This relationship is explicitly mentioned in the domain model of the concept as the „owns“-relation between a requesting and a requested entity (cf. Figure 5.4). The privacy-by-default approach from FR2 is enabled by the whitelist-based approach of policy specification (cf. Section 5.1.5). FR3 is related to the specification of conditions under which requesting entities should be granted access to requested entities. This is implemented through policies and conditions as described in Section 5.2. The incorporation of contextual information of both the requesting and requested entities into the decision-making process as mentioned in FR3.1 is realized by using the attribute-based approach (cf. Section 3.2 and Section 5.1.5). From a conceptual point-of-view, the attribute-based approach allows to include any type of contextual information. FR3.2 requires the possibility to constrain the data returned to the requesting entity in terms of completeness or accuary. With the element of constraints, which can be included in policies, this can be achieved. The concept suggests two types of constraints – filtering and modifying constraints (cf. Figure 5.4). The concept does not provide any details on how to design or implement the user interface for access control configuration as mentioned in FR4, due to the generic design of the concept and the fact, that the concept has been designed primarily for (existing) IoT platforms which usually already ship with a frontend. However, the prototypical implementation of the access control system in the MBP demonstrates how to incorporate the configuration functionality into an existing frontend (cf. Section 6.3.2). FR5 relates to the probably most important part of an access control system for IoT environments – the actual enforcement of privacy policies. In Section 5.1.3 and Section 5.1.4, the workflow and the different components of the access control system have been discussed.

7.1.2 Verification of Non-Functional Requirements

In Section 4.3, the non-functional requirements for an access control system in IoT environments have been discussed. In the following, we evaluate whether each of them is fulfilled by design (of the concept) or at least can be achieved with a suitable implementation of the concept.
The first non-functional requirement (NFR1) is concerned with the expressiveness of the access control (domain) model. As discussed in the previous section, context-awareness and fine-granular policy/condition specification is enabled by the attribute-based access control model. With attributes, any kind of contextual information can be used. Moreover, the number of different attributes is virtually unlimited, which allows a very high degree of differentiation in terms of specifying policies and conditions. Furthermore, policies are prioritized and can be restricted to certain access types and access purposes. Conditions can be of different forms and can also be combined to logical expressions with any number of nested conditions. Additionally, the concept suggests using more sophisticated conditions, such as situation conditions, which can be implemented using situation recognition technology. A data-centric permission model is enabled by allowing data to be specified as a requested entity. The aspect of data minimization is supported by using constraints, which have been discussed in the previous section.
NFR2 deals with the aspect of continuous evaluation and enforcement of access requests. Due to the request-based approach of access request evaluation it is ensured, that every request is evaluated individually and access is never granted based on previous evaluation results. Therefore, up-to-date contextual information is required. This information is contained in the access request and supplemented by the attribute provider, which is invoked for each request, respectively.
The aspect of transparency required by NFR3 is handled by the access type and access purpose abstraction in the access control domain model. This way, users can restrict access to their data with regard to requesting entities, for which they do not know how their data is used.
NFR4 covers the requirement of extendability, which is achieved by both the generic design of the domain model with its many abstractions and the loosely coupled design of the access control components with regard to the existing IoT platform (cf. Figure 5.3). For instance, additional types of entities, conditions, or constraints can be easily added just as well as additional components.
NFR5 discusses the need for an efficient access control evaluation process, which is especially important when employing a distributed architecture (cf. Section 5.1.2). The whitelist-based evaluation approach in combination with the prioritized policies enables fast evaluation of policies and thereby reduces the increase in response times by a minimum. Moreover, the design of the access control components allows horizontal scaling by design, since it is completely stateless.
Lastly, NFR6 is about the usability of the access control system. The specification of (complex) conditions can be facilitated by providing a graphical conditions builder, that visualizes, e.g., composite conditions, as demonstrated by the prototypical implementation of the access control system in the MBP (cf. Figure 6.2). In addition to that, users do not have to specify policies separately for each entity they own, but rather can specify policies as individual entities and assign them to entities as needed. The same applies for conditions and constraints which also can be specified as standalone entities and then be used in different policies. Therefore, also non-expert users are able to easily configure their privacy settings.

7.2 Limitations

In order to complete the evaluation of the concept, this section discusses the limitations of the concept.

Verification of Attributes Requesting entities can send their contextual information along with the access request. However, the attribute provider might not be able to retrieve the same set of attributes as in the request. Therefore, some attributes might not get verified. One solution would be to define for each attribute, whether it has to be verified by comparing it to the one provided by the attribute provider or not. Another solution could be to define whether certain entities are trusted or not. Both of these solutions could be combined and outsourced into an (external) verification facility. Nonetheless, the system would increase in terms of complexity, regarding both the implementation and the configuration overhead for end-users.

Attribute Discovery For requesting entities, it is important to know which attributes are supported or required in order to possibly get access to a requested entity. The concept currently does not mention any explicit solution for this problem. One approach is to provide information about required and supported attributes via an exposed API endpoint that corresponds to the actual endpoint of the requested entity. This way, requesting entities could use a preflight request to determine which attributes need to be included in the access requested. In addition to that, the access control system could implement a mechanism, that includes missing attributes in the access decision response.

Derivation Transparency Modeling the relationship between two types of data currently is not supported (cf. Section 4.3.3). However, due to the whitelist-based approach of specifying policies, this is not required, since access to any entity is denied by default (privacy-by-default).

Difference to Traditional Models The attribute-based access control model significantly differs from traditional access control models, such as the role-based model or simple access control lists (cf. Section 3). Hence, users might experience difficulties when having to specify policies and conditions based on attributes.

Security Since the concept developed in this work purely focuses on privacy and access control (cf. Section 5.1.1), it heavily relies on existing security mechanisms in the target environment. Therefore, the concept is only suitable for target environments, where appropriate security measures have been implemented.

8 Conclusion & Future Work

With the IoT becoming more and more present in out everyday life, the need for privacy preserving mechanisms arises, due to the large amount of sensitive data that is captured by devices in IoT environments. More importantly, these mechanisms must be able to keep up with upcoming trends and technologies. The goal of this work is to create a generic concept that can be used to introduce privacy and access control in IoT environments. Therefore, the related work in the field of access control models is evaluated extensively especially focusing on suitability for the resource-constrained and ever-changing nature of IoT environments. In combination with an end-to-end reference architecture for IoT environments, we discuss both the functional and non-functional requirements towards an access control system for IoT environments, which include expressiveness in terms of specifying fine-granular conditions and constraints, adaptability in terms of different approaches of integration, and usability regarding the administration of access control settings for end-users. Therefore, we propose a generic and extendable concept for access control systems in IoT environments based on the attribute-based access control model and a request-based enforcement approach. Besides different options for integrating the access control system into existing IoT environments, the access control workflow and the components of the access control system are discussed. Furthermore, a generic domain model is provided, that demonstrates how to model policies, conditions, and constraints as well as their relationship to resources in the IoT environment. To showcase the concept, it has been prototypically implemented for the MBP IoT Platform.

As discussed in Section 7.2, there are some limitations to the concept. Future work will have to address the verification of attributes, especially if sent by the requesting entity, the discovery of supported attributes, and even more fine-granular abstractions of requested entities, such as data of a single type from a device, that provides multiple types of data. Furthermore, Stach and Mitschang [SM19] propose the concept of derivation transparency, which aims at automatically restricting access to data, that can be derived from other restricted data. This could be achieved by adding a relationship between two requested entities (data) to the domain model. However, whether such relationships can be detected automatically or have to specified by the end-user will have to be addressed by future work. Finally, it should be evaluated, how the concept presented in this work can be combined with security measures in order to provide one solution for security and privacy, that can be used in different IoT environments.

Bibliography

[AD06] A. Abou El Kalam, Y. Deswarte. "Multi-OrBAC: A new access control model for distributed, heterogeneous and collaborative systems". In: *IEEE Symp. on Systems and Information Security (SSI 2006), Sao Paulo, Brazil*. 2006 (cit. on p. 32).

[AEH18] A. Alkhresheh, K. Elgazzar, H. S. Hassanein. "Context-aware Automatic Access Policy Specification for IoT Environments". In: *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*. June 2018, pp. 793–799. DOI: 10.1109/IWCMC.2018.8450323 (cit. on pp. 26–28).

[AEH19] A. Alkhresheh, K. Elgazzar, H. S. Hassanein. "CAPE: Continuous Access Policy Enforcement for IoT Deployments". In: *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*. June 2019, pp. 1576–1581. DOI: 10.1109/IWCMC.2019.8766772 (cit. on p. 28).

[AIM10] L. Atzori, A. Iera, G. Morabito. "The Internet of Things: A survey". In: *Computer Networks* 54.15 (2010), pp. 2787–2805. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2010.05.010. URL: http://www.sciencedirect.com/science/article/pii/S1389128610001568 (cit. on p. 17).

[AK05] A. E. Abdallah, E. J. Khayat. "A Formal Model for Parameterized Role-Based Access Control". In: *Formal Aspects in Security and Trust*. Ed. by T. Dimitrakos, F. Martinelli. Boston, MA: Springer US, 2005, pp. 233–246. ISBN: 978-0-387-24098-5 (cit. on p. 27).

[AMP+13] B. Anggorojati, P. N. Mahalle, N. R. Prasad, R. Prasad, F. Theoleyre, A. Pang. "Secure access control and authority delegation based on capability and context awareness for federated iot". In: *Internet of Things and M2M Communications*. River Publisher, 2013, pp. 135–160 (cit. on p. 31).

[AMPP12] B. Anggorojati, P. N. Mahalle, N. R. Prasad, R. Prasad. "Capability-based access control delegation model on the federated IoT network". In: *The 15th International Symposium on Wireless Personal Multimedia Communications*. Sept. 2012, pp. 604–608 (cit. on p. 31).

[And20] Android. *Android Logo*. 2020. URL: https://source.android.com/setup/images/Android_symbol_green_RGB.png.

[Ang20] Angular. *Angular Logo*. 2020. URL: https://angular.io/presskit.

[AS02] M. A. Al-Kahtani, R. Sandhu. "A model for attribute-based user-role assignment". In: *18th Annual Computer Security Applications Conference, 2002. Proceedings.* Dec. 2002, pp. 353–362. DOI: 10.1109/CSAC.2002.1176307 (cit. on p. 27).

[AS11] A. Almutairi, F. Siewe. “CA-UCON: A Context-Aware Usage Control Model”. In: *Proceedings of the 5th ACM International Workshop on Context-Awareness for Self-Managing Systems*. CASEMANS ’11. Beijing, China: Association for Computing Machinery, 2011, pp. 38–43. ISBN: 9781450308779. DOI: 10.1145/2036146.2036153. URL: https://doi.org/10.1145/2036146.2036153 (cit. on p. 31).

[BAAO15] I. Bouij - Pasquier, A. Ait Ouahman, A. Abou El Kalam, M. Ouabiba de Montfort. “SmartOrBAC security and privacy in the Internet of Things”. In: *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*. Nov. 2015, pp. 1–8. DOI: 10.1109/AICCSA.2015.7507098 (cit. on pp. 32, 36).

[BHS16] J. Bernal Bernabe, J. L. Hernandez Ramos, A. F. Skarmeta Gomez. “TACIoT: multidimensional trust-aware access control system for the Internet of Things”. In: *Soft Computing* 20.5 (2016), pp. 1763–1779. ISSN: 1433-7479. URL: https://doi.org/10.1007/s00500-015-1705-6 (cit. on p. 31).

[BLM+11] A. B. Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, C. Dixon. “Home Automation in the Wild: Challenges and Opportunities”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 2115–2124. ISBN: 9781450302289. DOI: 10.1145/1978942.1979249. URL: https://doi.org/10.1145/1978942.1979249 (cit. on p. 15).

[BMA15] E. Barka, S. S. Mathew, Y. Atif. “Securing the Web of Things with Role-Based Access Control”. In: *Codes, Cryptology, and Information Security*. Ed. by S. El Hajji, A. Nitaj, C. Carlet, E. M. Souidi. Cham: Springer International Publishing, 2015, pp. 14–26. ISBN: 978-3-319-18681-8. DOI: https://link.springer.com/chapter/10.1007%2F978-3-319-18681-8_2 (cit. on p. 24).

[BYG+14] G. Bai, L. Yan, L. Gu, Y. Guo, X. Chen. “Context-aware usage control for web of things”. In: *Security and Communication Networks* 7.12 (2014), pp. 2696–2712. DOI: 10.1002/sec.424. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sec.424. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.424 (cit. on p. 31).

[CCT07] L. Cirio, I. F. Cruz, R. Tamassia. “A Role and Attribute Based Access Control System Using Semantic Web Technologies”. In: *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*. Ed. by R. Meersman, Z. Tari, P. Herrero. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1256–1266. ISBN: 978-3-540-76890-6 (cit. on p. 27).

[CD+99] J. Clark, S. DeRose, et al. *XML path language (XPath)*. 1999 (cit. on p. 27).

[CGLO08] I. F. Cruz, R. Gjomemo, B. Lin, M. Orsini. “A Location Aware Role and Attribute Based Access Control System”. In: *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS ’08. Irvine, California: Association for Computing Machinery, 2008. ISBN: 9781605583235. DOI: 10.1145/1463434.1463530. URL: https://doi.org/10.1145/1463434.1463530 (cit. on p. 27).

[CM12] G. Cugola, A. Margara. “Processing Flows of Information: From Data Stream to Complex Event Processing”. In: *ACM Comput. Surv.* 44.3 (June 2012). ISSN: 0360-0300. DOI: 10.1145/2187671.2187677. URL: https://doi.org/10.1145/2187671.2187677 (cit. on p. 19).

[CPP20] A. Chatterjee, Y. Pitroda, M. Parmar. “Dynamic Role-Based Access Control for Decentralized Applications”. In: (Feb. 13, 2020). arXiv: 2002.05547v2 [cs.CR] (cit. on p. 25).

[CW13] E. Coyne, T. R. Weil. “ABAC and RBAC: Scalable, Flexible, and Auditable Access Management”. In: *IT Professional* 15.3 (May 2013), pp. 14–16. ISSN: 1941-045X. DOI: 10.1109/MITP.2013.37 (cit. on p. 24).

[CZH+19] F. Cai, N. Zhu, J. He, P. Mu, W. Li, Y. Yu. “Survey of access control models and technologies for cloud computing”. In: *Cluster Computing* 22.3 (2019), pp. 6111–6122. ISSN: 1573-7543. URL: https://doi.org/10.1007/s10586-018-1850-7 (cit. on p. 32).

[DHYJ12] N. Dan, S. Hua-Ji, C. Yuan, G. Jia-Hu. “Attribute Based Access Control (ABAC)-Based Cross-Domain Access Control in Service-Oriented Architecture (SOA)”. In: *2012 International Conference on Computer Science and Service System*. Aug. 2012, pp. 1405–1408. DOI: 10.1109/CSSS.2012.354 (cit. on p. 27).

[DTS+16] N. Davies, N. Taft, M. Satyanarayanan, S. Clinch, B. Amos. “Privacy Mediators: Helping IoT Cross the Chasm”. In: *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*. HotMobile ’16. St. Augustine, Florida, USA: Association for Computing Machinery, 2016, pp. 39–44. ISBN: 9781450341455. DOI: 10.1145/2873587.2873600. URL: https://doi.org/10.1145/2873587.2873600 (cit. on p. 20).

[DV66] J. B. Dennis, E. C. Van Horn. “Programming Semantics for Multiprogrammed Computations”. In: *Commun. ACM* 9.3 (Mar. 1966), pp. 143–155. ISSN: 0001-0782. DOI: 10.1145/365230.365252. URL: https://doi.org/10.1145/365230.365252 (cit. on p. 31).

[DWHY18] Y. Dong, K. Wan, X. Huang, Y. Yue. “Contexts-States-Aware Access Control for Internet of Things”. In: *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*. May 2018, pp. 666–671. DOI: 10.1109/CSCWD.2018.8465364 (cit. on p. 27).

[FCF+09] A. Ferreira, D. Chadwick, P. Farinha, R. Correia, G. Zao, R. Chilro, L. Antunes. “How to Securely Break into RBAC: The BTG-RBAC Model”. In: *2009 Annual Computer Security Applications Conference*. Dec. 2009, pp. 23–31. DOI: 10.1109/ACSAC.2009.12 (cit. on p. 25).

[FHPM18] A. C. Franco da Silva, P. Hirmer, R. K. Peres, B. Mitschang. “An Approach for CEP Query Shipping to Support Distributed IoT Environments”. In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. Mar. 2018, pp. 247–252. DOI: 10.1109/PERCOMW.2018.8480241 (cit. on p. 19).

[FHS+20] A. C. Franco da Silva, P. Hirmer, J. Schneider, S. Ulusal, M. Tavares Frigo. "MBP – Not Just an IoT Platform". In: *Proceedings of the 18th Annual IEEE Intl. Conference on Pervasive Computing and Communications Demonstrations*. 2020 (cit. on pp. 18, 19).

[FKC03] D. Ferraiolo, D. R. Kuhn, R. Chandramouli. *Role-based access control*. Artech House, 2003. URL: https://books.google.de/books?hl=en&lr=&id=48AeIhQLWckC&oi=fnd&pg=PR15&dq=arXiv:0903.2171&ots=LMZEFItSC7&sig=SNTWFS6I5jJ18SsGyZxD4uctpuk&redir_esc=y#v=onepage&q&f=false (cit. on p. 24).

[FSG+01] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli. "Proposed NIST Standard for Role-Based Access Control". In: *ACM Trans. Inf. Syst. Secur.* 4.3 (Aug. 2001), pp. 224–274. ISSN: 1094-9224. DOI: 10.1145/501978.501980. URL: https://doi.org/10.1145/501978.501980 (cit. on p. 24).

[GBF+16] J. Guth, U. Breitenbücher, M. Falkenthal, F. Leymann, L. Reinfurt. "Comparison of IoT platform architectures: A field study based on a reference architecture". In: *2016 Cloudification of the Internet of Things (CIoT)*. Nov. 2016, pp. 1–6. DOI: 10.1109/CIOT.2016.7872918 (cit. on pp. 17, 18, 33, 41).

[GO04] M. Ge, S. L. Osborn. "A Design for Parameterized Roles". In: *Research Directions in Data and Applications Security XVIII*. Ed. by C. Farkas, P. Samarati. Boston, MA: Springer US, 2004, pp. 251–264. ISBN: 978-1-4020-8128-6 (cit. on p. 27).

[Gon+89] L. Gong et al. "A Secure Identity-Based Capability System." In: *IEEE symposium on security and privacy*. 1989, pp. 56–63 (cit. on p. 31).

[GPR13] S. Gusmeroli, S. Piccione, D. Rotondi. "A capability-based security approach to manage access control in the Internet of Things". In: *Mathematical and Computer Modelling* 58.5 (2013). The Measurement of Undesirable Outputs: Models Development and Empirical Analyses and Advances in mobile, ubiquitous and cognitive computing, pp. 1189–1205. ISSN: 0895-7177. DOI: https://doi.org/10.1016/j.mcm.2013.02.006. URL: http://www.sciencedirect.com/science/article/pii/S089571771300054X (cit. on p. 31).

[GW11] Z. Guoping, G. Wentao. "The research of access control based on UCON in the internet of things". In: *Journal of Software* 6.4 (2011), pp. 724–731 (cit. on p. 31).

[HBS+16] P. Hirmer, U. Breitenbücher, A. C. F. da Silva, K. Képes, B. Mitschang, M. Wieland. "Automating the Provisioning and Configuration of Devices in the Internet of Things". Englisch. In: *Complex Systems Informatics and Modeling Quarterly* 9 (Dezember 2016), pp. 28–43. ISSN: 2255 - 9922. DOI: 10.7250/csimq.2016-9.02. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=ART-2016-23&engl=0 (cit. on p. 19).

[HFK+13] V. C. Hu, D. Ferraiolo, R. Kuhn, A. R. Friedman, A. J. Lang, M. M. Cogdell, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, et al. "Guide to attribute based access control (abac) definition and considerations (draft)". In: *NIST special publication* 800.162 (2013). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.298.3381&rep=rep1&type=pdf (cit. on p. 25).

[HHM+17] M. Hüffmeyer, P. Hirmer, B. Mitschang, U. Schreier, M. Wieland. "SitAC – A System for Situation-aware Access Control - Controlling Access to Sensor Data". In: Jan. 2017, pp. 113–125. DOI: 10.5220/0006186501130125 (cit. on p. 29).

[HJMS13a] J. L. Hernández-Ramos, A. J. Jara, L. Marin, A. F. Skarmeta. "Distributed capability-based access control for the internet of things". In: *Journal of Internet Services and Information Security (JISIS)* 3.3/4 (2013), pp. 1–16 (cit. on p. 31).

[HJMS13b] J. Hernández-Ramos, A. J. Jara, L. Marín, A. Skarmeta. "Distributed Capability-Based Access Control for the Internet of Things". In: Oct. 2013 (cit. on p. 31).

[HKFV15] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, J. Voas. "Attribute-Based Access Control". In: *Computer* 48.2 (Feb. 2015), pp. 85–88. ISSN: 1558-0814. DOI: 10.1109/MC.2015.33 (cit. on p. 26).

[HKR09] P. Hitzler, M. Krotzsch, S. Rudolph. *Foundations of semantic web technologies*. CRC press, 2009. URL: https://books.google.de/books?hl=en&lr=&id=BdzL24RqcGIC&oi=fnd&pg=PP1&dq=semantic%20web%20technologies&ots=EgGMPucA-I&sig=K0ylEbKCzonQzR6hjM7sR5nglEs&redir_esc=y#v=onepage&q=semantic%20web%20technologies&f=false (cit. on p. 27).

[HNBH12] J. Huang, D. M. Nicol, R. Bobba, J. H. Huh. "A Framework Integrating Attribute-Based Policies into Role-Based Access Control". In: *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*. SACMAT '12. Newark, New Jersey, USA: Association for Computing Machinery, 2012, pp. 187–196. ISBN: 9781450312950. DOI: 10.1145/2295136.2295170. URL: https://doi.org/10.1145/2295136.2295170 (cit. on p. 27).

[HS15] M. Hüffmeyer, U. Schreier. "An attribute based access control model for RESTful services". In: (2015) (cit. on p. 28).

[HS16a] M. Hüffmeyer, U. Schreier. "Analysis of an Access Control System for RESTful Services". In: *Web Engineering*. Ed. by A. Bozzon, P. Cudre-Maroux, C. Pautasso. Cham: Springer International Publishing, 2016, pp. 373–380. ISBN: 978-3-319-38791-8 (cit. on p. 29).

[HS16b] M. Hüffmeyer, U. Schreier. "Formal Comparison of an Attribute Based Access Control Language for RESTful Services with XACML". In: *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*. SACMAT '16. Shanghai, China: Association for Computing Machinery, 2016, pp. 171–178. ISBN: 9781450338028. DOI: 10.1145/2914642.2914663. URL: https://doi.org/10.1145/2914642.2914663 (cit. on p. 29).

[HS16c] M. Hüffmeyer, U. Schreier. "RestACL: An Access Control Language for RESTful Services". In: *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*. ABAC '16. New Orleans, Louisiana, USA: Association for Computing Machinery, 2016, pp. 58–67. ISBN: 9781450340793. DOI: 10.1145/2875491.2875494. URL: https://doi.org/10.1145/2875491.2875494 (cit. on pp. 28, 45, 46).

[Hüf19] M. Hüffmeyer. *Effiziente Gestaltung und Anwendung von attributbasierter Zugriffskontrolle für RESTful Services*. de. 2019. DOI: 10.18419/OPUS-10788 (cit. on p. 28).

[HWBM16a] P. Hirmer, M. Wieland, U. Breitenbücher, B. Mitschang. "Automated Sensor Registration, Binding and Sensor Data Provisioning". Englisch. In: *Proceedings of the CAiSE'16 Forum, at the 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016)*. Vol. 1612. CEUR Workshop Proceedings. Ljubljana, Slovenia: CEUR-WS.org, June 2016, pp. 81–88. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2016-22&engl=0 (cit. on p. 19).

[HWBM16b] P. Hirmer, M. Wieland, U. Breitenbücher, B. Mitschang. "Dynamic Ontology-Based Sensor Binding". In: *Advances in Databases and Information Systems*. Ed. by J. Pokorný, M. Ivanović, B. Thalheim, P. Šaloun. Cham: Springer International Publishing, 2016, pp. 323–337. ISBN: 978-3-319-44039-2 (cit. on p. 19).

[HWS+15] P. Hirmer, M. Wieland, H. Schwarz, B. Mitschang, U. Breitenbücher, F. Leymann. "SitRS-a situation recognition service based on modeling and executing situation templates". In: *Proceedings of the 9th symposium and summer school on service-oriented computing*. 2015, pp. 113–127 (cit. on p. 29).

[Inf20] InfluxDB. *InfluxDB Logo*. 2020. URL: https://twitter.com/influxdb.

[ISO96] ISO. *Security frameworks for open systems: Access control framework, Technical Report ISO/IEC 10181-3*. 1996 (cit. on p. 24).

[JF06] P. Jin, Y. Fang-chun. "Description Logic Modeling of Temporal Attribute-Based Access Control". In: *2006 First International Conference on Communications and Electronics*. Oct. 2006, pp. 414–418. DOI: 10.1109/CCE.2006.350888 (cit. on p. 27).

[JXC12] J. Jindou, Q. Xiaofeng, C. Cheng. "Access Control Method for Web of Things Based on Role and SNS". In: *2012 IEEE 12th International Conference on Computer and Information Technology*. Oct. 2012, pp. 316–321. DOI: 10.1109/CIT.2012.81 (cit. on p. 25).

[KAPI18] M. Karimibiuki, E. Aggarwal, K. Pattabiraman, A. Ivanov. "DynPolAC: Dynamic Policy-Based Access Control for IoT Systems". In: *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*. Dec. 2018, pp. 161–170. DOI: 10.1109/PRDC.2018.00027 (cit. on p. 26).

[KBB+03] A. A. E. Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel, G. Trouessin. "Organization based access control". In: *Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks*. June 2003, pp. 120–131. DOI: 10.1109/POLICY.2003.1206966 (cit. on p. 32).

[KDBK09] A. [E. Kalam], Y. Deswarte, A. Baïna, M. Kaâniche. "PolyOrBAC: A security framework for Critical Infrastructures". In: *International Journal of Critical Infrastructure Protection* 2.4 (2009), pp. 154–169. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2009.08.005. URL: http://www.sciencedirect.com/science/article/pii/S1874548209000262 (cit. on p. 32).

[KMJ+05] Y.-G. Kim, C.-J. Mon, D. Jeong, J.-O. Lee, C.-Y. Song, D.-K. Baik. "Context-Aware Access Control Mechanism for Ubiquitous Applications". In: *Advances in Web Intelligence*. Ed. by P. S. Szczepaniak, J. Kacprzyk, A. Niewiadomski. Berlin,

Heidelberg: Springer Berlin Heidelberg, 2005, pp. 236–242. ISBN: 978-3-540-31900-9. URL: https://link.springer.com/chapter/10.1007/11495772_37 (cit. on p. 25).

[KT08] D. Kulkarni, A. Tripathi. "Context-Aware Role-Based Access Control in Pervasive Computing Systems". In: *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*. SACMAT '08. Estes Park, CO, USA: Association for Computing Machinery, 2008, pp. 113–122. ISBN: 9781605581293. DOI: 10.1145/1377836.1377854. URL: https://doi.org/10.1145/1377836.1377854 (cit. on p. 25).

[MAPP+13] P. N. Mahalle, B. Anggorojati, N. R. Prasad, R. Prasad, et al. "Identity authentication and capability based access control (iacac) for the internet of things". In: *Journal of Cyber Security and Mobility* 1.4 (2013), pp. 309–348 (cit. on p. 31).

[MASM13] S. S. Mathew, Y. Atif, Q. Z. Sheng, Z. Maamar. "The Web of Things - Challenges and Enabling Technologies". In: *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence*. Ed. by N. Bessis, F. Xhafa, D. Varvarigou, R. Hill, M. Li. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–23. ISBN: 978-3-642-34952-2. DOI: 10.1007/978-3-642-34952-2_1. URL: https://doi.org/10.1007/978-3-642-34952-2_1 (cit. on p. 24).

[Mon20] MongoDB. *MongoDB Logo*. 2020. URL: https://www.mongodb.com/brand-resources.

[Mos20] Mosquitto. *Mosquitto MQTT Broker Logo*. 2020. URL: https://www.pinclipart.com/pindetail/iRJRmJb_more-informations-on-https-mosquitto-mqtt-broker-clipart/.

[MXCM14] H. A. Maw, H. Xiao, B. Christianson, J. A. Malcolm. "An evaluation of break-the-glass access control model for medical data in wireless sensor networks". In: *2014 IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom)*. Oct. 2014, pp. 130–135. DOI: 10.1109/HealthCom.2014.7001829 (cit. on p. 25).

[MXCM16] H. A. Maw, H. Xiao, B. Christianson, J. A. Malcolm. "BTG-AC: Break-the-Glass Access Control Model for Medical Data in Wireless Sensor Networks". In: *IEEE Journal of Biomedical and Health Informatics* 20.3 (May 2016), pp. 763–774. ISSN: 2168-2208. DOI: 10.1109/JBHI.2015.2510403 (cit. on p. 25).

[MYAZ15] R. Mahmoud, T. Yousuf, F. Aloul, I. Zualkernan. "Internet of things (IoT) security: Current status, challenges and prospective measures". In: *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. Dec. 2015, pp. 336–341. DOI: 10.1109/ICITST.2015.7412116 (cit. on p. 20).

[OMA15] A. Ouaddah, H. Mousannif, A. Ait Ouahman. "Access control models in IoT: The road ahead". In: *2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA)*. Nov. 2015, pp. 1–2. DOI: 10.1109/AICCSA.2015.7507090 (cit. on pp. 15, 38).

[OMEO17] A. Ouaddah, H. Mousannif, A. [Elkalam], A. [Ouahman]. "Access control in the Internet of Things: Big challenges and new opportunities". In: *Computer Networks* 112 (2017), pp. 237–262. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2016.11.007. URL: http://www.sciencedirect.com/science/article/pii/S1389128616303735 (cit. on pp. 15, 24, 25, 31, 32, 36).

[Par03] J. Park. "Usage control: a unified framework for next generation access control". In: (2003) (cit. on p. 30).

[PDY+17] P. Pappachan, M. Degeling, R. Yus, A. Das, S. Bhagavatula, W. Melicher, P. E. Naeini, S. Zhang, L. Bauer, A. Kobsa, S. Mehrotra, N. Sadeh, N. Venkatasubramanian. "Towards Privacy-Aware Smart Buildings: Capturing, Communicating, and Enforcing Privacy Policies and Preferences". In: *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. June 2017, pp. 193–198. DOI: 10.1109/ICDCSW.2017.52 (cit. on p. 20).

[PI20] R. PI. *Raspberry Pi Logo*. 2020. URL: https://www.raspberrypi.org/files/Raspberry_Pi_Logo.zip.

[PS02] J. Park, R. Sandhu. "Towards Usage Control Models: Beyond Traditional Access Control". In: *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*. SACMAT '02. Monterey, California, USA: Association for Computing Machinery, 2002, pp. 57–64. ISBN: 1581134967. DOI: 10.1145/507711.507722. URL: https://doi.org/10.1145/507711.507722 (cit. on p. 30).

[PS04] J. Park, R. Sandhu. "The UCONABC Usage Control Model". In: *ACM Trans. Inf. Syst. Secur.* 7.1 (Feb. 2004), pp. 128–174. ISSN: 1094-9224. DOI: 10.1145/984334.984339. URL: https://doi.org/10.1145/984334.984339 (cit. on p. 30).

[Ray18] P. Ray. "A survey on Internet of Things architectures". In: *Journal of King Saud University - Computer and Information Sciences* 30.3 (2018), pp. 291–319. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2016.10.003. URL: http://www.sciencedirect.com/science/article/pii/S1319157816300799 (cit. on p. 17).

[RJK15] Q. M. Rajpoot, C. D. Jensen, R. Krishnan. "Integrating Attributes into Role-Based Access Control". In: *Data and Applications Security and Privacy XXIX*. Ed. by P. Samarati. Cham: Springer International Publishing, 2015, pp. 242–249. ISBN: 978-3-319-20810-7. URL: https://link.springer.com/chapter/10.1007/978-3-319-20810-7_17 (cit. on p. 24).

[SCFY96] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman. "Role-based access control models". In: *Computer* 29.2 (Feb. 1996), pp. 38–47. ISSN: 1558-0814. DOI: 10.1109/2.485845 (cit. on pp. 24, 25).

[SGM20] C. Stach, C. Gritti, B. Mitschang. "Bringing Privacy Control Back to Citizens: DISPEL — a Distributed Privacy Management Platform for the Internet of Things". In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. SAC '20. Brno, Czech Republic: Association for Computing Machinery, 2020, pp. 1272–1279. ISBN: 9781450368667. DOI: 10.1145/3341105.3375754. URL: https://doi.org/10.1145/3341105.3375754 (cit. on pp. 30, 33, 34, 36, 37, 46).

[SH06] H. Shen, F. Hong. "An Attribute-Based Access Control Model for Web Services". In: *2006 Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06)*. Dec. 2006, pp. 74–79. DOI: 10.1109/PDCAT.2006.28 (cit. on p. 26).

[She09] H. Shen. "A Semantic-Aware Attribute-Based Access Control Model for Web Services". In: *Algorithms and Architectures for Parallel Processing*. Ed. by A. Hua, S.-L. Chang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 693–703. ISBN: 978-3-642-03095-6 (cit. on p. 27).

[SHWM16] A. C. F. da Silva, P. Hirmer, M. Wieland, B. Mitschang. "SitRS XT-towards near real time situation recognition". In: *Journal of Information and Data Management* 7.1 (2016), pp. 4–4 (cit. on p. 29).

[SM19] C. Stach, B. Mitschang. "Elicitation of Privacy Requirements for the Internet of Things Using ACCESSORS". In: *Information Systems Security and Privacy*. Ed. by P. Mori, S. Furnell, O. Camp. Cham: Springer International Publishing, 2019, pp. 40–65. ISBN: 978-3-030-25109-3. URL: https://link.springer.com/chapter/10.1007/978-3-030-25109-3_3 (cit. on pp. 15, 29, 36, 37, 61).

[SNCC18] A. R. Sfar, E. Natalizio, Y. Challal, Z. Chtourou. "A roadmap for security challenges in the Internet of Things". In: *Digital Communications and Networks* 4.2 (2018), pp. 118–137. ISSN: 2352-8648. DOI: https://doi.org/10.1016/j.dcan.2017.04.003. URL: http://www.sciencedirect.com/science/article/pii/S2352864817300214 (cit. on p. 20).

[SO17] D. Servos, S. L. Osborn. "Current Research and Open Problems in Attribute-Based Access Control". In: *ACM Comput. Surv.* 49.4 (Jan. 2017). ISSN: 0360-0300. DOI: 10.1145/3007204. URL: https://doi.org/10.1145/3007204 (cit. on pp. 27, 32).

[Spr20] Spring. *Spring Boot Logo*. 2020. URL: https://github.com/topics/spring-boot.

[SRGC15] S. Sicari, A. Rizzardi, L. Grieco, A. Coen-Porisini. "Security, privacy and trust in Internet of Things: The road ahead". In: *Computer Networks* 76 (2015), pp. 146–164. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2014.11.008. URL: http://www.sciencedirect.com/science/article/pii/S1389128614003971 (cit. on p. 15).

[SSXL09] J. Shu, L. Shi, B. Xia, L. Liu. "Study on Action and Attribute-Based Access Control Model for Web Services". In: *2009 Second International Symposium on Information Science and Engineering*. Dec. 2009, pp. 213–216. DOI: 10.1109/ISISE.2009.80 (cit. on p. 27).

[Sta05] O. Standard. *extensible access control markup language (xacml) version 3.0*. 2005 (cit. on pp. 26, 27).

[SWW15] A. Sadeghi, C. Wachsmann, M. Waidner. "Security and privacy challenges in industrial Internet of Things". In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. June 2015, pp. 1–6. DOI: 10.1145/2744769.2747942. (cit. on p. 21).

[TJA10] H. Takabi, J. B. D. Joshi, G. Ahn. "Security and Privacy Challenges in Cloud Computing Environments". In: *IEEE Security Privacy* 8.6 (Nov. 2010), pp. 24–31. ISSN: 1558-4046. DOI: 10.1109/MSP.2010.186 (cit. on p. 24).

[W3C20] W. W. W. C. (W3C). *Web of Things (WoT) Thing Description*. Apr. 2020. DOI: https://www.w3.org/TR/wot-thing-description/. URL: https://www.w3.org/TR/wot-thing-description/ (cit. on p. 24).

[WM11] M. E. Whitman, H. J. Mattord. *Principles of information security*. Cengage Learning, 2011 (cit. on p. 20).

[WO11] H. Wang, S. Osborn. "Static and Dynamic Delegation in the Role Graph Model". In: *IEEE Transactions on Knowledge and Data Engineering* 23.10 (Oct. 2011), pp. 1569–1582. ISSN: 1558-2191. DOI: 10.1109/TKDE.2010.205 (cit. on p. 27).

[Woh19] M. Wohlfarth. "Ein Sicherheitskonzept für IoT-Plattformen". Deutsch. Masterarbeit. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, Dezember 2019, p. 72. URL: http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=MSTR-2019-88&engl=0.

[WSBL15] M. Wieland, H. Schwarz, U. Breitenbücher, F. Leymann. "Towards situation-aware adaptive workflows: SitOPT — A general purpose situation-aware workflow management system". In: *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. Mar. 2015, pp. 32–37. DOI: 10.1109/PERCOMW.2015.7133989 (cit. on p. 29).

[YT05] E. Yuan, J. Tong. "Attributed based access control (ABAC) for Web services". In: *IEEE International Conference on Web Services (ICWS'05)*. July 2005, p. 569. DOI: 10.1109/ICWS.2005.25 (cit. on p. 26).

[ZACF18] S. Zheng, N. Apthorpe, M. Chetty, N. Feamster. "User Perceptions of Smart Home IoT Privacy". In: *Proc. ACM Hum.-Comput. Interact.* 2.CSCW (Nov. 2018). DOI: 10.1145/3274469. URL: https://doi.org/10.1145/3274469 (cit. on p. 20).

[ZCDV17] J. Zhou, Z. Cao, X. Dong, A. V. Vasilakos. "Security and Privacy for Cloud-Based IoT: Challenges". In: *IEEE Communications Magazine* 55.1 (Jan. 2017), pp. 26–33. ISSN: 1558-1896. DOI: 10.1109/MCOM.2017.1600363CM (cit. on p. 15).

[ZPSP05] X. Zhang, F. Parisi-Presicce, R. Sandhu, J. Park. "Formal Model and Policy Specification of Usage Control". In: *ACM Trans. Inf. Syst. Secur.* 8.4 (Nov. 2005), pp. 351–387. ISSN: 1094-9224. DOI: 10.1145/1108906.1108908. URL: https://doi.org/10.1145/1108906.1108908 (cit. on p. 30).

[ZT10] G. Zhang, J. Tian. "An extended role based access control model for the Internet of Things". In: *2010 International Conference on Information, Networking and Automation (ICINA)*. Vol. 1. Oct. 2010, pp. V1-319-V1-323. DOI: 10.1109/ICINA.2010.5636381 (cit. on p. 24).

[Zue10] D. Zuehlke. "SmartFactory—Towards a factory-of-things". In: *Annual Reviews in Control* 34.1 (2010), pp. 129–138. ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2010.02.008. URL: http://www.sciencedirect.com/science/article/pii/S1367578810000143 (cit. on p. 17).

[ZWWL14] Y. S. Zhang, M. F. Wu, L. Wu, Y. Y. Li. "Attribute-Based Access Control Security Model in Service-Oriented Computing". In: *Proceedings of the 2012 International Conference on Cybernetics and Informatics*. Ed. by S. Zhong. New York, NY: Springer New York, 2014, pp. 1473–1479. ISBN: 978-1-4614-3872-4 (cit. on p. 27).

All links were last followed on October 31, 2020.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature