How to Realize Device Interoperability and Information Security in mHealth Applications

Christoph Stach, Frank Steimle, and Bernhard Mitschang

Institute for Parallel and Distributed Systems, University of Stuttgart, Universitätsstraße 38, D-70569 Stuttgart, Germany
{stachch,steimlfk,mitsch}@ipvs.uni-stuttgart.de

Abstract. More and more people suffer from chronic diseases such as the chronic obstructive pulmonary disease (COPD). This leads to very high treatment costs every year, as such patients require a periodic screening of their condition. However, many of these checks can be performed at home by the patients themselves. This enables physicians to focus on actual emergencies. Modern smart devices such as Smartphones contribute to the success of these telemedical approaches. So-called mHealth apps combine the usability and versatility of Smartphones with the high accuracy and reliability of medical devices for home use. However, patients often face the problem of how to connect medical devices to their Smartphones (the **device interoperability** problem). Moreover, many patients reject mHealth apps due to the lack of control over their sensitive health data (the **information security** problem).
In our work, we discuss the usage of the **P**rivacy **M**anagement **P**latform (*PMP*) to solve these problems. So, we describe the structure of mHealth apps and present a real-world COPD application. From this application we derive relevant functions of an mHealth app, in which device interoperability or information security is an issue. We extend the PMP in order to provide support for these recurring functions. Finally, we evaluate the utility of these PMP extensions based on the real-world mHealth app.

Keywords: mHealth · Device Interoperability · Information Security · COPD.

1 Introduction

Due to long stand-by times and multiple built-in sensors, the Smartphone became our ubiquitous companion. New use cases are constantly emerging. Especially in the health sector, the use of Smartphones can be highly beneficial to save treatment costs and help patients who cannot visit their physicians regularly [33]. The usability of Smartphone apps in the health sector—the so-called mHealth apps—is limitless. There is an app for almost any situation [32].

While there are some mHealth apps for medical reference (that is, apps providing information about diseases) as well as hospital workflow management apps (i. e., apps supporting physicians in their everyday duties), mHealth apps are mainly from the health management domain [23]. The latter includes cardio fitness, medication adherence, and chronic disease management. To this end, these apps support two essential features: self-observation and feedback [21]. That is, the patient performs health measurements instructed by the app, transmits the measured values to the app and the app performs analyses on these values. Based on the results, the app gives the user medical recommendations.

Since mHealth apps involve the patient actively into the treatment and monitoring process, s/he gets more aware of his or her condition. So, mHealth apps change the physician patient relationship especially for patients with a chronic disease. These patients have to visit their physician periodically in order to check certain health values. However, such a metering can be performed by the patients autonomously, if they receive a proper guidance tailored to their know-how. That is, with the help of an mHealth app they are able to do the monitoring in a telemedical manner. This take the load off both, patients as well as physicians. Patients benefit from the freedom to do the metering at any arbitrary place and time while physicians are able to concentrate on emergencies. This is not just a huge saving potential but also improves the quality of healthcare at the same time [30].

To capture health data, mHealth apps make use of various sensors. In addition to the already broad spectrum of sensors built into modern Smartphones that can be used for healthcare (e. g., a heart rate sensor, a camera, or a microphone), even medical devices for home-use can be connected to a Smartphone. The *Vitalograph copd-6 bt* is an example for such medical device.

Yet, the data interchange between medical devices and Smartphones often fails because of non-uniform communication protocols. The **device interoperability** of Smartphones and medical devices is a key challenge for the success of mHealth apps [6]. Also the assurance of **information security** is vital for mHealth, as patients have to trust their apps [1]. Thus, we address these two challenges in our work. Therefore, we come up with a concept for an enabler for device interoperability and information security in mHealth apps. To achieve this objective, we proceed as follows: (*I*) We analyze a real-world mHealth app regarding the collected data and used devices. (*II*) We deduce a generic data model for mHealth apps. Due to this data model, our approach is applicable for any kind of mHealth app. (*III*) We identify five recurring tasks within mHealth apps for which device interoperability and information security are key requirements, namely login, metering, localization, data storage and analytics, and data transmission. For each of these tasks, we introduce an extension for the **P**rivacy **M**anagement **P**latform (*PMP*) [37], to ensure interoperability and information security. (*IV*) We use these extensions to revise the analyzed mHealth app in terms of sensor support and the patients' privacy. With this revised app, we assess the practical effect of our approach.

This paper is the extended and revised version of the paper entitled "The Privacy Management Platform: An Enabler for Device Interoperability and Information Security in mHealth Applications" [41] presented at the 11th International Conference on Health Informatics (HEALTHINF) 2018.

The remainder of this paper is as follows: The layer architecture of mHealth systems is introduced in Sect. 2. Section 3 discusses a real-world application scenario from the mHealth domain, namely *ChronicOnline*, an mHealth app for COPD patients. Then, Sect. 4 takes a look at related work concerning connection techniques used by medical device as well as information security mechanisms in mHealth apps. Based on these findings, we introduce a generic interchange data model for mHealth apps and induct briefly in the PMP, the foundation platform for our solution approach in Sect. 5. Then, Sect. 6 details on the key components of our approach and demonstrates their applicability, using the example of a COPD app. In Sect. 7 we assess how our approach contributes to solve information security and interoperability problems for mHealth apps. Section 8 concludes this work and gives an outlook on some future work.

2 Design of an mHealth System

Typically, mHealth systems consist of three layers [18]. The *Sensor Layer* manages the access to any sensor required by mHealth apps, i. e., it collects the sensors' raw data and provides it to subsequent layers. In the *Smartphone Layer*, this data is collected, assessed, and processed. The processing step transforms the raw data to information. An optional analysis phase can derive events from the information, e. g., to detect a seizure automatically. The data can also be stored on the Smartphone, e. g., to monitor the progress of the disease. Additionally, the data is forwarded to on-line servers managed by the *Back-End Layer*. While the Smartphone Layer holds health data of a single patient, in the Back-End Layer the data of multiple patients is assembled (e. g., to derive new insights into the course of a disease or to prepare the data for subsequent in-depth analyses). Via this layer physicians are able to perform analyses and receive diagnosis support for each of their patients. For this purpose, there is sometimes an additional *Presentation Layer*. This fourth layer preprocesses the health data stored in the back-end and presents the relevant data in a user-friendly manner. So, physicians are able to interpret the results.

Figure 1 shows the interaction of these four layers. A metering device can be connected to a Smartphone via Bluetooth. A patient can install an mHealth app on his or her Smartphone and use the medical data recorded by the external metering device. This data can be enriched by data from the Smartphone's built-in sensors (e. g., location data). The app sends the gathered data to an mHealth back-end for thorough analyses and to store the data at a central repository. Physicians can access the repository via their diagnosis tools to find an adequate method of treatment.

Concerning the two key issues addressed in this paper, namely device interoperability and information security, several components have to be taken into consideration. Any data interchange between two layers is a problem, since there are heterogeneous interchange formats and multiple connection standard. This concerns especially the data interchange between Smartphones and medical devices, since these devices commonly define proprietary communication protocols.

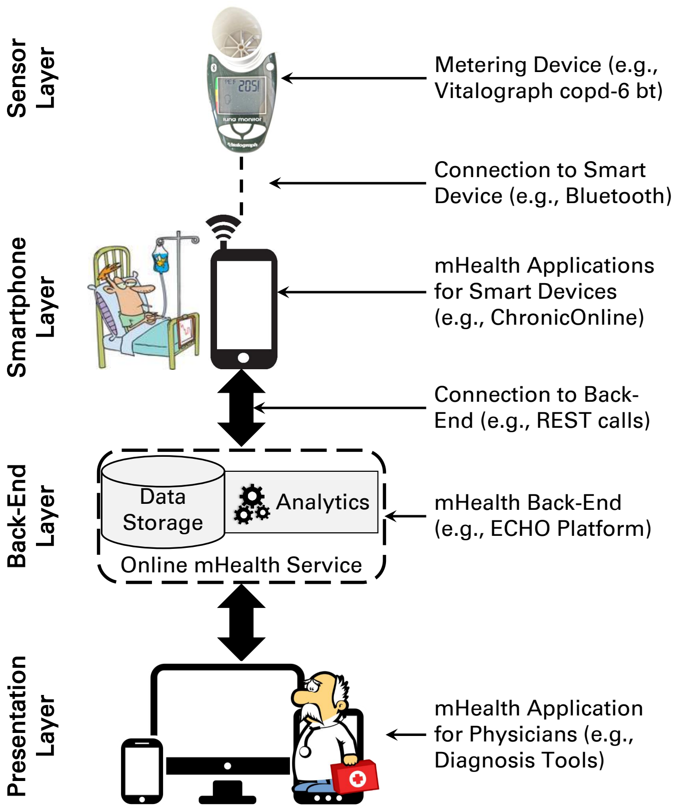

Sensor Layer
Smartphone Layer
Back-End Layer
Presentation Layer
Metering Device (e.g., Vitalograph copd-6 bt)
Connection to Smart Device (e.g., Bluetooth)
mHealth Applications for Smart Devices (e.g., ChronicOnline)
Connection to Back-End (e.g., REST calls)
Data Storage
Analytics
Online mHealth Service
mHealth Back-End (e.g., ECHO Platform)
mHealth Application for Physicians (e.g., Diagnosis Tools)

Fig. 1: Design of an mHealth System [based on 41].

Yet, the harmonization of the medical data formats used by mHealth apps and back-ends has to be considered as well. While there are several approaches towards a server-sided unified data model for health data (e. g., in the *HealthVault* [2]), there are no such approaches for apps.

Information security has to be considered at any layer. However, as users cannot influence the security mechanisms implemented in sensors, data protection has to be assured mainly on the Smartphone Layer. While there are approaches to protect sensitive data on the Back-End Layer (e. g., [15]), an adequate solution for the Smartphone Layer is missing. The Presentation Layer does not create any sensitive data.

We focus on the Sensor Layer and the Smartphone Layer in our work regarding device interoperability and information security, as there are solutions for the other layers to solve theses issues. Yet, the usability and security of an mHealth system is impaired by its weakest component. That is, the existing usability and security solutions for the Back-End Layer and the Presentation Layer are worthless with respect to the whole system, as long as there are no appropriate approaches for the Sensor Layer and the Smartphone Layer as well [34].

3 Application Scenario

The chronic obstructive pulmonary disease or short COPD is an obstructive lung disease. COPD patients suffer from a poor airflow which worsens over time. According to the World Health Organization, approximately 6% of all deaths in 2012 resulted directly from COPD [51]. Even though COPD is not curable, a fast and persistent therapy can slow the progression of the disease significantly down. For this purpose telemedicine is very appropriate, since the required measurements can be carried out very easily with affordable medical measuring devices [46].

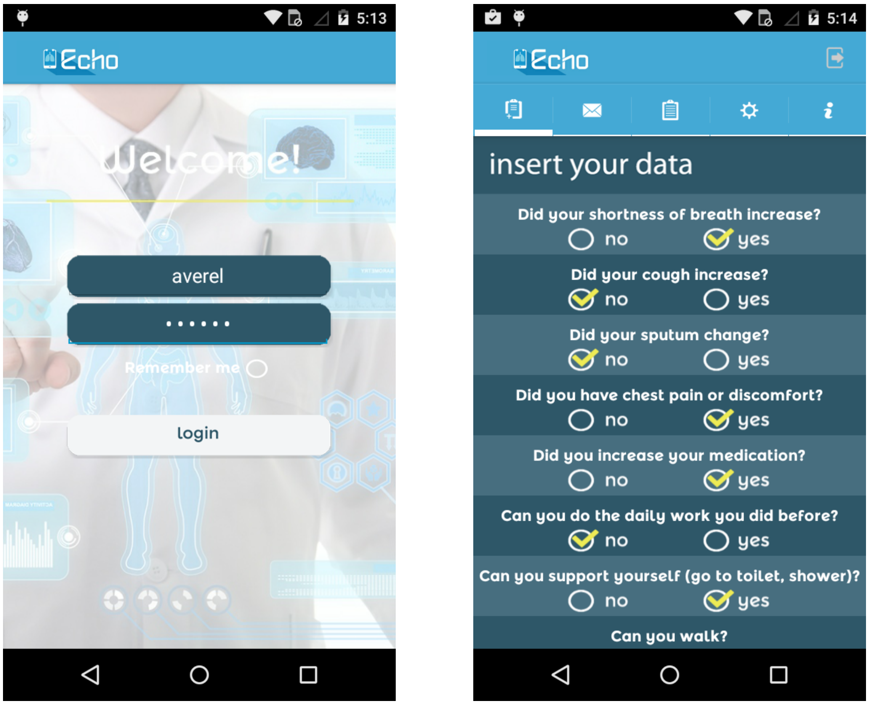
In the following, we introduce a real-world mHealth app for COPD patients which is based on the *ECHO* project[1]. This example covers all four mHealth layers introduced in the previous section: Patients get an app for Smartphones (see Sect. 3.2) which gathers various health data from medical devices (see Sect. 3.1). The app sends health data to a data analysis back-end and receives medical outcomes from it likewise. Physicians can also access their patients data via the back-end, e. g., via diagnosis tools running in the Presentation Layer (see Sect. 3.3).

3.1 The Vitalograph COPD-6 BT

The Vitalograph copd-6 bt[2] is used to screen the pulmonary function. So, patients with respiratory conditions can perform the metering by themselves. It records various lung function parameters including the Peak Expiratory Flow (PEF) and the Forced Expiratory Volume (FEV) among others. PEF and FEV are key readings for the diagnosis of COPD. The measurement results are transmitted via Bluetooth LE. As today's Smartphones support this battery-saving connection standard, the Vitalograph copd-6 bt provides a sound hardware foundation for telemedical mHealth apps.

For data interchange the proprietary *Terminal I/O* protocol [45] is used which operates on top of Bluetooth LE GATT [5, 11]. The basic idea behind this protocol is that a client (e. g., a Smartphone) has to request credits from the server (i. e., the Vitalograph copd-6 bt). Following this, the client can retrieve data from the server (e. g., health data); for each message the client has to pay one credit. Hereby it is ensured that the client is able to make a specific number of requests without having to wait for a response from the server. However, as soon as the client runs out of credits, it has to wait for the server's response in order to get additional credits. For more information on the Terminal I/O protocol, please refer to the literature [45]. The application of such proprietary protocols impedes the development of mHealth apps since only a limited number of device types supports the respective protocol. As a common communication standard is not in sight, app developers are in great need of other approaches with which the device interoperability is enhanced.

[1] see http://chroniconline.eu
[2] see https://vitalograph.com/product/162427



insert your data

Did your shortness of breath increase? no [✓ yes]

Did your cough increase? [✓ no] yes

Did your sputum change? [✓ no] yes

Did you have chest pain or discomfort? no [✓ yes]

Did you increase your medication? no [✓ yes]

Can you do the daily work you did before? [✓ no] yes

Can you support yourself (go to toilet, shower)? no [✓ yes]

Can you walk?

(a) Login Screen **(b)** Questionnaire

Fig. 2: The ChronicOnline App's Key Functionality [41].

3.2 The ChronicOnline App

mHealth apps are good for regularly recording various parameters but they are insufficient for a comprehensive COPD screening. For this reason, the University of Stuttgart, the University of Crete, and OpenIT launched the ECHO project in 2013. Within the scope of this project several mHealth apps collect various health data and gather the data in a Cloud infrastructure [3]. Online services are available for physicians enabling various data analytic functions and giving them an holistic overview of their patients' condition. In addition to it, the patients and the physicians remain in contact with each other whereby the physicians can give their patients advices by sending them messages via their mHealth app. The ChronicOnline app [4] is a mobile front-end for the ECHO project. Its key functions (access control and a COPD questionnaire) are shown in Fig. 2.

Initially, the user has to log in his or her account (see Fig. 2a). The app differentiates two user groups, patients who have only access to their personal account and physicians who monitor several patients. The login process as well as any other communication between the app and the ECHO back-end is realized by REST calls.

After authorization is complete, the patient has access to several tabs. The most significant tab is the questionnaire tab (see Fig. 2b). Here the user has to

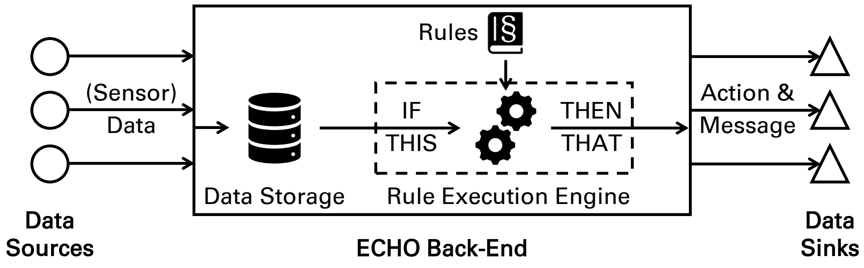


Fig. 3: The Operation Principle of the ECHO Rule Execution Engine.

answer five questions about his or her condition. Each question can be answered with 'yes' or 'no'. Depending on the given answers up to six subquestions appear to refine the medical finding. Afterwards, the results are transferred to the back-end as a JSON object. More details on the data format are given in Sect. 5.1.

This app is a representative sample for the innumerable COPD apps available in various app stores. We could use any of them without a loss of argument.

3.3 The ECHO Back-End

After the patient has submitted his or her answers, the back-end performs analyses and preprocesses the data for physicians. The incoming data—i. e., the answers to the questionnaire as well as health data from metering devices such as the Vitalograph copd-6 bt—is processed by the ECHO back-end using a rule execution engine. This strategy is loosely based on the *If This Then That* (*IFTTT*) approach[3]. That approach allows users to define the behavior of their IoT devices in a simple but yet highly versatile manner [22]. Via this so-called trigger-action approach, it is also possible for non-IT experts (e. g., physicians or medical experts) to configure the rule execution engine [47].

Figure 3 shows how the ECHO back-end operates. All incoming data from data sources such as apps (queries) or sensors (medical data) is initially stored as raw data for documentation and subsequent analyses. Then, the data is forwarded to the rule execution engine. All defined rules have a simple IFTTT-like structure:

$$Trigger \rightarrow Action \tag{1}$$

A trigger is a boolean expression describing which conditions have to apply in order to initiate a particular action. The action defines messages being sent to actuators, which carry out the actual action. The following example illustrates this: **IF** Questions 1 **AND** Question 3 have been answered with 'yes' **THEN** send inform the patient, that s/he should call his or her physician immediately. As an ECHO data processing rule, this would look like this:

$$Q_1 == true \land Q_3 == true \rightarrow send\_mail('Call\ physician') \tag{2}$$

[3] see https://ifttt.com

Since the results obtained from the questionnaire are only indicators for a possible aggravation, the discussion of the medical findings still has to take place in a personal meeting with the physician. If the app has also access to data from medical devices, the rules for the analysis could be adapted to to get more useful information about the patients' condition. In case of COPD, examples for important measurements to monitor the patients condition would be Peak Expiratory Flow (PEF) or the Forced Expiratory Volume (FEV). Using these values, it could be even possible to skip the personal meeting with the physician while getting a more detailed diagnosis of the patients health condition. This diagnosis could the be used to cope with the disease in a personalized manner. In order to enable the physician to create personalized rules for his or her patients, concepts for editing the rules for the analysis are needed. In the domain of manufacturing, such concepts are already being developed to promote digitalization [49]. The goal is to enable every worker in manufacturing environments to create easy-to-use rules similar to the ones described above to link events to corresponding actions [50]. These concepts can easily be transferred to mHealth apps, such that a physician can define monitoring rules for special groups of patients or even personalized rules for individual patients.

As the definition of the triggers gets complex when more data sources are involved, domain experts can define reusable patterns:

$$Critical\_Condition :== Q_1 == true \wedge Q_3 == true \tag{3}$$

Then, this pattern can be used in Eq. (2) in order to simplify the left part of the rule ($Critical\_Condition \rightarrow send\_mail('Call\ physician')$).

Obviously, the action can also be used to control apps on the Presentation Layer. Thereby history data from the back-end loaded into the app, e. g., to display former questionnaire answers. Detailed analysis results and history charts can also be preprocessed to include them in web-based dashboards for physicians [43]. Thus, we cover all four layers of an mHealth system with this example, starting with the Sensor Layer up to actuators, i.e., the Presentation Layer.

Please note that the back-end is out of this paper's scope. More information about the back-end can be found in [44]. However, we give a brief outlook about *PATRON*[4] (see [35, 36]), a data privacy mechanism for the processing of sensitive data in such a back-end in Sect. 8.

4 Related Work

In the context of device interoperability in mHealth apps, a lot of work is done concerning the back-end systems. For this purpose, these systems introduced harmonized data models for health data and provide generic interfaces so that any kind of mHealth app can use them to collect and share heath data. One of the biggest systems is the HealthVault [2]. It supports various health-related data types ranging from fitness data to entire personal health records. The

[4] PATRON is an acronym for "**P**rivacy in S**tr**eam Pr**o**cessi**n**g".

HealthVault acts as a middleman between the data producers—i. e., the mHealth apps—and the data consumers—i. e., analysis systems. Concerning app-sided device interoperability and information security, such a system provides no help. *Google Fit* [24] is another back-end for storing and processing health data. Google's system deals with fitness data (e. g., the heart rate), only. Google Fit provides interfaces for app developers which enhance the device interoperability and facilitate the reading of sensors in third-party devices—at least for devices that are supported by Google Wear[5]. As especially medical devices are not supported by Google Wear, [29] discuss, how Smartphones can be connected with these devices. However, their solution aims for physical connections and not for harmonized communication standards. [17] recommend to use IoT techniques to solve this problem. In their proposal all sensors are connected to the Internet and send their data to a Cloud-based database which is also accessible for mHealth apps. Even though there are secure transfer protocols for health data (e. g., [20]), a permanent and unrestricted transmission of such sensitive data to an unknown server is ineligible for most users. For that reason, [10] introduce an approach with which the patient stays in total control over his or her data; the mHealth app has full access to the health data while external services (e. g., apps for physicians) only get access as long as the patient grants it. In [27] an mHealth app is introduced which relies on a full encryption of the health data when the data is stored or transmitted. Yet, both approaches assume that only external entities constitute a threat for the security of health data. However, as studies prove, two out of three apps handle their data either carelessly or even maliciously [9]. Thus, none of these approaches solves the device interoperability and information security issues of mHealth apps. Despite the benefit of mHealth apps especially the device incompatibility and the mistrust in app developers repel patients from using such apps [28]. For this reason, we introduce a general approach dealing with both issues in the following sections.

5 Interoperability and Security Reflections

In order to come up with a design methodology for interoperable and secure mHealth apps, a secure data management, data input techniques, and defined data access conditions are required. Our solution to this is built upon the PMP. When looking at the ChronicOnline app, it does not take advantage of the full potential of the ECHO back-end and the prevailing hardware. With its analytic functions and notification services, the back-end is able to process more complex data such as location data or respiratory data in addition to the replies to the disease-specific questionnaire. By the integration of medical devices, the ChronicOnline app becomes a full-fledged telemedicine solution.

Therefore we extend the ChronicOnline app by adding location services[6]. Studies show that the location can have a relevant influence on the progression of a disease [16]. Moreover, we provide support for various third-party Bluetooth

[5] see https://www.android.com/wear

[6] Location based services in general constitute a severe threat to a user's privacy [19].

respiratory monitors. The measurement results are added to the electronic health record (see Sect. 5.1 for its data model) and transferred to the ECHO back-end. There, the data can be automatically pre-analyzed which not only unburdens the physicians in charge, but also results in a faster feedback for the patients. As a consequence this enhanced app has to deal with increasing interoperability and security issues.

```
DailyReport {
  patientId (integer): Unique Identifier of the Patient,
  recordId  (integer): Unique Identifier of this Record,
  date (string): Date of Report,
  q1  (boolean): Answer to Question 1,
  q1a (boolean): Answer to Question 1a,
  q1b (boolean): Answer to Question 1b,
  q1c (boolean): Answer to Question 1c,
  q2  (boolean): Answer to Question 2,
  ...
  q5  (boolean): Answer to Question 5
}
```

List. 1: The ChronicOnline Data Model (Excerpt) [based on 41].

Amulet, a tiny Smart Device operating as an information hub, tries to solve both problems [12]. It confirms the user's identity and identifies any available devices in the surrounding belonging to him or her. Then, Amulet ascertains that only trusted third-party devices can be used for the metering and ensures a secure connection to these devices. Moreover, Amulet provides mechanisms to protect the health data against external attackers. In order to transfer the data to (trusted) servers for further processing, Amulet is able to connect to a Smartphone and use it for transmission. Unfortunately, this approach has a severe drawback: The user has to possess another device in addition to his or her Smartphone and the actual medical device. This causes further costs and it is unpractical since the user has to carry the Amulet all the time. The PMP (see Sect. 5.2) is a middleware for application platforms which provides similar features.

5.1 The Internal Data Model

The ChronicOnline app sends up to eleven boolean values, representing the answers to the questionnaire, to the ECHO back-end. For this reason, its data model for the data exchange is quite plain (see List. 1). In addition to the questionnaire answers, a daily report entails an ID for the patient and for the report itself as well as the submission date. Please note that authorization data to confirm the identity of the submitter is not part of this data model. The authorization is managed via the authorization header of the HTTP protocol.

We add latitude and longitude to this basic schema to support location data as well as entries for the most relevant COPD readings, including the Peak Expiratory Flow and the Forced Expiratory Volume among others (see List. 2).

```
DailyReport {
  patientId (integer): Unique Identifier of the Patient,
  ...
  q5 (boolean): Answer to Question 5,
  lat (number): Latitude of GPS Location.,
  lon (number): Longitude of GPS Location,
  pef (integer): Peak Expiratory Flow,
  fev1 (number): Forced Expiratory Volume (First Second),
  fev10 (number): Forced Expiratory Volume (Ten Seconds),
  fef2575 (number): Mid-Breath Forced Expiratory Flow,
  ...
}
```

List. 2: The Extended Data Model (Excerpt) [based on 41].

Since the JSON format is well-suited for the data exchange between the app and the back-end, but not for the data processing within the app, we apply wrapper classes for the conversion of such JSON files to Java objects and vice versa.

5.2 Overview of the PMP

We use the PMP in order to realize interoperability and security features as needed by mHealth apps. We give a brief overview of the PMP at first and describe in detail the new components, which are developed in this work in the next section. The PMP is an intermediate layer between apps and the operating system. It prevents any (potentially malicious) app from accessing sensitive data.

When an app needs access to such data, it has to ask the PMP for permission. The PMP operates several data provisioning services, the so-called *Resources*. Each Resource is responsible for a certain type of data but it is not committed to a certain technology; e. g., the Location Resource is able to provide location data from a GPS provider or from a network provider. Thereby it can adapt its functionality to the available hardware. In addition to it, the user can define how accurate the data should be in order to obfuscate his or her private data. Further Resources can be hooked into the PMP at runtime need-based.

A Resource defines *Privacy Settings* which restrict the usage of the corresponding Resource. By default, there is a Privacy Setting for granting or permitting the usage of a Resource. Furthermore, a Privacy Setting can be more specific depending on the type of Resource. For instance, the Location Resource can reduce the location's accuracy.

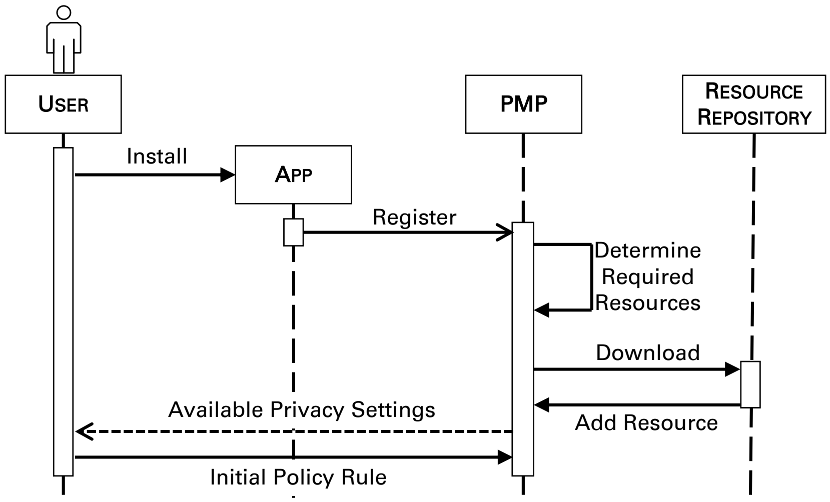


Fig. 4: The Registration of an App at the PMP [41].

At installation time an app has to register at the PMP. The PMP identifies which Resources are required and installs missing Resources if necessary. The user then postulates an initial policy rule for this app defining, which data should be given to the app and how accurate this data should be. This registration process is shown in Fig. 4.

Since the user is able to deny that an app gets access to a certain Resource, the app model of the PMP encapsulates logically coherent parts of the app in so-called *Service Features*. So, the withdrawal of access rights simply deactivates the affected Service Features but the app itself can still be executed. Moreover, the user can modify access rights of individual Service Features at runtime. The permission allocation is shown in Fig. 5.

The PMP is primary a fine-grained permission system with additional privacy features (e. g., data obfuscation). However, in the context of interoperability and uncertainty of available hardware, the PMP serves a dual purpose. Each Resource is abstracted from a certain technology and can have several implementations. As a consequence, the app developer only has to request a certain type of data (e. g., respiratory data) and the dedicated Resource ensures that it gets this data from the available hardware (e. g., a Vitalograph copd-6 bt). If no hardware providing this kind of data is available—which is similar to a user-defined prohibition to use the hardware via the PMP—then the app gets informed and adapts to this condition by deactivating affected Service Features.

The PMP is able to degrade an app's functionality when it cannot access all of its requested data. Certain Services Features can be deactivate instead of feeding an app with random data. Especially, in the context of mHealth apps, the usage of random medical values is inappropriate. For this reason, the PMP's data obfuscation for health data is severely restricted. For further details on the PMP, please refer to the literature [37, 38].

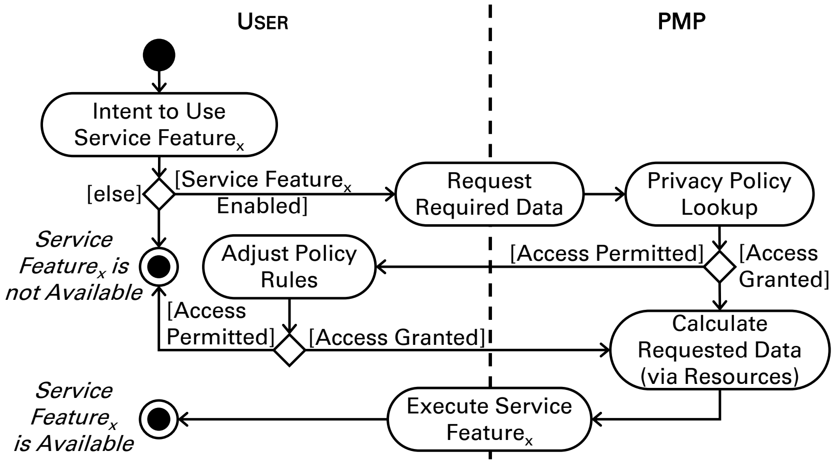

USER
PMP
Intent to Use Service Feature$_x$
[else]
[Service Feature$_x$ Enabled]
Request Required Data
Privacy Policy Lookup
Service Feature$_x$ is not Available
Adjust Policy Rules
[Access Permitted]
[Access Granted]
[Access Permitted]
[Access Granted]
Calculate Requested Data (via Resources)
Service Feature$_x$ is Available
Execute Service Feature$_x$

Fig. 5: The PMP's Permission Allocation Process [41].

6 Design of mHealth PMP Resources

In the following we introduce five PMP Resources which are developed for mHealth apps, namely a secure **authentication** Resource, a **metering** Resource, a **localization** Resource, a **data storage** Resource, and a **connection** Resource. In addition, a Resource for health data **encryption** is introduced. Finally, we revise the ChronicOnline app with the help of these Resources.

6.1 The Dialog Box Resource

In the ChronicOnline app a user has to enter credential information which is forwarded to an ECHO server for verification. While the back-end is operated by a trusted organization (e. g., a hospital), any developer can implement apps for ECHO. Thus the front-end is potentially insecure—yet the user has to reveal his or her login data to it.

In order to solve this problem, we introduce an isolated *Dialog Box Resource*. An app can invoke the dialog box and specify the displayed text as well as where the entered information should be forwarded to. The server's response is sent back to the invoking app. In this way, the dialog box is completely generic and can be used in any context where the user has to enter private data. The dialog box is executed as a part of the PMP and is completely isolated from the invoking app. No information is passed to the app except for the back-end's reply. The user cannot only grant or permit the usage of the dialog box, but also specify which back-ends are legit recipients.

6.2 The Metering Resource

One of the biggest problems for mHealth apps is the integration of third-party medical devices as there is currently no uniform standard for intercommunication with such devices. This is why most of the currently available apps support some hand-picked medical device only.

That is why we introduce a *Metering Resource* with a simple, yet generic interface (see List. 3). In order to support as many use-cases as possible, we designed a new data type for health data, called *HealthRecord*. A HealthRecord consists of an id, the answers to the questionnaire, the health data itself, location data, and a timestamp (see List. 2). The HealthRecord is able to maintain any kind of health data via the generic attributes that can be made up of any (i. e., unspecified or schema-less) JSON object. As a consequence, the Metering Resource defines no fixed schema for the health data, but processes this data as a JSON object. Currently, we adhere to the data schema which is defined by the particular data source and use the same schema for the JSON object. However, as soon as a common standard for electronic health records is defined, we can switch to this standard internally without having to revise any app using the Metering Resource.

```
interface IMetering {
  SealedHealthRecord performMetering();
}
```

List. 3: Interface of the Metering Resource.

Since most modern medical devices exchange their data with Smartphones via a Bluetooth connection, we focus our work on this kind of connection. As there is no common Bluetooth transmission protocol, the Metering Resource has to be able to support several protocols. Currently, we implemented interfaces for the ISO/IEEE 11073 Personal Health Data (PHD) standards [14] as well as the Terminal I/O protocol [45] of the Vitalograph Lung Monitor. Further protocols (e. g., Android Wear) or other connection standards (e. g., USB) can also be supported in the future due to the modular expandability of PMP Resources (see [42]). The Metering Resource defines no fixed schema for health data, but processes this data as a JSON object with an attribute for every measured value.

As the Metering Resource autonomously connects to any available device, an app developer simply has to request the health record from the Resource. However, concerning information security, the app should not be able to read the health record. Thus, the PMP encrypts the JSON object containing the health data. As a consequence, authorized Resources can process the health data while (potentially malicious) apps cannot access the data. Nevertheless, sometimes an app needs access to certain values of the health data, e. g., in order to display the data. For this purpose, the *Unsealer Resource* (see Sect. 6.6) can be used in order to decrypt certain excerpts of the record.

Moreover, the user is able to restrict which devices are allowed to provide health data for a certain app via a Privacy Setting of the Metering Resource. For instance, s/he can permit an app to use the Metering Resource, but only a

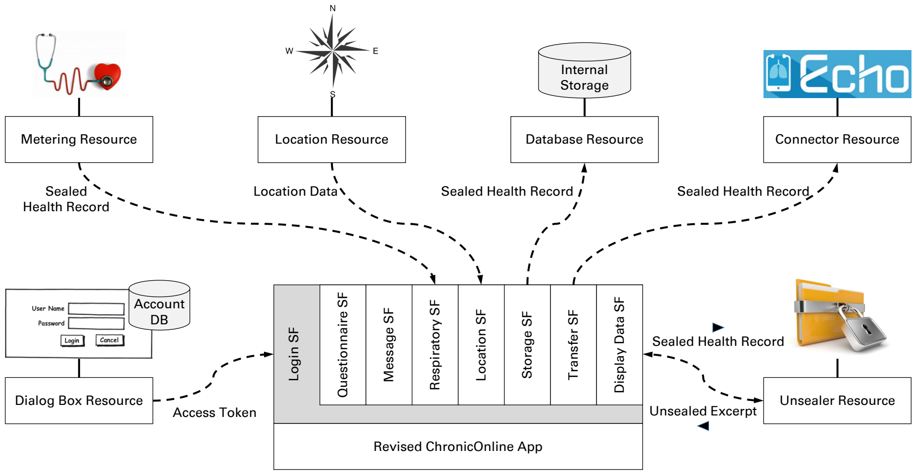


Fig. 6: Service Features of the COPD App and Their Applied PMP Resources [41].

certain kind of data from a specific smart watch such as the heart rate is sent back by the Resource. A manipulation or randomization of the health data for privacy reasons is disregarded, since this kind of data has to be as accurate as possible in order to be useful.

6.3 The Location Resource

Since it is relevant for some medical outcomes to know the location where a reading has been recorded in order to detect potential relationships between a patient's condition and his or her whereabouts, we provide a special *Location Resource* for mHealth apps. Android supports different location providers—mainly the *Network Location Provider* for cell tower and Wi-Fi based location and the more accurate *GPS Provider*—which periodically report on the geographical location of the device. Our Location Resource always requests the most accurate location provider which is currently available. Additionally, also indoor positioning can be realized, e. g., by using accelerometers and gyroscopes [13], barometric pressure sensors [53], or compass sensors [52]. All of these sensors are available in today's Smartphones.

Normally, in Android an app has to subscribe for location updates and the provider sends any update to the app during its lifetime. Especially for the GPS Provider this consumes a lot of energy. However, such a behavior is not necessary in the context of a metering app which requires location data only once after the metering is executed. As a consequence the Location Resource supports two modes of operation (see List. 4): On the one hand, an app can request a periodical location update for a certain time (from `startLocationLookup` to `endLocationLookup`), check for updates (`isUpdateAvailable`), and obtain location data (`getLocation`). On the other hand, an app can also request a singular location (`getSingularLocation`) or even add it directly to an existing

HealthRecord (`addPosition`). For the latter mode, the Resource arranges the subscription to and unsubscription from a location provider automatically—since a Resource can be used mutually by several apps, already existing results from another app's location request are utilized for this purpose.

```
interface ILocation {
  void startLocationLookup();
  void endLocationLookup();
  boolean isUpdateAvailable();
  Location getLocation();
  Location getSingularLocation();
  void addPosition(inout SealedHealthRecord record);
}
```

List. 4: Interface of the Location Resource.

In order to respect the user's privacy, the Location Resource provides a Privacy Setting to reduce the accuracy of the location data. The user is able to stipulate a maximal accuracy in meters. If the provided location data is more accurate then the Resource adds a random factor depending on the user settings to the latitude and longitude in order to reduce the accuracy. The user is also allowed to use completely random location data, so that his or her location remains unknown to the app.

6.4 The Secure Database Resource

Since Android stores its data in a clear-text readable form, attackers may harvest all stored data. The in Android 5.0 introduced file system encryption functions are no solution for this threat, since the encryption is not kept up during operation, i. e., as soon as the system is fully booted. So, sensitive data such as health data requires additional security features. For this very reason, additional security features are required in order to ensure information security for sensitive data. The **S**ecure **D**ata **C**ontainer (*SDC*) is a *Secure Database Resource* for the PMP [39]. It encrypts the stored data with AES-256 encryption and only the PMP has the key. The SDC has a fine-grained access control to share the stored data with selected apps. Performance measurements show that the thereby caused overhead is within reasonable limits.

We tailor the SDC slightly to our requirements: The database's internal data model is comparable to a key-value store. In accordance with *CURATOR* [40], the Secure Database Resource operates with JSON objects and adopts the therein defined keys and values directly. As several stored JSON objects can use a common key, an internal id is applied to each object. The database's primary key consists of this id and the key. The Secure Database Resource partitions the stored data in several SQLite tables internally (one per app) for performance and security reasons.

The Secure Database Resource's interface (see List. 5) is minimalistic, yet demand-actuated. A JSON object can be stored (`store`), obtained (`get`), and deleted (`delete`) via its id. In addition to it, the Secure Database Resource supports the usage of HealthRecords which can be passed as a total and the Resource arranges the decryption of the health data and the decomposition into key-value pairs. Obtaining complete HealthRecords operates analog.

```
interface IDatabase {
  int store(in JSONObject data);
  JSONObject get(in int id);
  boolean delete(in int id);
  boolean storeRecord(in SealedHealthRecord record);
  boolean storeAll(in List<SealedHealthRecord> records);
  SealedHealthRecord getRecord(in int id);
  List<SealedHealthRecord> getAllRecords();
}
```

List. 5: Interface of the Secure Database Resource.

The most important contribution of the Secure Database Resource concerning information security is indeed its full encryption. Shmueli et al. [31] describe various attack models against databases and assess different database encryption schemata that should prevent these attacks. The result of their study is quite simple: Only by encrypting the whole database as a total, information leakage as well as unauthorized modifications can be prevented reliably. Therefore, each SQLite file is encrypted in the Secure Database Resource with an AES-256 encryption and only the PMP has the key. This encryption causes an overhead. However, by the partitioning of the data this overhead is within reasonable limits and the fact that the Resource is meant for sensitive data such as health data, any overhead is justified. Moreover, modern Smartphones possess sufficient computing power, whereby this overhead becomes negligible.

6.5 The Connector Resource

Nowadays, most apps do not only operate locally on a single Smartphone, but include various external services. This is why almost every app requests the permission to establish a network connection. The user is not even informed about this request. However, an app having this permission is able to upload sensitive data to any server. To use external services in a secured way, we introduce the *Connector Resource.*

Within the Connector Resource various trusted external services can be included. An app is able to either upload data to or retrieve data from one of these services (e. g., the ECHO back-end or Amazon's SNS). Additional domain-specific information protection policies can be applied within the Connector Resource (such as the *Mobile access to Health Documents* profile [26] or the

Audit Trail and Node Authentication profile [25]), if they are supported by the back-end.

Currently, only the basic functionality for the ECHO back-end is supported. Hence, the current Resource's interface (see List. 6) comprises three functions: Analog to the Secure Database Resource, HealthRecords can be stored (`storeRecord`), obtained (`getRecord`), and deleted (`deleteRecord`). Certainly, this interface needs to be extended, when further services are added.

```
interface IConnector {
  boolean storeRecord(in SealedHealthRecord record);
  SealedHealthRecord getRecord(in int id);
  boolean deleteRecord(in int id);
}
```

List. 6: Interface of the Connector Resource.

Concerning interoperability, the Connector Resource handles any interface changes of the external services for the apps. I. e., adjustments have to be done only once at the Resource and not for every app. Additionally, it has protective function. Since an app can only pass sensitive data to the Connector Resource and has no further connectivity it is guaranteed that the data is only sent to the appointed service—only connections to trusted services are provided by the Resource. Additionally, the user can remove any service s/he does not trust via the Resource's Privacy Settings.

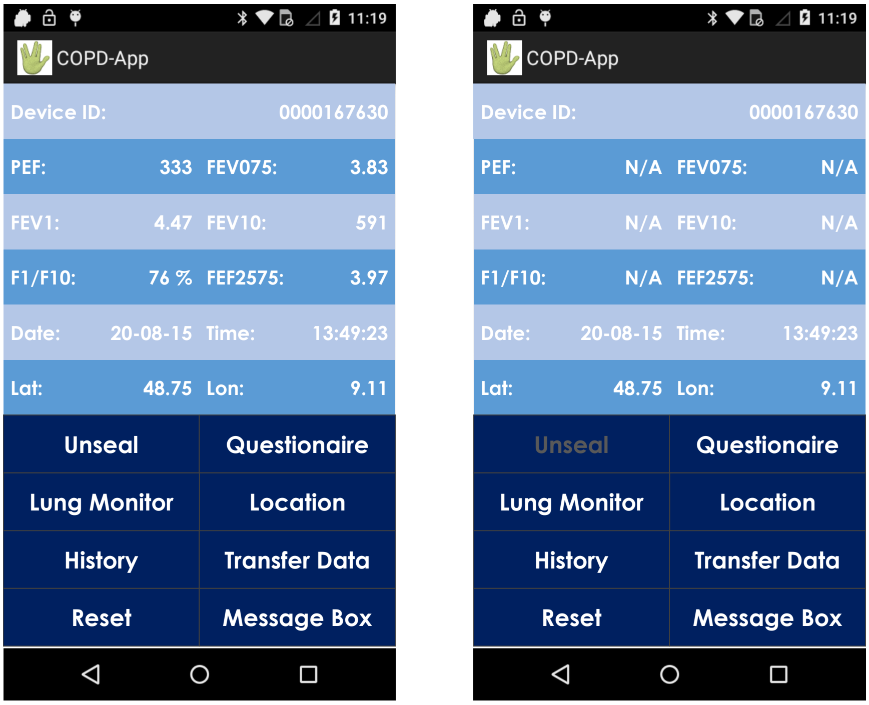
6.6 The Unsealer Resource

As mentioned above, the health data within the HealthRecord gets encrypted as soon as it is passed from any Resource to any app by the PMP. As a consequence, the app does not have access to this sensitive data. Basically, the health data is only processed in Resources and therefore the encryption represents no constraint for the user (cf. [48]). However, in rare cases it might be necessary for an app to get access to excerpts of the health data, e. g., to display it. For this very reason, we introduce the *Unsealer Resource*. Its interface consists of a single function which converts a SealedHealthRecord into a HealthRecord (see List. 7).

```
interface IUnsealer {
  HealthRecord unseal(in SealedHealthRecord sealedRecord);
}
```

List. 7: Interface of the Unsealer Resource.

However, the user is not only able to grant or permit the usage of the Unsealer Resource, but s/he can also define via a Privacy Setting which excerpt of the data



11:19
COPD-App

Device ID:			0000167630
PEF:	333	FEV075:	3.83
FEV1:	4.47	FEV10:	591
F1/F10:	76 %	FEF2575:	3.97
Date:	20-08-15	Time:	13:49:23
Lat:	48.75	Lon:	9.11

11:19
COPD-App

Device ID:			0000167630
PEF:	N/A	FEV075:	N/A
FEV1:	N/A	FEV10:	N/A
F1/F10:	N/A	FEF2575:	N/A
Date:	20-08-15	Time:	13:49:23
Lat:	48.75	Lon:	9.11

(a) Unsealed Data Screen **(b)** Sealed Data Screen

Fig. 7: The ChronicOnline Privacy-Driven App [41].

should be revealed. To that end, the Resource provides a multi-state selection (e. g., general information, lung-related content only, etc.) in order to facilitate the configuration. Since the health data is stored as a JSON object, any unrevealed data is simply cut out of it. For the user's safety also the Unsealer does not provide falsified values, as such an approach is disregarded for the processing of health data.

6.7 Revised ChronicOnline App

Based on these Resources, we revise the ChronicOnline app by including sensor data (e. g., respiratory meters or location sensors) while regarding device interoperability and information security.

After the user has answered the questionnaire, the revised app expects respiratory data from a connected device via the Metering Resource. The user only has to push the "*Lung Monitor*" button and the connection and data transfer is arranged automatically by the PMP. When the measured data is available, the results of the questionnaire are applied to the DailyReport (see List. 2) and the location data is inquired. Then, the complete DailyReport object is stored internally (e. g., if the data cannot be transmitted immediately) and is transferred subsequently to the ECHO back-end (see Fig. 7a). Please note, that for this capture any data is unsealed for demonstration purpose. The user is informed in each step about the collected data. So, s/he is in total control over the data. The effect of a deactivated Unsealer Resource is shown in Fig. 7b. Without the Unsealer Resource only data which are not related to any health issues are known to the app.

In order to react properly to the restriction of a Resource, an app needs to define Service Features. Figure 6 shows the 8 Service Features of the revised ChronicOnline App (denoted as SF) and which Resources are required for each of them. Not every Service Feature necessarily requires data from a Resource (e. g., the *Questionnaire SF*). The Service Features have a modular design and can be plugged in and out at runtime (e. g., when the corresponding Resource is deactivated). In the app's program flow these features are skipped. However, since the authentication of the user is mandatory, the *Login SF* and therewith the Dialog Box Resource cannot be deactivated or else any other Service Feature is also deactivated.

7 Assessment

Concerning information security, the literature speaks of 7 key protective goals, namely *auditability*, *authenticity*, *availability*, *confidentiality*, *integrity*, *non-repudiation*, and *privacy* [7, 8]. The original ChronicOnline app only fulfills the authenticity goal and the confidentiality goal directly due to its login mechanism and the auditability goal, the integrity goal, and the non-repudiation server-sided due to the security mechanism of the ECHO back-end. However, as soon as real health data is processed by the app, the user cannot rely on the prevailing mechanisms. Our revised app supports all 7 protective goals due to the used Resources. The auditability as well as the non-repudiation is guaranteed, since the PMP logs any Resource access of an app. The authenticity is given via the login mechanism and since the login data is not shared with the app, it cannot commit an identity theft. The availability is given, as all data is stored on the device using mature database technologies. The confidentiality is ensured, since any app functionality is only usable after the login process is completed. The integrity is guaranteed since any relevant data is encrypted and therefore cannot be manipulated by third-parties. Privacy is retained, as the user decides, which data can be used by an app and s/he can specify for any non-health data how accurate or even randomized it should be provided. As an app cannot access any data without using the PMP, data access is strictly constrained by the Resources' interfaces. Thus, from an information security's point of view the revised app satisfies all requirements.

Concerning device interoperability, the modular expandability of the Resources turns out to be beneficial. For instance, support for additional devices can be added to the Metering Resource need-based at runtime. An app developer only has to code against a Resource's interface, no matter which hardware is actually available. Therefore, complex and labor-intensive coding is required only once (for the Resource) and it can be reused many times (in the apps). Additionally,

Table 1: Feature Summary of the Implemented PMP Resources [based on 41].

PMP Resource	Device Interoperability	Information Security
Dialog Box	• tailoring of dialog text • tailoring of data processing	• app has no access to data
Metering	• support of different devices	• device restriction
Location	• different location sensors • different modes of operation	• restriction of accuracy • randomized location data
Secure Database	• generic data model	• full database encryption
Connector	• support of different services	• no direct network access • restriction of usable services
Unsealer		• limited access to health data

due to their generic design the Resources are usable in many different application scenarios even for non-mHealth apps. Thus, from an interoperability's point of view the PMP Resources satisfy all requirements regarding compatibility and reusability. Table 1 lists the key contributions of the introduced Resources.

8 Conclusion and Outlook

Today's medical devices for home use can often be connected to Smartphones via Bluetooth. On the one hand, health data can be sent from the medical device to the Smartphone for processing or presentation. On the other hand, the Smartphone can be used to control the medical device. By connecting to several medical devices, the Smartphone virtually becomes a health hub for the pants pocket. This provides entirely new possibilities for telemedicine. This is particularly beneficial for the treatment of chronic diseases such as COPD. The treatment of such a disease makes it necessary for patients to undergo regular screenings. This results in increasing treatment costs as well as overburdened physicians. However, mHealth Apps can drastically reduce the number of visits to the physician. Yet, the patients' confidence in mHealth apps is impaired due to device interoperability and information security issues.

We address these problems in our work. For this purpose, we describe the multilayered architecture of mHealth apps and analyze one particular mHealth app for COPD patients. Based on this analysis, we create a generic data model for mHealth apps. Furthermore, we derive relevant functions of an mHealth app, in which device interoperability or information security is an issue. We design and implement Resources for the PMP (a privacy-aware data provisioning system) which support each of these mHealth functions. To evaluate the practical effect of these PMP extensions, we re-engineer the analyzed COPD app and integrate our Resources into it. Based on this revised app, we assess whether our approach contributes to solve the device interoperability and information security issues.

While our approach is a solution to the device interoperability issue, further comprehensive measures have to be taken regarding information security. As the analysis of the multilayered architecture of mHealth apps shows (see Sect. 2), the health data is preprocessed on the Smartphone, only. Most of the data processing takes place on external servers. A large number of different data sources are combined on these servers. Thus an immense amount of data is available for this external data processing, from which a lot of knowledge about the patients can be derived. Therefore, information security measures for Smartphones, as presented in this paper, are not sufficient. If a user has control over the data on his or her Smartphone, the back-end might still be able to obtain the same data from other sources. Thus, an holistic information security approach has to be applied to both, the Smartphone Layer as well as the Back-End Layer. Therefore, future work has to determine, how a information security system for back-ends such as PATRON [35, 36] can be integrated in our PMP-based approach [34].

Acknowledgments This paper is part of the PATRON research project which is commissioned by the Baden-Württemberg Stiftung gGmbH. The authors would like to thank the BW-Stiftung for the funding of this research.

References

1. Bai, Y., Dai, L., Li, J.: Issues and Challenges in Securing eHealth Systems. International Journal of E-Health and Medical Communications **5**(1), 1–19 (2014)
2. Bhandari, V.: Enabling Programmable Self with HealthVault. O'Reilly Media, Inc., Beijing, Cambridge, Farnham, Köln, Sebastopol, Tokyo (2012)
3. Bitsaki, M., Koutras, C., Koutras, G., Leymann, F., Mitschang, B., Nikolaou, C., Siafakas, N., Strauch, S., Tzanakis, N., Wieland, M.: An Integrated mHealth Solution for Enhancing Patients' Health Online. In: Proceedings of the 6th European Conference of the International Federation for Medical and Biological Engineering. pp. 695–698. MBEC '14 (2014)
4. Bitsaki, M., Koutras, C., Koutras, G., Leymann, F., Steimle, F., Wagner, S., Wieland, M.: ChronicOnline: Implementing a mHealth solution for monitoring and early alerting in chronic obstructive pulmonary disease. Health Informatics Journal **23**(3), 197–207 (2016)
5. Bluetooth SIG, Inc.: GATT Specifications. Tech. rep. (2017), https://www.bluetooth.com/specifications/gatt
6. Chan, M., EstèVe, D., Fourniols, J.Y., Escriba, C., Campo, E.: Smart Wearable Systems: Current Status and Future Challenges. Artificial Intelligence in Medicine **56**(3), 137–156 (2012)
7. Cherdantseva, Y., Hilton, J.: A Reference Model of Information Assurance & Security. In: Proceedings of the 2013 International Conference on Availability, Reliability and Security. pp. 546–555. ARES '13 (2013)
8. Dhillon, G., Backhouse, J.: Technical Opinion: Information System Security Management in the New Millennium. Communications of the ACM **43**(7), 125–128 (2000)

9. Enck, W., Gilbert, P., Chun, B.G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation. pp. 393–407. OSDI '10 (2010)
10. Gardner, R.W., Garera, S., Pagano, M.W., Green, M., Rubin, A.D.: Securing Medical Records on Smart Phones. In: Proceedings of the First ACM Workshop on Security and Privacy in Medical and Home-care Systems. pp. 31–40. SPIMACS '09 (2009)
11. Gupta, N.: Inside Bluetooth Low Energy. Artech House Publishers, Boston, London (2013)
12. Hester, J., Peters, T., Yun, T., Peterson, R., Skinner, J., Golla, B., Storer, K., Hearndon, S., Freeman, K., Lord, S., Halter, R., Kotz, D., Sorber, J.: Amulet: An Energy-Efficient, Multi-Application Wearable Platform. In: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems. pp. 216–229. SenSys '16 (2016)
13. Hsu, H.H., Peng, W.J., Shih, T.K., Pai, T.W., Man, K.L.: Smartphone Indoor Localization with Accelerometer and Gyroscope. In: Proceedings of the 2014 17th International Conference on Network-Based Information Systems. pp. 465–469. NBiS '14 (2014). https://doi.org/10.1109/NBiS.2014.72
14. IEEE 11073 Standards Committee: ISO/IEC/IEEE Health informatics–Personal health device communication–Part 20601: Application profile–Optimized exchange protocol. ISO/IEEE 11073-20601:2014 (2014)
15. Jafari, M., Safavi-Naini, R., Sheppard, N.P.: A Rights Management Approach to Protection of Privacy in a Cloud of Electronic Health Records. In: Proceedings of the 11th Annual ACM Workshop on Digital Rights Management. pp. 23–30. DRM '11 (2011)
16. Knöll, M., Moar, M.: On the Importance of Locations in Therapeutic Serious Games: Review on current health games and how they make use of the urban landscape. In: Proceedings of the 2011 5th International Conference on Pervasive Computing Technologies for Healthcare and Workshops. pp. 538–545. PervasiveHealth '11 (2011)
17. Kouris, I., Koutsouris, D.: Identifying Risky Environments for COPD Patients Using Smartphones and Internet of Things Objects. International Journal of Computational Intelligence Studies **3**(1), 1–17 (2014)
18. Kumar, S., Nilsen, W., Pavel, M., Srivastava, M.: Mobile Health: Revolutionizing Healthcare Through Transdisciplinary Research. Computer **46**(1), 28–35 (2013)
19. Marcelino, L., Silva, C.: Location Privacy Concerns in Mobile Applications, pp. 241–249. Springer, Cham (2018)
20. Mare, S., Sorber, J., Shin, M., Cornelius, C., Kotz, D.: Hide-n-Sense: Preserving Privacy Efficiently in Wireless mHealth. Mobile Networks and Applications **19**(3), 331–344 (2014)
21. Mattila, E., Korhonen, I., Salminen, J.H., Ahtinen, A., Koskinen, E., Särelä, A., Pärkkä, J., Lappalainen, R.: Empowering Citizens for Well-being and Chronic Disease Management with Wellness Diary. IEEE Transactions on Information Technology in Biomedicine **14**(2), 456–463 (2010)
22. Mi, X., Qian, F., Zhang, Y., Wang, X.F.: An Empirical Characterization of IFTTT: Ecosystem, Usage, and Performance. In: Proceedings of the 2017 Internet Measurement Conference. pp. 398–404. IMC '17 (2017)
23. Milošević, M., Shrove, M.T., Jovanov, E.: Applications of Smartphones for Ubiquitous Health Monitoring and Wellbeing Management. Journal of Information Technology and Applications **1**(1), 7–15 (2011)
24. Mishra, S.M.: Wearable Android: Android Wear and Google FIT App Development. Wiley Online Library, Hoboken, New Jersey (2015)

25. Moehrke, J.: Audit Trail and Node Authentication. Tech. rep., IHE International (Aug 2017), https://wiki.ihe.net/index.php/Audit_Trail_and_Node_Authentication
26. Moehrke, J.: Mobile access to Health Documents (MHD). Tech. rep., IHE International (Oct 2017), https://wiki.ihe.net/index.php/Mobile_access_to_Health_Documents_(MHD)
27. Murad, A., Schooley, B., Abed, Y.: A Secure mHealth Application for EMS: Design and Implementation. In: Proceedings of the 4th Conference on Wireless Health. pp. 15:1–15:2. WH '13 (2013)
28. Murnane, E.L., Huffaker, D., Kossinets, G.: Mobile Health Apps: Adoption, Adherence, and Abandonment. In: Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers. pp. 261–264. UbiComp/ISWC '15 Adjunct (2015)
29. O'Donoghue, J., Herbert, J.: Data Management Within mHealth Environments: Patient Sensors, Mobile Devices, and Databases. Journal of Data and Information Quality **4**(1), 5:1–5:20 (2012)
30. Schweitzer, J., Synowiec, C.: The Economics of eHealth and mHealth. Journal of Health Communication **17**(Supplement 1), 73–81 (2012)
31. Shmueli, E., Vaisenberg, R., Elovici, Y., Glezer, C.: Database Encryption: An Overview of Contemporary Challenges and Design Considerations. ACM SIGMOD Record **38**(3), 29–34 (2010)
32. Siewiorek, D.: Generation Smartphone. IEEE Spectrum **49**(9), 54–58 (2012)
33. Silva, B.M., Rodrigues, J.J., de la Torre Díez, I., López-Coronado, M., Saleem, K.: Mobile-health: A Review of Current State in 2015. Journal of Biomedical Informatics **56**(C), 265–272 (2015)
34. Stach, C., Alpers, S., Betz, S., Dürr, F., Fritsch, A., Mindermann, K., Murthy Palanisamy, S., Schiefer, G., Wagner, M., Mitschang, B., Oberweis, A., Wagner, S.: The AVARE PATRON: A Holistic Privacy Approach for the Internet of Things. In: Proceedings of the 15^{th} International Conference on Security and Cryptography. pp. 1–8. SECRYPT '18 (2018)
35. Stach, C., Dürr, F., Mindermann, K., Palanisamy, S.M., Tariq, M.A., Mitschang, B., Wagner, S.: PATRON — Datenschutz in Datenstromverarbeitungssystemen. In: Informatik 2017: Digitale Kulturen, Tagungsband der 47. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 25.9-29.9.2017, Chemnitz. LNI, vol. 275, pp. 1085–1096 (2017), *(in German)*
36. Stach, C., Dürr, F., Mindermann, K., Palanisamy, S.M., Wagner, S.: How a Pattern-based Privacy System Contributes to Improve Context Recognition. In: Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops. pp. 238–243. CoMoRea '18 (2018)
37. Stach, C., Mitschang, B.: Privacy Management for Mobile Platforms – A Review of Concepts and Approaches. In: Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management. pp. 305–313. MDM '13 (2013)
38. Stach, C., Mitschang, B.: Design and Implementation of the Privacy Management Platform. In: Proceedings of the 2014 IEEE 15th International Conference on Mobile Data Management. pp. 69–72. MDM '14 (2014)
39. Stach, C., Mitschang, B.: Secure Candy Castle — A Prototype for Privacy-Aware mHealth Apps. In: Proceedings of the 2016 IEEE 17th International Conference on Mobile Data Management. pp. 361–364. MDM '16 (2016)
40. Stach, C., Mitschang, B.: CURATOR—A Secure Shared Object Store: Design, Implementation, and Evaluation of a Manageable, Secure, and Performant Data

Exchange Mechanism for Smart Devices. In: Proceedings of the 33rd ACM/SIGAPP Symposium On Applied Computing. pp. 533–540. DTTA '18 (2018)

41. Stach, C., Steimle, F., Mitschang, B.: The Privacy Management Platform: An Enabler for Device Interoperability and Information Security in mHealth Applications. In: Proceedings of the 11th International Conference on Health Informatics. pp. 27–38. HEALTHINF '18 (2018)
42. Stach, C., Steimle, F., Franco da Silva, A.C.: TIROL: The Extensible Interconnectivity Layer for mHealth Applications, pp. 190–202. Springer, Cham (2017)
43. Steimle, F., Wieland, M.: ECHO — An mHealth Solution to Support Treatment of Chronic Patients. In: Proceedings of the 8th ZEUS Workshop. pp. 64–67. ZEUS '16 (2016)
44. Steimle, F., Wieland, M., Mitschang, B., Wagner, S., Leymann, F.: Extended provisioning, security and analysis techniques for the ECHO health data management system. Computing **99**(2), 183–201 (2017)
45. Stollmann Entwicklungs- und Vertriebs-GmbH: Terminal I/O Profile: Client implementation guide. Tech. rep., Telit (2014)
46. de Toledo, P., Jimenez, S., del Pozo, F., Roca, J., Alonso, A., Hernandez, C.: Telemedicine Experience for Chronic Care in COPD. IEEE Transactions on Information Technology in Biomedicine **10**(3), 567–573 (2006)
47. Ur, B., McManus, E., Pak Yong Ho, M., Littman, M.L.: Practical Trigger-action Programming in the Smart Home. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 803–812. CHI '14 (2014)
48. Weerasinghe, D., Rajarajan, M., Rakocevic, V.: Device Data Protection in Mobile Healthcare Applications, pp. 82–89. Springer, Berlin, Heidelberg (2009)
49. Wieland, M., Hirmer, P., Steimle, F., Gröger, C., Mitschang, B., Rehder, E., Lucke, D., Rahman, O.A., Bauernhansl, T.: Towards a Rule-based Manufacturing Integration Assistant. Procedia CIRP **57**(1), 213–218 (2016)
50. Wieland, M., Steimle, F., Mitschang, B., Lucke, D., Einberger, P., Schel, D., Luckert, M., Bauernhansl, T.: Rule-Based Integration of Smart Services Using the Manufacturing Service Bus. In: Proceedings of the 2017 IEEE 14th International Conference on Ubiquitous Intelligence and Computing. pp. 22:1–22:8. UIC '17 (2017)
51. World Health Organization: Chronic obstructive pulmonary disease (COPD). Tech. rep., WHO Media Centre (2015)
52. Xie, H., Gu, T., Tao, X., Lu, J.: A Reliability-Augmented Particle Filter for Magnetic Fingerprinting Based Indoor Localization on Smartphone. IEEE Transactions on Mobile Computing **15**(8), 1877–1892 (2016). https://doi.org/10.1109/TMC.2015.2480064
53. Ye, H., Gu, T., Tao, X., Lu, J.: Scalable Floor Localization Using Barometer on Smartphone. Wireless Communications & Mobile Computing **16**(16), 2557–2571 (2016). https://doi.org/10.1002/wcm.2706

All links were last followed on June 15, 2018.