Remote Accessibility to Diabetes Management and Therapy in Operational Healthcare Networks

REACTION (FP7 248590)

D7.4 Virtualisation in distributed healthcare applications

Date: 2012-08-31

Version 1.0

Dissemination Level: Public

Table of Contents

Document control page

Code	D7-4_Virtualisation-in-distributed-healthcare-applications_V10_FHG-SIT.pdf
Version	1.0
Date	2012-08-31
Dissemination level	PU
Category	R
Participant Partner(s)	FHG-SIT
Author(s)	Matthias Enzmann (FHG-SIT), Frederik Franke (FHG-SIT)
Verified and approved by	
Work Package	WP 7
Fragment	No
Distribution List	All
Abstract	This document presents the rationales for and capabilities of the identity management and access control components developed within REACTION. It also discusses the security and usability of the employed authentication mechanisms.
Comments and modification	
Status	□ Draft ☒ Task leader accepted ☒ WP leader accepted □ Technical supervisor accepted □ Medical Engineering accepted □ Medical supervisor accepted □ Quality Manager checked ☒ Project Coordinator accepted
Action requested	□ to be revised by partners involved in the preparation of the deliverable □ for approval of the task leader □ for approval of the WP leader □ for approval of the Technical Manager □ for approval of the Medical Engineering Manager □ for approval of the Medical Manager □ for approval of the Quality Manager □ for approval of the Project Coordinator
Keywords	identity management, virtual identity, authentication mechanisms, access control
References	
Previous Versions	

Version Notes

Version	Author(s)	Date	Changes made
0.1	Enzmann, Franke	2012-05-16	Initial version
0.2	Enzmann, Franke	2012-08-17	Reviewers' version
1.0	Enzmann, Franke	2012-08-31	Final version submitted to the European Commission

Internal Review History

Reviewed by	Date	Comments made
Stefan Asanin (CNET)	2012-08-27	Approved with comments
Bernhard Höll (MUG)	2012-08-24	Approved with comments

1 Executive Summary

Virtualisation in REACTION has two sides, virtualisation of devices and virtualisation of identities. Virtualisation of medical devices means that similar devices, e.g., all weight scales or all glucose meters, can be accessed in a uniform way through a virtual interface from any place, i.e., physical access to the device is no longer necessary. This functionality is largely provided by the Hydra middleware which was developed in the EU project of the same name and has been extended for REACTION.[1] The second aspect, virtualisation of identities, is the subject of this deliverable.

A virtual identity can be seen as an entity's digital self. It is the sum of the various identities, traits or privileges any given entity may have. For instance, a human entity may have an identity as "employee", which grants him certain privileges, and another one as the "head of department" which gives him additional rights. The virtual identity captures the fact that the person behind the two identities is the same. In the IT world, users typically have several identities which they have to manage, e.g., one for access to their computer, one for email access, one for access to an online service, etc. Even within an organisation, such as a hospital, multiple identities might be required for accessing different services, since the IT landscape is typically not homogeneous. Hence, it can become difficult to manually and still securely manage different identities and their associated credentials, e.g., passwords.

In this document, the identity model for REACTION is introduced. It defines a virtual identity as the sum of an entity's digital identities, which in turn are abstractions of concrete credential mechanisms like user name / password schemes or challenge-response schemes making use of digital certificates. The abstraction of a digital identity allows to handle identities in an identity management (IdM) component in a uniform way. Such an IdM component is introduced for REACTION's in-hospital scenario where human entities may assume different identities that can be attached to different services. Identities, in turn, can be attached to roles, e.g., administrator or health professional. A role can be seen as a container for permissions required by a certain group of users. For instance, all physicians within a ward have the same permissions and, thus, these permissions can be collected in a role "physician". This approach is widely known as Role-Based Access Control (RBAC) and together with IdM forms the basis for the in-hospital's access control architecture.

The primary care scenario of REACTION is very different from the in-hospital scenario, e.g., devices / services are operated / used by lay persons (patients) in uncontrolled environments (home PC, smart phone, etc.). Therefore a less uniform way of dealing with identities was necessary due to restrictions that were imposed by these environments. Since primary care patients are typically not IT professionals and only have basic skills in operating PCs, ease of use and flexibility had to be balanced with static, possibly more secure but also more complex approaches. A discussion on this topic can also be found in this document.

The identities and the associated authentication schemes employed in REACTION scenarios are also presented and discussed with respect to usability and security. Furthermore, usability and security aspects are weighed for each scenario. Finally, the identity architecture for both scenarios and its technical implementation is presented and evaluated in terms of security and usability. For the access control component, two policy languages, XACML and OPL, had been evaluated in terms of expressiveness, extensibility, and ease of administration. OPL was finally given precedence over XACML due to its simpler language constructs. The roles defined with the policy language can be attached to identities which must be presented by entities before they are given access to any service. This allows enforcement of access rights, as unknown identities or identities with restricted access may not use services for which they are not authorised. Hence, the RBAC model implemented in REACTION allows fine-grained control over data access which adds to the security of the overall system.

[1] The Hydra middleware had been renamed to LinkSmart middleware due to brand name issues.

2 Terms and Definitions

2.1 Abbreviations and Acronyms

ACC	Access Control Component
AES	Advanced Encryption Standard
EE	End Entity
EPR	Electronic Patient Record
GMS	Glucose Management System
GP	General Practitioner
HTTP	HyperText Transfer Protocol
HTTPS	HTTP over SSL/TLS
IdM	Identity Management
IEEE	Institute of Electrical and Electronics Engineers
IPC	Interprocess Communication
ISO	International Organization for Standardization
IT	Information Technology
LAN	Local Area Network
OPL	ORKA Policy Language
OS	Operating System
PAP	Policy Administration Point
PBKDF2	Password-Based Key Derivation Function 2
PC	Personal Computer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKC	Public Key Cryptography
PMS	Profile Management Service
P4HP	Portal for Health Professionals
P4P	Portal for Patients
RBAC	Role-Based Access Control
SKC	Shared Key Cryptography
SOAP	Simple Object Access Protocol
SSA	Security Service App
SSL	Secure Sockets Layer
TLS	Transport Layer Security
URL	Uniform Resource Locator
USB	Universal Serial Bus
WWW	World Wide Web
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language

3 Introduction

Virtualisation has become an important factor in the IT industry. It normally refers to the virtualisation of hardware, i.e., components or complete computer systems are wholly or partly emulated in software on a single machine. In the EU project Hydra[2], virtualisation had been defined in a much broader sense [10]:

> "[Virtualisation is] *a method to create logical entities that are present within a certain scope (e.g., an application) but do not need to physically exist in the same form.*"

The REACTION project 'inherited' this definition from Hydra as REACTION makes use of technology that had been developed in the Hydra project. In particular, Hydra's ability to integrate medical devices by providing virtual device interfaces for different classes of devices, e.g., weight scales, glucose meters, etc., is beneficial to the REACTION project. The definition above can also be extended to REACTION where virtual identities play an important role. A virtual identity can be seen as an entity's digital (virtual) self. It can be composed of a number of different digital identities or traits the entity may use for different contexts. Therefore, a virtual identity is more a concept than something that can be actually used.

A digital identity can be thought of as an identifier used by an entity when communicating with another entity. For instance, a human entity "John Smith" may have different account names (and passwords) for different services, e.g., "jsmith" for logging in to his company's computer system, "js1984" for the social network Facebook, and "48htims" for the photo platform Flickr.[3] These identities are different from each service provider's point of view, however, the person behind them —the entity "John Smith"— is the same. Hence, a virtual identity reflects and sums up the different contexts in which a user is active under a certain digital identity. However, the virtual identity as such cannot be presented to someone in order to identify the end entity "John Smith" — in that sense it is literally virtual. Each digital identity may just reflect different characteristics or roles of the same entity. The 1:1 relationship between an entity and its virtual identity as well as the connection to digital identities is summarised in Figure 1.

The reason for having an identity in the virtual world is that the end entity behind it can be addressed and often that, based on his/her used identity, the end entity will obtain certain privileges. In colloquial language, identification is often understood as presenting an identity, including an implicit assumption that it can be verified by someone else. In security terms, this process is divided into two phases, identification and authentication. *Identification* is just an entity's claim of an identity, e.g., sending a login name. *Authentication of an identity* is the process of proving that the presented identity really belongs

[2] http://www.hydramiddleware.eu/

[3] Note that "the company's computer system", "Facebook", and "Flickr" are entities in themselves.

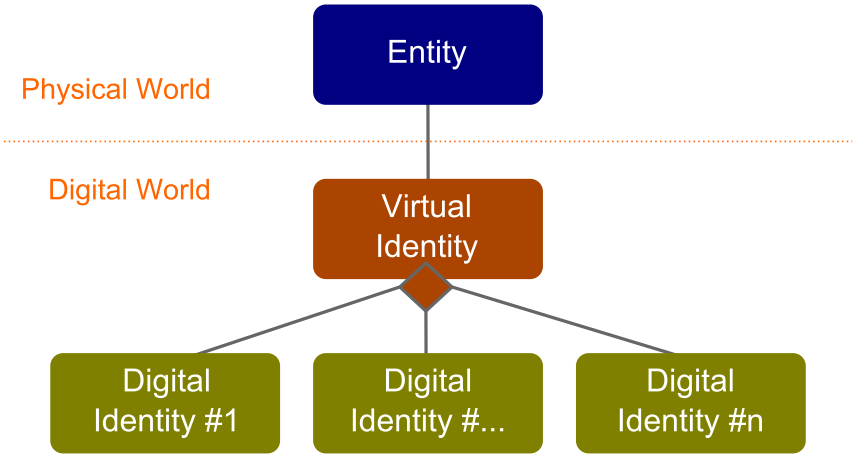

Figure 1: Virtual Identity

to the claimant. However, in certain contexts, it is not sufficient to just obtain an authenticated identity of an entity. It must also be clear if that identity had been *authorised* to access certain services or data. Again, security distinguishes two processes, authorisation and access control. *Authorisation* is the process of granting certain permissions to an identity while *access control* is the process of granting or denying access to resources based on the presented identity and its associated permissions. Hence, authorisation refers to the definition of permissions and their assignment to identities (access rules) while access control refers to the enforcement of these rules.

In the context of the REACTION project, ensuring identification, authentication, authorisation, and access control is one of the major tasks in terms of security. Clearly, electronic patient records (EPR) may only be accessed by authorised medical personnel, such as nurses or physicians. However, even patients may be allowed to access certain information from their own EPR, given that they can prove their identity. Since a virtual identity might be comprised of several digital identities, it can be a burden for users to remember which identity had been used for which service. This is where identity management (IdM) comes into play. It is supposed to support users in keeping track of their various identities in use. This should help to ease the cognitive load put on users but also to avoid mistakes, e.g., using the password of service A for service B. An IdM system is integrated into the REACTION hospital application that is used by medical personnel to help in the treatment of diabetic patients in a hospital ward. The medical personnel, however, is not completely uniform. It consists of physicians, heads of departments, nurses, trainees, etc., all of which have different training and consequently different roles in the hospital. Different roles can be associated with different identities, e.g., the head of endocrinology will also be a physician and, thus, may use two different digital identities, depending on whether the privileges of a physician or the head of department are required for the task at hand. Now, keeping track of which privilege is required for which service can be a burden for users, again. However, this task can also be handed over to the IdM which, based on a rule set, can select the identity required for the user's requested service.

In this deliverable, the models behind the identity and access control management implemented and used in REACTION's scenarios will be presented as well as a discussion on their security and on some of the design choices and options for the realisations.

4 Identity Management

Identity management (IdM) refers to a service consumer's need to maintain different identities with different service providers. Typically, managing identities goes along with managing credentials, i.e., information that is used to prove a certain statement with respect to the presented identity, e.g., you are who you claim to be, you are over 18, etc. In the following when we speak of an identity, we also mean its associated credentials.

The need to maintain different identities is not exclusive to the digital world. In the real world, different identities are quite natural. For instance, as a company employee one may have a badge allowing unhindered access to the company's buildings while for after work exercises in a gym a member pass may be required. Although the person behind these two identity cards can be the same, both attest different statements about an identity. Having different identities for different contexts, e.g., work and free time, also has implications for privacy and security. These potential problems are even worse in the digital world where context switches may happen very often, e.g., visiting different web sites, working on different tasks or cases, etc.

Privacy. If an entity, say a human computer user, would always use the same identifier to identify herself to different services then tracking the user would be easy. If users are aware of the tracking but do not want to be tracked they might start to 'counter' the tracking by avoiding certain places or situations as long as the tracking is active. This in turn affects the autonomy and informational self-determination of the user [13]. Thus from the privacy point of view, it would be better to allow users to use different identifiers for different services and leave the choice whether to allow linking of activities to the user.

However, this puts a cognitive burden on the user as she has to remember which identifier was meant for which service. □

Security. If an entity uses the same credentials with different services, the credentials might be 're-used' by unauthorised parties. For instance, if the credential would be a password, the service provider would also know it and may use this knowledge to misrepresent himself to other services used by the credential's true owner. Another problem would be a compromise of a service provider's IT infrastructure such that third parties gain unauthorised access to users' credentials and subsequently be able to use the stolen data to gain access to other services used by these users [15]. □

The implications above suggest that different identities should be used for different services, if the user does not have a good reason to link two or more of her activities. Typically, identities are reused by users to lower their cognitive burden but at the same time puts them at a risk [18]. Identity management is supposed to lower users' cognitive burden by helping them to manage different identities for different services such that neither privacy nor security is compromised.

In the following, we will introduce the background and general framework for the identity management in REACTION.

4.1 End Entities and Context

Our model for identity management incorporates different classes of actors. These actors can be

- humans,
- devices,
- services

and we will generally refer to them as *end entities*. Characteristically, an end entity (EE) for the purpose of this document is one end of a communication channel, e.g., a client or a server. An EE may have more than a single identity but always has at least one. In a message exchange between two EEs, each EE may only use one of its identities. In client-server communication, the client will normally choose its identity at will while the server will use the identity that is addressed by the client. Therefore, the context of a given message exchange will depend in part on the identities chosen or addressed by the EEs. In general, there will be other factors influencing the context that need to be taken into account, e.g., the type of message. As we will later see, context plays an important role when determining rights and authorisations for an EE.

4.2 Scenarios

The REACTION project deals with two major scenarios "in-hospital" and "primary care" [35]. The two scenarios have different user groups, different environments, and different needs. In the following, we will briefly outline the use cases from a technical point of view and highlight some security relevant details.

In-hospital. The Glucose Management System (GMS) is at the heart of the in-hospital scenario [36]. It is comprised of a client component running on a mobile Android device and a stationary server component, see Figure 2. The client is mainly used as an input/output device while the server does the processing and the recording of patient data. A device is carried by medical personnel, i.e., nurses or physicians, and the server is located on the premises of the hospital. A device accesses the hospital network using a wireless LAN while the server is connected to a wired LAN.

Medical personnel will use the client device during their ward round and other duties to access patient information, e.g., previously entered blood glucose values, and to enter new data or update previously recorded data [36, 24]. Patients seen as entities (in the physical world) are identified by their names,

date of birth, and room numbers while their digital identity in REACTION's hospital context is simply a unique identifier chosen and internally managed by the hospital system and therefore does not require authentication. In turn, the medical personnel operating the device will have to authenticate to the GMS client application in order to be able to use and access the system and the patients' data. □

Primary care. The primary care scenario is more heterogeneous than the in-hospital scenario. It is mainly comprised of a portal for patients (P4P), a portal for health professionals (P4HP), and components for sending and receiving patient observations from a patient's home network to the GP's network, see Figure 3. The portals' user interfaces are provided by plain browsers, i.e., they are realised as web applications. In a way this approach is similar to the in-hospital scenario's approach where the client app is mainly used for input/output of data but not for it's processing. The P4P can be accessed from any place while the P4HP will only be accessible within the GP's own network. The GP's network is assumed to be a trusted domain [16], i.e., some security requirements are relaxed. Specifically, cryptographic means providing confidentiality for data circulating in the GP's network are not mandatory. Of course, patient data going in and out of the network must be treated confidentially. Otherwise, the data might be read by unauthorised third parties. Hence, link 1 from Figure 3 must use HTTPS while internal communication may use HTTP, as seen with links 4 and 5.

The P4P allows patients to view their physiological measurements, e.g., weight or blood glucose, and provides a questionnaire that is supposed to help patients when they require information regarding diet, activity, or medication. Note that access to medical data, such as observations, is read-only for patients. Conversely, the P4HP is only meant for medical personnel. It allows health professionals to manage general patient data —name, birthday, gender, etc.— and medical data of the patient, e.g., physiological measurements or healthcare parameters like target values for blood pressure [36, 24].

The REACTION primary care client and server [7] are responsible for transmitting observations from the patient's home network to the GP's network, see link 3 in Figure 3. Observations can be measurements such as the patient's weight or blood pressure. The REACTION primary care client 'pulls' the data from a medical device which will take the measurement in the first place. The connection between the medical device and the client will often be made using wireless protocols like ZigBee [38, 39] or Bluetooth

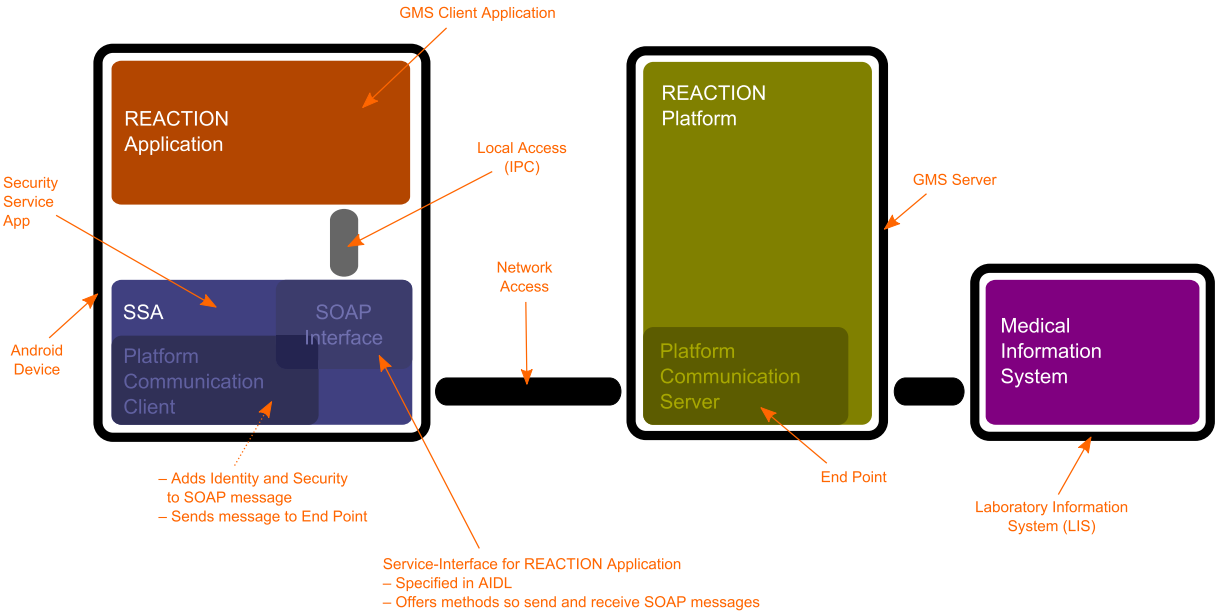

Figure 2: Communication architecture of the in-hospital scenario

[31, 33, 32] (see link 2) but may also use cable connections, e.g., USB. The REACTION primary care client and server use their own peer-to-peer protocol for communication [30]. In this protocol, XML Signatures [11] and XML Encryption [20] are used to protect the messages' contents.

4.3 Users and Environment

For the purpose of this document, the term "users" will only be used for human end entities operating some device or software. The users in REACTION can basically be divided into two groups:

- professionals,
- lay persons.

These terms do not necessarily refer to the IT skills of the users but rather to their medical training. Still, professionals can be assumed to also have received some form of training or information related to the usage of the applications developed within the REACTION project. Conversely, lay persons —patients, informal carers, family, etc.— can only be assumed to have at most general IT skills. Furthermore, professionals can be assumed to have and operate 'special devices' while lay persons can at most be assumed to own and be familiar with standard PCs and maybe even with smart phones.

The environment in which the systems developed in REACTION are operated will also depend on the user group. Professionals will normally use REACTION applications at their office, in the clinic, or at the GP's practice. Lay persons may use them at home, possibly at a shared family PC, or even on the move. Hence, in the professional setting, we assume to find a more controlled environment than in the lay person environment where the device might be taken to unforeseeable places and where software for all kinds of things might be installed. This also means that the security of the PC that is being used to access the REACTION portal must be taken into account, e.g., if the PC is running up-to-date anti-virus software.

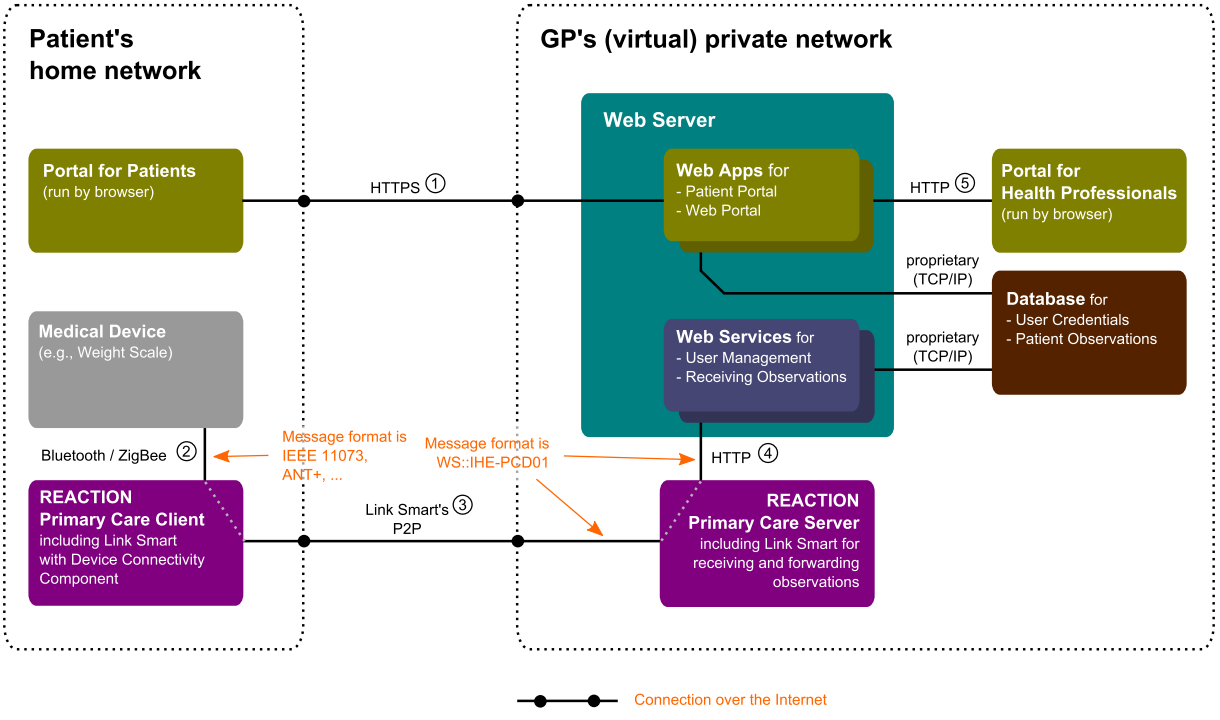

Figure 3: Communication architecture of the primary care scenario

4.4 Digital Identities

In general, a digital identity can be any identifier that is *unique within a given context*. In this document, we use a more restricted definition as we will only accept an identifier as a digital identity if the claimant can prove that he/she/it is who he/she/it claims to be. Therefore, a digital identity will always encompass some kind of proof information that can be verified by another party. In the following, we omit the "digital" from "digital identity" for brevity.

In the scope of the REACTION project, we have two different scenarios, "in-hospital" and "primary care", where different kinds of identities will be used. The necessity for employing different kinds of identities is mostly driven by the respective use cases and the technical constraints that come along with these use cases as we will see shortly. For REACTION applications, we consider the following identities:

- user name & password,
- digital certificate,
- shared key.

In the following, we will briefly sketch the motivation for having each of these identities incorporated into the scenarios. We will come back to these types of identities in section 5 where we will discuss their security.

User name & password. This scheme is probably the most common form of an identity used throughout the World Wide Web (WWW). It requires an entity —typically the user— to choose both a user name and a password. If users are free to choose their user names, the system ensures that the names are unique, i.e., it will reject user names already chosen by other users. Some systems also place constraints on the passwords, e.g., the password must have at least six characters, it must have at least one digit, it must have at least one 'special' character like ".", "#", "+", etc.

User name & password schemes are easy to implement for providers and do not require sophisticated software at the consumer side as a consumer just has to enter its name and password into a form which is subsequently sent to the provider. This scheme can be useful if users are expected to access a service from multiple locations or from multiple devices as it hardly makes a difference if a user enters her user name & password on a PC or on a smart phone. In particular, the devices do not need to store any data regarding the user's credential which is an advantage in terms of security. Thus, user name & password are easy to deploy and portable by nature which were the main reasons why this approach was chosen for REACTION's patient portal as well as for the web portal for health professionals. □

Digital Certificate. This scheme employs public key cryptography to prove ownership of an identity and thus requires key material that must be supplied by service users and service providers. In the WWW, digital certificates are frequently found in web servers supporting HTTPS (HTTP over TLS/SSL). These servers identify themselves using a digital certificate, i.e., they present a certificate that contains their respective domain name, like `www.reaction-project.eu`, and a public key along with some more technical information. Using the underlying cryptographic protocol TLS (or its predecessor SSL), servers prove ownership of the private key that belongs to the certified public key from their identity certificate. Naturally, the private key should only be known to its holder and must not be disclosed to anyone else.

Using certificates with TLS is a very sophisticated scheme but nevertheless has found its way into all major web browsers. Although digital certificates are most dominantly used to identify web servers, users can also employ certificates to identify themselves to a server. Of course, the server must support client authentication using digital certificates to make this work. As noted in the beginning of this paragraph, users will have to provide key material, i.e., the private key and a corresponding certificate, to make use of such an identity. Therefore, users of digital certificates are typically not as free as users of user name & password schemes, as they will always have to bring their private key and certificate on some kind of media, e.g., a USB stick or smart card, if they want to use them on different devices. In addition, users

may have difficulties in configuring their software to use certificates, if the software supports them at all. For instance, the Firefox browser supports user certificates, however, the settings for installing and using them are split into separate configuration areas, which are neither easy to find nor easy to understand for non-expert users.

In the web portals developed for the primary care scenario, certificates can be easily used for servers as the server will always run on the same machine, needs to be configured only once with a server certificate, and the installation can be done by an expert. In other words, the server is professionally managed, stationary, and its configuration is quite static. In contrast, users should not be burdened with carrying media and installing certificates, which is why it was decided to not give them identity certificates.

For the in-hospital scenario, however, certificates are used to identify all kinds of end entities. In this scenario, the environment is quite different from the primary care's which is why it was conceptionally easier to employ certificates for users, too. Firstly, the environment is closed, i.e., it is restricted to devices running within the clinic and users are not allowed to install additional software. Secondly, it is a completely professional environment where the devices are administered by trained personnel, i.e., lay persons —like patients— are never involved. And thirdly, the application developed for the hospital has been built from scratch as a stand-alone application. For this stand-alone application, no dependencies on third party software —like browser interfaces or other browser capabilities— had to be taken into account. Furthermore, the design of the application considered a user-friendly way of incorporating certificates right from the beginning. □

Shared Key. This identification scheme employs symmetric cryptography. Symmetric cryptography typically requires much less processing power than public key computations, like those required to verify digital certificates. Therefore, the hardware requirements for symmetric key cryptography can be met easier and cheaper than those for public key cryptography. This makes it particularly attractive for manufacturers of medical consumer devices.

Medical devices with data interfaces are often wirelessly connected to a base station and hence have to transmit their data over the air. In order to secure such transmissions, symmetric cryptography is typically used. For this, a cryptographic key is employed that is used by the sender to encrypt the data and by the receiver to decrypt it. Since both ends use the same key for their respective operation, the key must be known to both — hence the name "shared key".

Major wireless protocols —like Bluetooth or ZigBee— make use of symmetric cryptography and have devised schemes for the exchange of the key prior to encryption and decryption operations. Since REACTION makes use of medical consumer devices in the primary care scenario, shared key identities had to be included in the model for digital identities, too. □

5 Authentication Technologies

Authentication technologies allow to verify the validity of some statement presented by a claimant, e.g., they allow to verify that a claimed identity is truly the identity of the claimant. Many authentication schemes are available today and built on different technologies. Furthermore, technologies are also subject to different approaches, e.g., they may use

- something you know (e.g., a passphrase),
- something you have (e.g., a smart card with a key pair), or
- something you are (e.g., biometrics)

to authenticate a user. These approaches may also be combined resulting in a so-called multi-factor authentication. In REACTION, we only deal with the first two approaches as the latter is more complex to deploy and introduces greater costs, e.g., for additional hardware like retina scanners or fingerprint

readers. In the following, we will briefly introduce the authentication technologies deployed in REACTION as well as how and where they are used.

5.1 User Name and Password

User name and password schemes are very popular for their ease of use. Users can easily transport them and developers do not have to implement complex mechanisms to verify the correctness of a password. However, problems arise when users choose inadequate passwords. “Inadequate” typically means that passwords are

- short (8 characters or less),
- easy to guess (related to the user),
- easy to find by trial and error (common words found in a dictionary),
- limited in the character set (only letters or only digits).

Even minor variations of common words, such as replacing letters with similar looking special characters like “`$pec!@1`” instead of “`special`”, are not safe as such patterns are also programmed into ’professional’ password crackers [19, 27, 29]. Another problem is password reuse, i.e., users choose the same password for different services. Choosing inadequate passwords or reusing existing ones is a strategy that is often used when users have to memorise many passwords [18] and are not informed about how to choose ’safe’ passwords [1]. Password reuse may put users at risk if their passwords are stolen and the thief simply tries them with popular web sites like Ebay or Amazon and in case might gain access to the users’ accounts.

Password security depends on a password’s length, its complexity, and the amount of entropy it contains. For example, if a chosen password has a length of 12 characters and may be composed of lower-case letters (26 choices), upper-case letters (26 choices), digits (10 choices), and special characters (32 choices), 94^{12} (approximately 10^{23}) different passwords are possible which can be encoded in 79 bit. Maximum security against password guessing attacks is only achieved if each character of the password was chosen uniformly and at random from the previously mentioned set of characters. Only then each password from the set of twelve-character passwords is equally likely to have been chosen. However, users tend to choose passwords they can remember, i.e., which are meaningful to them. Unfortunately, such passwords are neither uniformly chosen nor random and thus many potential passwords are never chosen by users meaning that far less passwords have to be tried by an attacker before he succeeds in his attack and learns a user’s password.

5.2 Public Key Cryptography

In order to use public key cryptography (PKC), a key pair comprised of a private and a public key needs to be generated — ideally by the intended owner of the key pair. The public key, as its name suggests, is intended for publication while the private key must be kept secret. In this scheme, an identity is represented by a public key. The private key, in turn, is used to prove that a claimed identity, i.e., the presented public key, really belongs to the claimant without disclosing the private key itself. An advantage of PKC is that the same key pair may be used for different services without compromising security — so ’key pair reuse’ is not a security issue but might be a privacy issue (see Section 4). Attacks like guessing a private key are infeasible as the length will be in the order of magnitude of 1024 bit or more (approximately 10^{307}). The sheer size also makes it virtually impossible to use trial and error to find the private key corresponding to a given public key. Using public key cryptography, all previously mentioned security problems with user name and password schemes are non-existent.

However, using PKC also has its price. Clearly, a key pair is too big to remember. Hence, it needs to be stored on some media. If the user wants to use the key pair from different places, she needs to carry the media with her. Standard software like browsers require users to install the key pair on the machine where the software runs. Apart from the usability issue of installing the key pair, it likely creates a security problem as users may also have to uninstall the key pair. And even if the user uninstalls the

key pair, she cannot be sure that no traces of her key pair are left unprotected on the machine from which the key pair was allegedly deleted.

5.3 Shared Key Cryptography

Shared key cryptography (SKC) (also called "secret key cryptography"[4]) can be seen as a mixture of the properties of user name and password schemes and those of PKC. The security of SKC is based on keeping a certain cryptographic key secret between a certain group, say, a sender and a receiver. This is similar to a user agreeing on a password with another party — both will know the password and it is expected that nobody will disclose it to others. If the key is used only between two parties then each will recognise its respective communication partner and the shared key can be seen as an identity. Note that in contrast to PKC, 'key reuse' in SKC poses the same risk as 'password reuse' from user name and password schemes.

The length of an SKC key is typically 128 bit or more (approximately 10^{37}) which makes it impossible to remember and also infeasible to guess or find through trial and error, like with keys for PKC. The advantage of SKC over PKC is that it requires less processing power and less storage for a comparable level of security. As previously mentioned, keys used in PKC are typically 308 digits long while comparable SKC keys only have 38 digits. Apart from requiring more space for the larger numbers —in the given example, space for additional 270 digits—, PKC also consumes more computing power to make the calculations with the larger numbers and naturally such computations take longer. This might not be an issue for PCs but it is one for smaller devices like smart phones or medical devices. These devices typically have less computing power and run on battery such that frequently running processes with high power consumption will significantly reduce their batteries' life.

Many of the medical devices considered by REACTION use a wireless connection to transmit their data to a base station, e.g., a PC. In order to secure such connections, SKC is typically incorporated in the employed wireless transport protocols, like ZigBee or Bluetooth. The purpose of the SKC key is twofold. Firstly, it is used to assure the base station that its received data is authentic, i.e., comes from a device known to the station. Secondly, it is used to keep the transmitted data confidential.

Keys for SKC schemes can also be derived from passwords by using so-called password-based key derivation functions, such as PBKDF2 [21]. The downside of this approach is that the security of any SKC scheme's underlying cryptographic primitives is based on the assumption that the keys are kept secret and are chosen uniformly and at random from the set of all possible cryptographic keys. However, when the key is derived from a password chosen by a user, randomness cannot be expected (see discussion in Section 5.1). Hence, only a fraction of the underlying key space will be reached [37]. This means that an adversary, trying to find the correct key, does not have to search the entire key space but only the subset that can be reached by password-based key derivation. In addition, the security then depends on the 'quality' of the user's chosen password and not on the secrecy of the key and its choosing which could open the door for successful password guessing attacks, again.

5.4 Discussion

The previously presented authentication technologies clearly do not provide a uniform level of security. For instance, password based schemes are significantly weaker than those based on SKC or PKC. Still, all of them are important to REACTION, for practical reasons or for reasons of usability.

As depicted in Figure 3, REACTION aims to integrate existing medical devices like (wireless) weight scales. Such devices can only be bought as is, i.e., the security measures implemented in these devices, if any, cannot be changed. This means it is only possible to make a 'sensible' choice among the available devices or to not use them at all, if their security is deemed unacceptable for the given use case. In any case, the a priori security of a non-programmable device cannot be improved in general by means

[4] "Secret" is kind of misleading, since the key is not only known by one party but by at least one other party, which is why we use the term "shared".

developed within REACTION. However, as discussed in [16], operational procedures can be devised to alleviate certain security shortcomings.

It is also worth noting that a medical device typically does not know the identity of its user as the device only authenticates to its base station —and probably vice versa— but does not require the user to authenticate to the device. Hence, the link between a patient and the data recorded and transmitted by the device is made by the assumption that a device will only be operated by a single person. In the primary care scenario, the link between a device and a user is maintained through a database entry that holds the patient's name and the device's serial number. The "one user per device" assumption is similar to the assumption that a mobile phone is only operated by its owner which in practice is mostly true, though naturally anyone in possession of the phone could use it. Thus, if a medical device's base station —or another component that sends the device's observation to the final destination— does not allow to change a patient's identity the receiver will have to assume a fixed identity and has no reliable way of distinguishing between different users' observations.

For the primary care scenario, being able to access the patient portal from any place has been deemed more important than the added security of digital certificates. Therefore, patients will be given instructions on how to choose 'good' passwords and, in addition, certain rules for passwords, such as minimum length, will be enforced by the system. Another reason for choosing a user name & password scheme was that users do not have to install anything extra on the device from which they access the portal. This should help to make the portal more accessible, especially for inexperienced users. It also does not require patients to buy new expensive hardware as even 5 years old PCs will be able to run modern browsers.

For the in-hospital scenario, the initial situation was different from the primary care scenario. A new application was created from scratch and thus security aspects could be integrated in the design right from the start. Since the target platform and target environment are more narrow in the in-hospital scenario, a tighter integration and transparent handling of user identities was possible, which turned out to be beneficial in terms of usability. This does not mean that the in-hospital application is monolithic. In fact, it has a flexible layered architecture which clearly separates the handling of application specific tasks and security specific ones, as can be seen in Figure 2. Through this architecture a number of principles known from software engineering could be realised [16], e.g., isolation and modularity, which paid off during the evolution of the Security Service App (SSA).

6 Profile Management

Profile management is an integral part of the Security Service App (SSA) that is used in the in-hospital scenario. The SSA is an app that can be used on any device running Android 2.2 or higher. It basically installs as a system-wide service that can be used by any other app to access web services, i.e., the SSA allows to send and receive SOAP messages. The connection between the SSA and its client app is made using interprocess communication (IPC) which allows apps to exchange data using messages (see Figure 2). This form of communication is beneficial as it supports loose coupling between the SSA and its client app while still being efficient. The messages exchanged between the SSA and its client are not send and received over some network connection but only written to or read from locally shared memory that is accessible to both apps. This produces far less overhead than standard network communication.

The SSA mainly exposes two simple methods to its clients as part of its interface[5]:

- `ISOAPMessage send (in String pURL, in String pOp, in ISOAPMessage pMsg)`
- `String login (in String pUsername, in String pPassword)`

The `send` method is straightforward: it takes a SOAP message `pMsg`, delivers it to the web service reachable under the URL `pUrl` (and an optional operation `pOp`), and returns the service's response

[5]The complete interface has more methods, e.g., for handling errors. However, these methods are of no concern here.

message . The SSA currently only supports the two most dominant transport protocols for web services, HTTP and HTTPS. However, the SSA is more than just a SOAP client, it also offers a flexible profile management. This profile management is made available to the SSA's client by a simple call to the `login` method. In order to login, a user name and password is required that will 'unlock' the user's profile. After successfully logging in, any subsequent client call to the `send` method will automatically apply the profile management, i.e., employ the settings from the user's profile. There is no explicit logout method for the SSA client, since this is realised by dropping the IPC connection. To switch profiles, another call to `login` is sufficient, i.e., the IPC connection need not be dropped and re-established.

In the following sections, we will present the profile management realised in the SSA in more detail and provide a discussion on its security.

6.1 Profiles

In REACTION, a profile is the basis for configuring the SSA on a mobile device that is employed in the in-hospital scenario. It basically consists of a number of identities and a policy that contains information on how and when to use the identities. A profile is always related to a specific user. While it is typically the case that profiles of different users will also contain different identities, the policy might be the same for a number of users, e.g., when a department wants to ensure uniform handling of identities for all members of the department. An identity from a profile will be represented by a digital certificate, though extensions like user name & password are certainly possible. The relationship between a user and a corresponding profile is shown in Figure 4. As shown in the figure, a virtual identity is the sum of a number of digital identities and together with a policy it forms a profile. The virtual identity in turn, can be seen as the user's virtual self that captures all real-world capabilities that the user needs in any digital (virtual) context.

Policy. Roughly speaking, a policy is divided into rules and their associated actions. A rule is defined in terms of a matcher for a URL — this is why we also speak of a "matching rule". Matching rules are evaluated when the SSA's client wants to access a URL. A rule may specify an exact URL to fire or it may use wildcard matching to subsume a number of URLs for which the same actions should be applied, e.g., to treat all URLs related to a given server in the same manner. In any case, only the first rule that

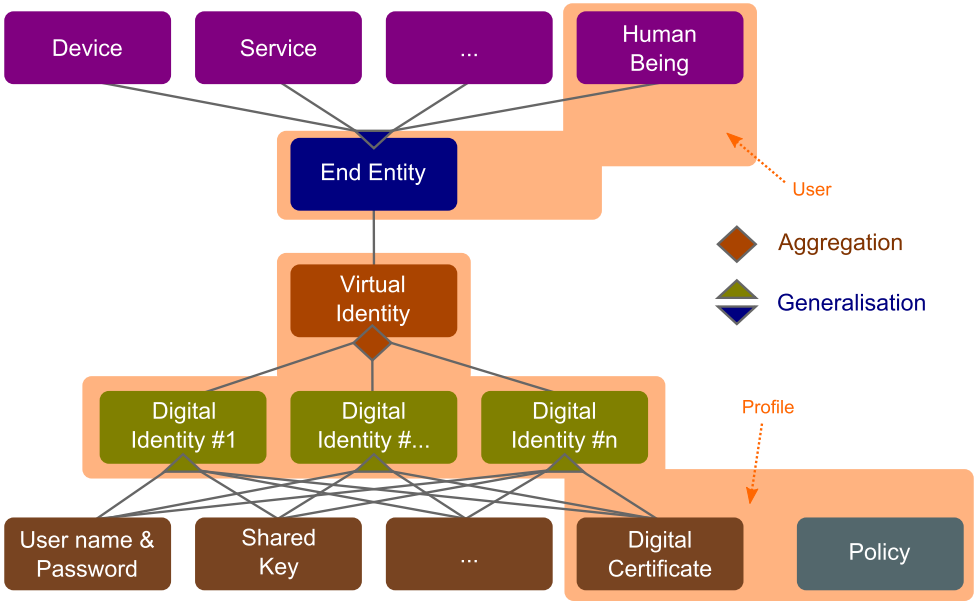

Figure 4: Relationship of constructs

matches a given URL will be executed. Any subsequent matching rule will be ignored. Actions to be performed within a rule can be:

- Identity selection
- Trust anchor selection
- URL transformation

Identity selection is straightforward: the identity for the given URL is selected by the user from the pool of available identities and will be used to authenticate to the communication partner. A given identity can be used for multiple services. However, vice versa, it is not possible to assign a service to more than one identity. In order to use different identities for the same URL different profiles must be used.

Trust anchor selection means that for the validation of an identity (see [16], Section 8.5.1) presented by the communication partner, the set of trust anchors used to verify the authenticity of the presented identity can be selected per URL(s). Trust anchors are typically identities of certification authorities issuing digital identities but can also be identities of end entities trusted by the user.

URL transformation can be used to 'rewrite' destination URLs. For instance, if an application wants to access a given endpoint using HTTP, the policy can be used to change the protocol to the more secure HTTPS. In a similar way hostnames, destination ports, and paths may be modified. For this a simple transformation language was introduced that complements the matching rules. In the following, we introduce the language using an example that also shows how matching rules work.

Assume that the following end point of a web service called *PatientService* is accessed by some app that employs the SSA.

```
http://192.168.0.2:8080/glucomansys-ws/services/PatientService
```

Now suppose the service provider has recently started to offer secure access to its service under the following endpoint — sequences shown in red are the ones that deviate from the original URL.

```
https://10.0.2.2:8443/glucomansys-ws/secure/services/PatientService
```

Normally, in order to use the new endpoint, either the app needs to be reconfigured or even recompiled. However, using the SSA's policy the app need not be changed at all. By using wildcard matching and transformation, the original URL can be turned into the desired target URL as shown in Example 5.

The wildcard character "*" from the matching rule matches zero or more characters and any match can be referenced as "\n" in a transformation, where n is an integer greater or equal to 1 and marks the n-th wildcard match in the rule. Note that the right-hand side of each processing step shown in Example 5 must match a substring of the original URL in order to proceed. If any step would result in a mismatch, the rule would not fire and the next rule would be evaluated. If no rule fires, the SOAP message is not send and an error is returned to the SSA's client.

In contrast to the default matching strategy in regular expressions, the matching here is performed non-greedy. That is, the matching algorithm will not look for the largest sequence that matches a given matching rule's fragment but stop with the first match encountered. For instance, if the URL would look like "`http://somedomain.com/somepath/somefile`" and the matcher would be "`*some`" then non-greedy matching matches "`http://`*`some`*", i.e., it stops with the first "some" encountered, while greedy matching matches as much as possible and only stops with the last "some", i.e., it would match `http://somedomain.com/somepath/`*`some`*. Non-greedy matching seems to be more user-friendly and 'natural' than the greedy variant as the effects of greedy matching often surprise users unfamiliar with the concept of greedy matching. Since the matching rules can be critical in terms of security, non-greedy matching was chosen to reduce the chance of errors that could be induced through greedy matching.

Original URL	`http://192.168.0.2:8080/glucomansys-ws/services/PatientService`
Matching Rule	`http*://*:*/glucomansys-ws/*`
Transformation	`https://10.0.2.2:8443/glucomansys-ws/secure/\4`

Processing steps
Substring “http” matches “http”
1st “*” matches “” (empty string)
Substring “://” matches “://”
2nd “*” matches “192.168.0.2”
Substring “:” matches “:”
3rd “*” matches “8080”
Substring “/glucomansys-ws/” matches “/glucomansys-ws/”
4th “*” matches “services/PatientService”
Wildcard match 1 is not referenced in the transformation and hence omitted from the result
Wildcard match 2 is not referenced in the transformation and hence omitted from the result
Wildcard match 3 is not referenced in the transformation and hence omitted from the result
Wildcard match 4 is referenced in the transformation by \4 and hence kept for the result
Copy Transformation to Result and replace any reference “\n” by Match n

Result	`https://10.0.2.2:8443/glucomansys-ws/secure/services/PatientService`

Example 5: Processing steps for a matching rule with corresponding transformation

6.2 Profile Management Service

The profile management service (PMS) was introduced to allow efficient management of profiles and resources available on the mobile device. The PMS is a client-server application whose client part is integrated in the SSA and whose server component is part of the Glucose Management System (GMS). In the following, when we speak of the PMS we mean its server part.

The PMS is accessed through a web service that allows the SSA to access and download user profiles from the server after successful authentication of the user. A user authenticates to the PMS by sending a user name and password. On success, the PMS responds with the user's profile. In this case, the SSA stores the profile on the device, encrypted with the user's password. The profile stays on the device for a configurable amount of time. During this time, the SSA is independent of the PMS with respect to profiles already downloaded, i.e., the SSA does not need to contact the PMS. When in 'offline mode' the user authenticates through a successful decryption of his/her profile. If the SSA is being removed from the device, all its configuration data including user profiles are removed. Figure 6 shows a sketch of the PMS and its integration into the SSA.

The PMS also offers services to applications that require information about the currently logged in user. This functionality is not part of the SSA as it is application-specific functionality that is meant for a specific context, such as the GMS. Applications may use these services by simply making web service calls. The information provided by the PMS with respect to a given user identity are:

- user name
- first name
- last name
- assigned ward
- roles (permissions)

The user name is the identifier used for the user's login to the SSA. The first and last name are the user's real world name. The assigned ward is the department in the hospital where the user is working and

the roles give the user's permissions with respect to the GMS. A role could be, for instance, "physician" or "nurse" where the former may imply the permission to change a patient's therapy plan while the latter may only imply permission to read the therapy plan. A user may have multiple roles or none at all. In the latter case, the user will only have permission to access publicly available information. The exact permissions cannot be extracted from the role identifiers as they are managed internally and may also change without notice. However, for an application that wants to customise its behaviour and possibly its GUI based on a user's roles, it might be enough to know the role identifier as such a specialised application is expected to have a basic understanding of duties related to a given role.

6.3 Device Profile

In addition to user-specific profiles, we also have so-called device profiles. Device profiles are conceptually similar to user profiles in the sense that they also consist of an identity —for the device— and a policy for configuration purposes. The device identity is assumed to be unique and assigned to exactly one device. The policy of a device captures various settings for the management of user profiles.

As mentioned in the previous section, the SSA may store user profiles on the device for a configurable amount of time. This time can be set in the device policy and it determines the maximum age of any user profile. If a profile is deemed too old, the locally cached profile will be replaced by a fresh copy requested from the PMS. The age of a profile is determined in absolute terms, i.e., the age is not reset when a profile was used since this may postpone the refresh of a profile indefinitely. In addition, the maximum number of cached profiles can be set in the device policy.

Clearly, the PMS is not a public service and neither one that is world-wide available. Hence, every organisation or even every department of an organisation is expected to have its own PMS installation. Therefore, the end point for the PMS can also be configured in the device policy to adapt to the local environment.

Similar to user policies, a device profile may also include a list of trust anchors to allow authentication of communication partners contacted via HTTPS, e.g., the PMS server. Although the end point for the PMS may be configured to use HTTP instead of HTTPS, it is highly discouraged to do this as this means that user names and passwords would be sent in clear and this may jeopardise not only the user's security but also the patient's privacy and safety.

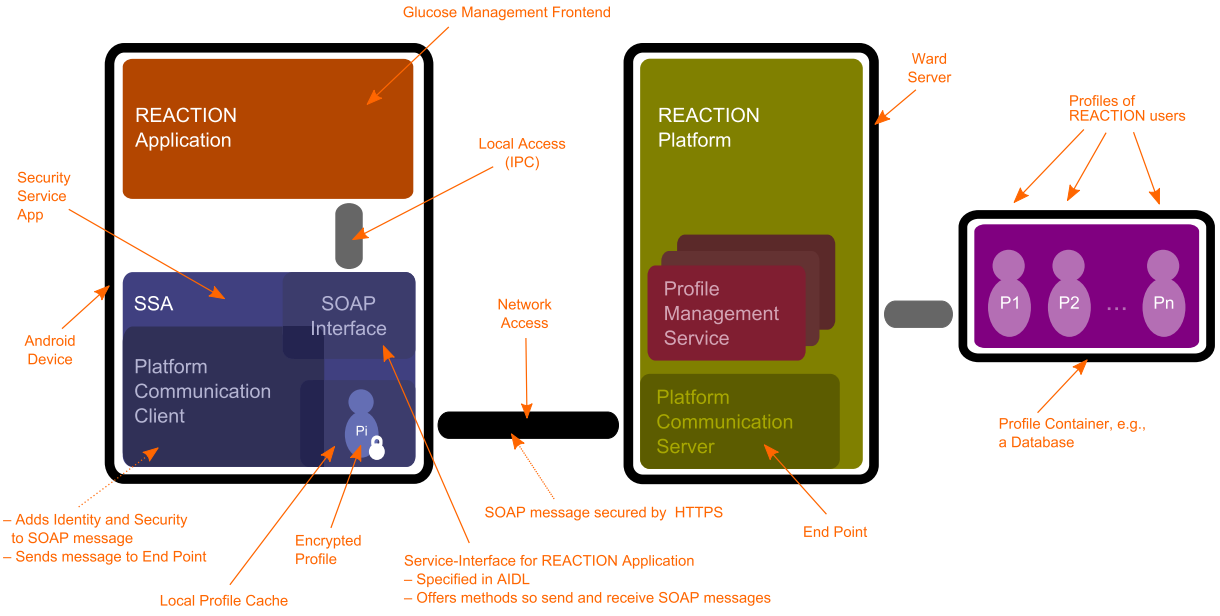

Figure 6: Sketch of the Security Service App and the Profile Management System

Note, that the device profiles described in here should not be confused with device specialisations, such as ISO/IEEE Std 11073-10417 for glucose monitor devices [26], or transport layer protocols, such as the Bluetooth Health Device Profile [32]. These standards are important in the medical context and are supported by the Hydra framework [9] but have already been treated in [8].

6.4 Security of Profiles

In this section, we will introduce the security features of the PMS, the suggested default security settings, and the rationales for design decisions that had to balance usability and security.

Initialisation. After the SSA is installed on a mobile Android device, it needs to be initialised. Initialisation essentially means to provide the SSA with a device profile. For this, an SD card holding the device profile must be inserted into the mobile device. When the SSA service is started, it first checks if a profile is available on the device's internal storage and if so it takes this profile. If not, it looks for a profile on any SD card. If a profile is found on an SD card, it is copied to the device's protected internal application storage and used henceforth. Note that after a profile is stored on the device, a device profile on any external storage will be ignored. This is a security precaution that should make it harder for attackers to overwrite existing device profiles by just inserting their own SD card into temporarily unattended devices and 're-initialise' the SSA. An internally stored device profile can only be removed by deleting the SSA altogether or by infecting the device with malware that circumvents the file permissions of Android's Operating System. All of these attacks are assumed to be noticeable given that a device is only shortly unattended — a stranger connecting his laptop to a ward tablet PC should arouse suspicion. If the device had been reported missing for a longer period, it should be regarded as compromised. Procedures to deal with allegedly compromised devices will be discussed below.

A device profile that is stored on the device's internal storage is encrypted. However, this encryption offers only weak protection against attacks since the SSA uses a key that is hardcoded into the SSA's program code. Although it would have been possible to use a pass phrase to better protect the device profile, it was deemed too user-unfriendly as for every start of the SSA users might have to enter two passwords in quick succession, one for the device profile and another one for their own profile. In addition, the password for a device profile would have to be shared by users of the device and thus, would hardly be a secret. In addition, it is not guaranteed that users will always operate the same device and thus they would have to remember additional passwords for different devices which would be user-unfriendly, too. Therefore, the security employed for the device profile is only meant to thwart naive, simple attacks but not to offer sufficient resistance against determined attackers. Having said this, the device profile will only be used as an additional factor for security but is not intended to be self-sufficient. □

Profile Management Service. Clearly, the device profile should not be regarded as a high-security token that would allow access to sensitive information, such as electronic patient records. The purpose of the device profile, or rather the device identity, is to gain access to the PMS. However, this does not mean that anyone in possession of a valid device identity would be able to access any user profile. In fact, in order to gain access to a user profile, two factors are required. The first factor is an initialised device, i.e., one that holds a device profile. The second factor is a user name and password. Only if a user can present both factors, he/she will be able to successfully authenticate to the PMS and gain access to the profile corresponding to the user name.

If an attacker would get hold of a device identity, e.g., by copying it from a (temporarily) stolen or lost device, she would be able to mount password guessing attacks. However, this can be noticed at the PMS and the device identity can subsequently be blacklisted, effectively locking out the attacker. In fact, the ability to lockout attackers easily and effectively after a device compromise was one of the main reasons for introducing device identities. Note, however, that this mitigation strategy would *not work* if the PMS is made accessible through HTTP (c.f. Section 6.3) as in this case no device identity would be sent by the PMS client and consequently, no device authentication would be possible. □

Cached profiles. Caching of user profiles was introduced to the PMS system in order to allow for quicker user logins, e.g., in case a user was automatically logged out after some time of inactivity. Cached profiles are stored in a local storage area of the Android device that hosts the SSA. This storage area is an app-specific private area, i.e., only the SSA has the permission to read and write data from/to this area. Access to the private area is controlled by way of file permissions which are enforced by the Android OS. Therefore, any other app that would try to gain access to the private area would have to circumvent the operating system's security. However, if the Android device would be connected to a PC, the file system would be exposed and the data could be read. Since an attacker would make use of this method when he got hold of a device, the profiles require extra protection. Therefore, profiles are only stored in encrypted form on the device. Currently, encryption is done using the AES algorithm [25], where the key is derived from the user's password using a password-based key derivation function, such as PBKDF2 [21].

Once an attacker got hold of a cached profile, its confidentiality will largely depend on the quality of the user's password (see Section 5.1). In an attack, the attacker would use trial and error to find the user's password by deriving a key from the guessed password and use this key to try the decryption in each trial. Such attacks can be sped up by using pre-computed keys derived from words found in dictionaries. In order to raise the bar for pre-computation attacks —sometimes also called 'offline' attacks—, a so-called *salt* had been introduced. A salt is essentially a public per-user value of random data that becomes an additional parameter to the key derivation function. Since the salt is random, a pre-computed list is not attractive for attacks as a complete dictionary would have to be targeted to one specific salt. Therefore, attackers will have to use 'online' attacks, i.e., may only start their password guessing as soon as they learned the salt. This clearly delays the attack but it may still ultimately succeed if the user chose a weak password. However, the delay might be enough to notice the theft/loss of a device holding the user profile and allow administrators to revoke the user's identity and lockout the attacker before he can compromise the GMS. □

7 Access Control

In REACTION, sensitive data, such as patient data or medical history, are processed. To ensure that these data can only be read or modified by authorised entities, access control is employed. Access control is the process of granting or denying access to resources based on the requester's identity and his or her associated permissions. For example, in a hospital it must be ensured that only authorised personnel, such as health professionals, can read or modify patient data. In Section 6 of [16], access control is identified as one of the main security objectives within REACTION.

In general, access control is applied in IT systems to control *who* can access *what* and *how*. The resources to protect (the *what*), such as data but also processes or web services are called objects. The *who* is represented by entities, such as users or processes that want to access these objects. And finally, the *how* represents the way an object is accessed, e.g., read, write, execute. Typically, specific rights to access an object are modelled as *permissions*.

In [16] we presented different strategies, models, standards, and concepts for access control. Regarding these findings, it was concluded that both Discretionary Access Control and Mandatory Access Control do not have the flexibility and ease of administration required by the use cases of REACTION. In contrast, Role-Based Access Control (RBAC) "allows and promotes the central administration of an organisational specific security policy" [17]. Therefore, RBAC was chosen to implement the access control in REACTION since it seems to be best suited to map and reflect the organisational structures covered in this project.

7.1 Role-Based Access Control

The basic idea of RBAC is that permissions are not assigned to single entities but to a special construct, the so-called *roles*. Each entity is assigned a set of roles according to their tasks within the organisation

and each role is then assigned to a corresponding set of permissions which is required by the subject to fulfil her tasks. In a hospital, for example, a nurse requires certain permissions to carry out work-specific activities. Instead of assigning permissions to every nurse, in RBAC permissions are assigned to a *nurse* role which is, in turn, assigned to all nurses. Therefore, nurses' permissions need not be managed on an individual basis but can be centrally managed through the *nurse* role which simplifies administration.

A formal model for RBAC was introduced by Ferraiolo and Kuhn [17]. In 2004, the American National Standards Institute (ANSI) published the RBAC standard INCITS 359-2004 [6]. This standard describes a reference model and functional specifications of RBAC features.

In RBAC, the following sets can be identified:

Users. In RBAC, the definition of users is not restricted to human beings but also encompasses machines, software, networks, or intelligent autonomous agents. This is contrary to the definition of users from section 4.3 which refers to users of the REACTION platform that are assumed to be humans only. In the following we will use the RBAC definition of users. □

Roles. A role represents a job function or position in the context of an organisation. It incorporates the authority and responsibility granted to the user that is assigned to the role. In other words, a role is a collection of access rights needed to perform a certain job function. □

Permissions. A permission captures the fact that a certain operation can be performed on one or more protected objects/resources. Often permissions are modelled as an (*Object*, *Operation*)-tuple. In RBAC, permissions are positive, i.e., they define what a subject is allowed to access and not what it is not allowed to. □

For a better overview, Sandhu et al. [28] categorised RBAC into four sub-models. The core model, referred to RBAC_0, encompasses theses sets combined with the following two relations:

Role Assignment. The role assignment is a many-to-many relation between elements of the set of users and the set of roles. It is used to express which roles have been assigned to which user. For example, the relation $(\{\textit{Alice}, \textit{Bob}\}, \{\textit{Nurse}\})$ says that both Alice and Bob have been assigned the role *Nurse*. □

Permission Assignment. The permission assignment is a many-to-many relation between the set of roles and the set of permissions. With this relation it can be defined which role is allowed to access which objects. For example, the relation $(\{\textit{Nurse}\}, \{(\textit{file}_0, \textit{read})\})$ states that all users that own the *Nurse* role are allowed to *read* the object $\textit{file}_0$. □

The core model is defined in terms of individual users being assigned to roles and permissions being assigned to roles. As such, a role is a means for naming many-to-many relationships among individual users and permissions.

The core model can be extended to allow for role hierarchies. In this case, Sandhu et al. call the extended model RBAC_1 [28]:

Role hierarchies. In terms of permissions, role hierarchies define an inheritance relation among roles and "are a natural means of structuring roles to reflect an organization's lines of authority and responsibility" [6]. For example, the role *administrative nurse* "inherits" the role *nurse*, if all permissions of the role *nurse* are also allowed for the role *administrative nurse*.

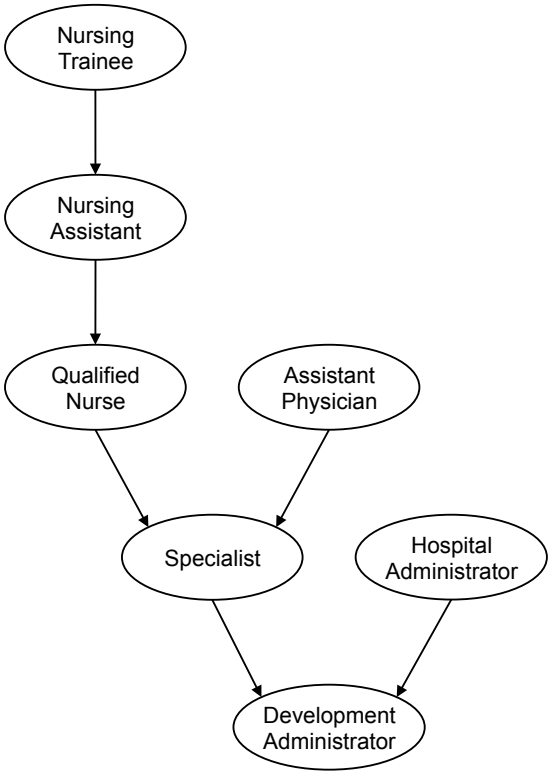

Figure 7: Example role hierarchy of a hospital

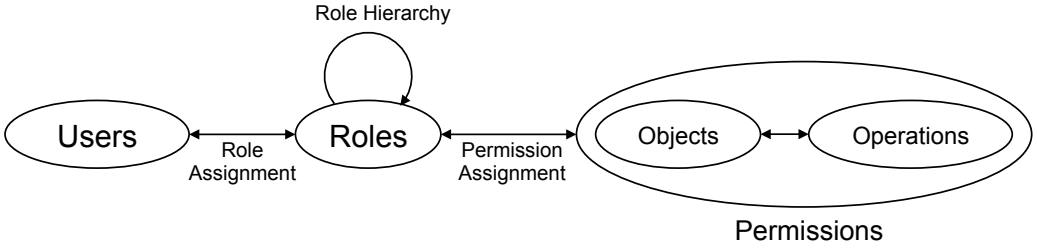

Figure 8: Sets and relations of the RBAC core model extended by role hierarchies

Role hierarchies allow to map hierarchically organised tasks and responsibility of real organisational structures [14, 28]. For example[6], in a hospital the hierarchy of treating physicians could be modelled as *Qualified Nurse* $\succeq$ *Nursing Assistant* $\succeq$ *Nursing Trainee*. The tree representation of an example role hierarchy within a hospital is shown in Figure 7.

Multiple inheritance, i.e., the ability to inherit permissions from two or more roles, is possible. This allows for combining multiple roles and, thus, provides additional options to represent organisations and business structures.

Figure 8 depicts the RBAC core model with the extension of role hierarchies. □

[6]Here, $A \succeq B$ represents a role inheritance relation between A and B that implies that role A inherits all permissions assigned to role B.

The core model can also be extended by constraints. This is independent of the extension of role hierarchies. If the core model is extended by constraints, Sandhu et al. call this extended model $RBAC_2$ [28]:

Constraints. Constraints are additional access restrictions based on context information. With constraints, is it possible to express complex access control rules that rely on information like time, location, and access history for instance. In REACTION, an example that shows the need of additional constraints is that in a hospital, some nurses should be granted additional permission depending on their experience which could be measured by their duration of service or certification. Another example in which additional constraints could ease administration in the hospital, is the restriction that a physician should only be able to treat patients that are assigned to her own ward.

Both Ferraiolo and Kuhn [17] as well as Sandhu et al. [28] introduced two kinds of constraints with practical relevance. Both deal with the principle of *separation of duty*. According to this principle, it should not be possible that a subject is assigned to roles that are mutual exclusive. For example, a physician should not be able to prescribe drugs to herself. □

It should be mentioned that if the core model is extended by both role hierarchies and constraints, Sandhu et al. [28] call this extended model $RBAC_3$ [28].

7.1.1 Access Control in REACTION

In order to achieve the flexibility to best match the organisational structures concerned in this project, the RBAC core model with the extension of role hierarchies and constraints will be applied. The sets formally described in the previous section can be mapped to the following entities of REACTION:

Users. In REACTION, users are the entities that use the web services. However, these entities will not only be human users such as physicians, nurses, and patients but also information processing units such as platform internal processes or mobile devices. Each user will be represented by one or more digital identities. How digital identities are formed within REACTION is described in section 4.4. □

Roles. Roles can be seen as a set of permissions a user or a set of users can perform within the context of REACTION. Here, roles will be defined with regard to the underlying medical facility and the tasks and duties of the facility's users. In case of a hospital, some roles could for example be *Assistant Physician* or *Qualified Nurse*. □

Permissions. Since the REACTION platform is implemented as a service oriented architecture, the resources to be secured are web services. A permission is modelled as a tuple comprised of an object and an operation. In this case, the object is represented by a web service endpoint, i.e., the web service's URL and the operation is represented by one of the operations offered by the web service.

An example permission is the *registerPatient* operation offered by the GMS *PatientManagementWebservice* which can be used to register new patients in the hospital system. □

Role hierarchies. In addition, role hierarchies will be implemented for REACTION to allow a mapping of the hierarchical structures existing in medical facilities, such as hospitals and primary care centres. For example, in a hospital the role *Specialist* could inherit the role *Assistant Physician* and, thus, transitively own all permissions assigned to the latter. □

Constraints. Context constraints, see $RBAC_2$, will also be implemented. Especially, the principle of *separation of duty* can be practical and necessary for reasons of security and privacy. For example, a physician should not be able to treat patients that do not fall into his/her area of competence, i.e. he/she should only be able to treat patients assigned to his/her own ward. □

7.2 Access Control Policy Languages

The rules according to which access control must adhere are usually defined in policies. In order to describe and enforce access control policies, a formalised language is required. On the one hand, an access control policy language has to be flexible and expressive and thus, provide means to define a variety of different rules. On the other hand, it should be simple and easy to administrate.

For REACTION, three main requirements for a suitable policy language were identified:

(1) Expressiveness. The policy language used in REACTION shall be flexible enough to support all kinds of organisational control principles and workflows. It must provide means to express all required objects worthy of protection. For the enforcement of a provider's security requirements, a flexible way to model permissions and policy rules is required. Furthermore, the language shall not constrain the RBAC model. □

(2) Ease of Administration. In the medical context it can be assumed that the sets of subjects and objects are dynamic. For example in a hospital, health professionals will be employed or dismissed with some dynamics. This does not only require for scalability but also for simple administration.

Furthermore, improper administration may cause severe security breaches. The administration of access control systems is a critical task and suitable administration models and methods have to be considered. □

(3) Extensibility. Expressiveness may be sufficient to describe all access control rules required. However, the formulation of complex rules might be complicated and result in expensive and error-prone administration. Therefore, existing concepts that extend the RBAC model —such as role hierarchies or constraints— shall be supported to allow for more sophisticated access restrictions. □

Many different policy languages exist, varying with regard to the underlying access control model and system. For REACTION two access control policy languages were evaluated with regard to the requirements stated above. In the following, these two languages will be shortly presented. Subsequently, an evaluation, its results, and the language applied in REACTION will be presented.

7.2.1 The Extensible Access Control Markup Language (XACML)

The Extensible Access Control Markup Language (XACML) is a standard describing a declarative access control policy language based on XML. XACML can also be used to implement RBAC. In 2005 the OASIS[7] standardisation committee ratified Version 2.0 of the standard [23]. XACML is used in distributed systems like web service oriented architectures. Besides language constructs for access rule descriptions, XACML also provides a model for communicating standardised messages with the access control system.

Besides syntax and semantics for the policy language and the standardised format for access control decision requests and responses, XACML also defines modular entities of a corresponding XACML system as well as the information flow between these components that leads to the access control decision.

The following entities are part of the access control system:

[7] https://www.oasis-open.org/

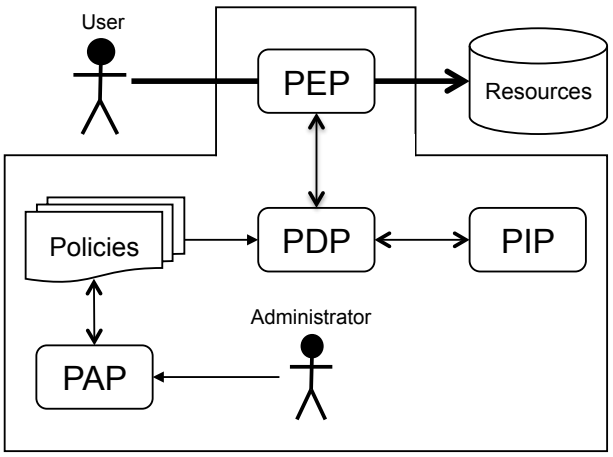

Figure 9: Information flow in XACML

Policy Administration Point (PAP) Provides an interface for managing access control policies.

Policy Enforcement Point (PEP) The PEP enforces an access control decision taken by the PDP. The enforcement results in either granting or denying access to the requested resource.

Policy Decision Point (PDP) Makes an access control decision based on the policies provided by the PAP and additional information provided by the PIP. The result of a requested access control decision in XACML can be *allow* for granted access, *deny* for denied access, *indeterminate* for erroneous requests, or *not applicable* for requests for which no decision could be made.

Policy Information Point (PIP) Evaluates attributes in compliance with additional information about resources or general context information, such as date, time or IP address.

The information flow in XACML is depicted in Figure 9. Initially, the administrator uses the PAP to define the access control rules for the objects to be secured. The PAP persistently stores these rules in policies. Now, when the user wants to access a resource, she requests access at the PEP. Here, the PEP is a gateway between the resources the user wants to access and the policy holding the access control rules. Rule enforcement and access control decision are logically separated, i.e., the PDP does not decide independently if access should be granted or not. Instead, a request is sent to the PDP. The PDP consolidates information from the PIP as well as rules defined in the policies to reach a decision. The decision is sent back to the PEP which finally enforces it. In a web service oriented architecture, the PEP will forward the user's web service request to the endpoint if the access was granted. The user will learn the decision implicitly through the web service's response if access was granted or explicitly through an error message if access was denied.

Several commercial and free XACML implementations exists. A widely known Java reference implementation of the standard is offered by SUN[8]. However, this implementation has not been updated since 2006. Nonetheless, XACML is an accepted industry standard which is, for example, used in IBM's WebSphere[9] middleware or RedHat's JBoss[10] application server .

Even though XACML can be extended to RBAC principles by the RBAC profile [5], this profile only supports role hierarchies but no constraints. Since XACML is a low-level language, high-level access controls principles, such as *separation of duty* or history-based control, can only be expressed by complex and verbose sets of rules which hinder administration. Furthermore, some principles cannot be expressed directly but rather manually by the administrator or external tools. Administrators have to deal

[8] http://sunxacml.sourceforge.net
[9] http://www.ibm.com/software/de/websphere
[10] http://www.jboss.org

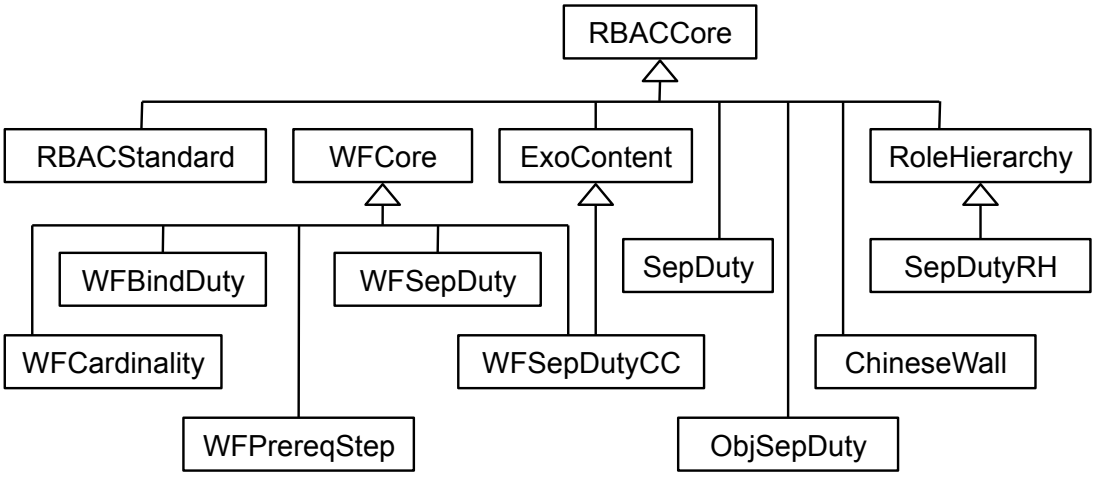

Figure 10: Dependency hierarchy for OPL modules [3]

with a lot of low-level language constructs which opens the door for human mistakes that make the policy inconsistent or invalid.

7.2.2 The ORKA Policy Language (OPL)

Alm et al. [2] compared existing access control policy languages with regard to the competing properties of complexity and expressiveness. They concluded that none of the considered languages meets these requirements. Therefore, they introduced the ORKA Policy Language (OPL) [4]. OPL is an XML-based language for policies based on the RBAC model. It "aims at bridging the gab between the expressiveness required by organisations and the simplicity needed for practical administration" [3]. Compared with other approaches, OPL reduces the complexity in terms of lines of code required to model an access control rule. At the same time, the expressiveness is increased and, thus, allows to implement a multitude of different concepts, such as Chinese-Wall-Policies [12] or *separation of duty* constraints. A formal definition of OPL can be found in [3].

The OPL policy model is defined as a set of policy modules. Each policy module is focusing on a certain access control principle. For example, the core RBAC module $RBAC_0$ is part of OPL's *RBACCoreModule*.

For the specification of the policy model, the Object Z notation was used. This notation is both formal and object-oriented and "it provides means to cope with complexity, flexibility, and feature reuse [and it] leaves no room for ambiguities and it can be subject to formal reasoning" [3].

New policy modules may be introduced or existing ones may be refined. For example, the *SepDutyModule* extends the *RBACCoreModule* by *separation of duty* constraints. Thereby, extensibility is achieved. The expressiveness of OPL is only limited by the expressiveness of underlying Object Z notion which is based on first-order logic and set theory. Furthermore, since modularisation is one of the key countermeasures against complexity, complexity is reduced. Figure 10 shows the module hierarchy of OPL.

Besides *RBACCoreModule*, which offers the functionality of the RBAC core model, OPL offers many other modules which allow to describe miscellaneous policies and supplementary concepts of RBAC. For example, by using the module *RoleHierarchyModule* role hierarchies can be described. Most notably, OPL is characterised by its compact modelling and its extensive expressiveness.

7.2.3 Evaluation and selection of a suitable policy language for REACTION

Considering these two policy languages with respect to the requirement of expressiveness (1), it can be concluded that XACML and OPL are comparable. This point of view is also shared by the findings in [4].

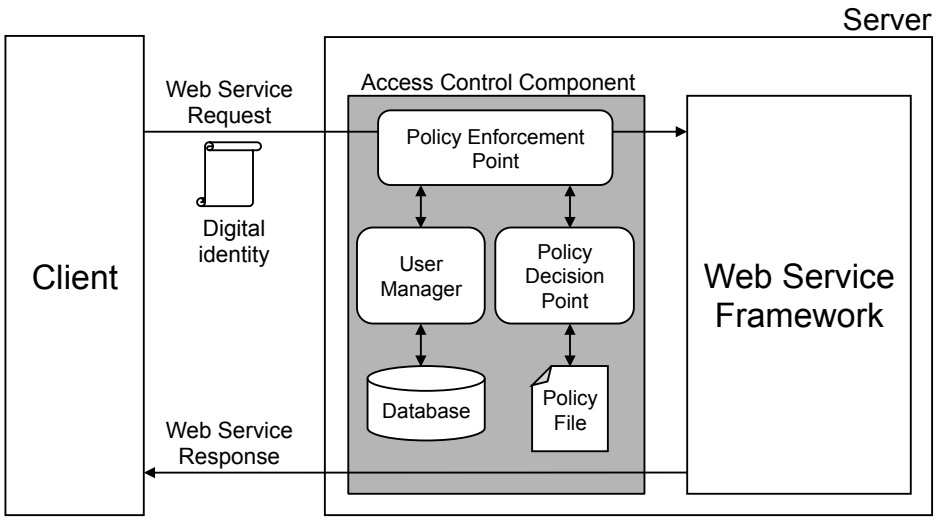

Figure 11: Architecture and integration of the Access Control Component

The ease of administration (2), however, is clearly an advantage of OPL. The high level of flexibility and expressiveness provided by XACML comes at the cost of complexity and verbosity, i.e., it is difficult to work with the language or policy files directly since it can hardly be read by humans. Some tools for XACML editing exist but Lorch et al. state that "even with good tools in place, there is an inherent semantic complexity that's separate from the syntactic complications" [22]. OPL, however, is easier to administrate since it comes with a syntax that is more simple and, for example, can be managed with less XML metadata than XACML.

Concerning extensibility (3), the module-based architecture of OPL provides the means to easily insert or extend modules from the rich set of available modules or to implement new modules to provide ways to deploy additional access control principles. Due to its expressiveness and flexibility, RBAC extensions like role hierarchies and constraints can, in general, also be expressed in XACML. However, these principles can only be implemented by complex and verbose sets of rules which, in turn, lower the ease of administration.

In summary, the benefits of XACML are clearly its expressiveness and the fact that it includes standardised components, formats, and protocols. It is accepted in the industry and several implementations and tools exist. However, when it comes to administration and the extensibility of additional RBAC concepts, XACML shows its weaknesses.

OPL not only provides the necessary expressiveness, it is also simple, readable, and easily extensible without introducing new complexity. Since it is dealt with sensitive data in REACTION, a misconfigured access control policy can potentially harm privacy and even safety. In order to lower the risk of misconfiguration, OPL is chosen as the basis for the access control policy language that is developed and applied in REACTION.

7.3 Implementation

This section gives a high-level overview of the access control implementation for REACTION. Figure 11 depicts the architecture of the Access Control Component (ACC) and how it is integrated in the platform's architecture. The individual components are described in the following subsections.

7.3.1 User Manager

In the User Manager component, the role assignment of the RBAC core model, see Section 7.1, is managed. In REACTION, users are represented by digital identities, see Section 4.4. The User Manager

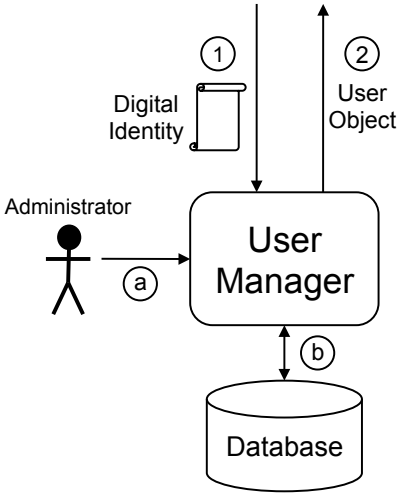

Figure 12: Workflows in the User Manager

is an interface between the identity management and access control. Here, a user's digital identity is transformed into an access control specific user.

To each identity an arbitrary set of roles as well as a unique user name can be assigned. Both, the set of roles and the user name are optional. The managed data and the corresponding assignments are persistently stored in a database. The user name is used as a reference to identify the user in the GMS to retrieve the information provided by the PMS, see Section 6.2.

The User Manager's workflow is shown in Figure 12. Initially, an administrator has to register digital identities to which roles and a unique user name can be assigned (a). The User Manager stores this information in a database (b). Later on, the administrator can add and delete identities or modify the corresponding assignments.

In a separate process, the User Manager can be requested to return a User object for a given digital identity (1). The User object (2) encapsulates the assigned set of roles and the user name. This software representation of the user's identity can be used for further processing in the Policy Decision Point, i.e., to evaluate access control decisions .

With the User Manager, the role and permission assignments of the RBAC core model are decoupled. On the one hand, it is assumed that users frequently join or leave the system. In addition, it is assumed that these users often change their positions and thus, their tasks and duties. Therefore, the role assignment can be seen as a dynamic process that is done with the help of the User Manager. On the other hand, it is assumed that the permissions assigned to a specific organisational position, i.e., a role, change rather infrequently. So, the permission assignment is assumed to be a quite static process and hence can be managed within a policy file. By decoupling these assignments, flexibility is gained and the policy, which is a very sensitive component, has to be edited less often and, therefore, the chance of misconfiguration is reduced.

For REACTION, we introduce a special role that is automatically assigned to each user. This role is called the *empty role*. This role is a root element in the role hierarchy from which all roles implicitly inherit. Even if the user does not provide a digital identity, e.g., when sending the request via HTTP, the User Manager will return a User object that does at least have the empty role assigned. This technical extension was implemented to facilitate access to non-critical web services without the need of a digital identity. Considering tree representation of the role hierarchy example shown in Figure 7, the empty role can be represented as a root node which is inherited by the roles *Nursing Trainee*, *Assistant Physician* and *Hospital Administrator*.

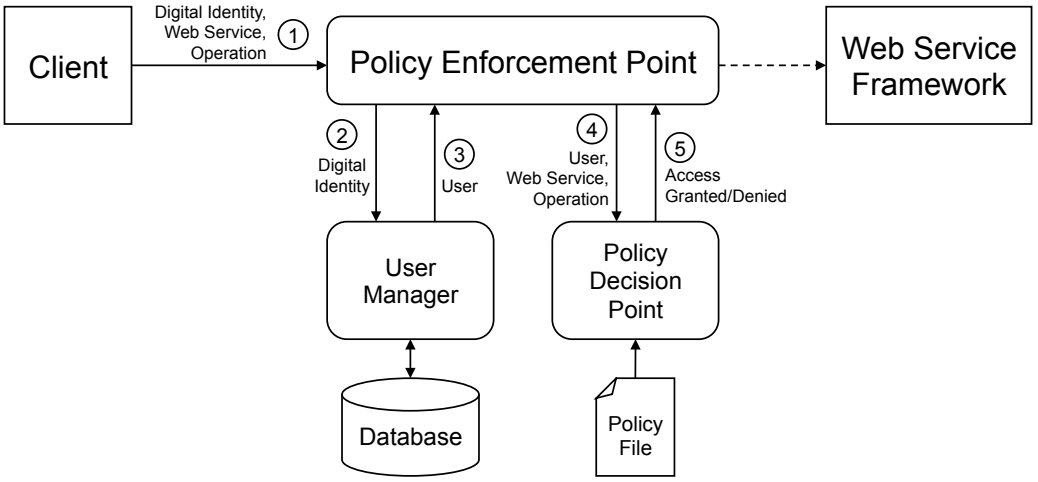

Figure 13: Information flow within the Access Control component

7.3.2 Policy Decision Point

The Policy Decision Point (PDP) is a system component that evaluates the access control policy and renders an authorisation decision. Based on this decision the Policy Enforcement Point (PEP) will either grant or deny the user's access to the requested resource.

The information flow of the access control component is shown in Figure 13. When a web service request is intercepted by the PEP, the requested web service URI, the corresponding web service operation and, the digital identity of the requester is extracted (1). For the evaluation of an access control decision, a User object is required. By providing the identity to the User Manager (2), a User object can be retrieved (3). This User object encapsulates the roles assigned to the requester's digital identity. If the web service request was not send via HTTPS and, thus without a digital identity, the user is automatically assigned the *empty role*, see 7.3.1

The PEP requests the PDP to decide if the requester is allowed to access the requested web service (4). For this access control evaluation, three parameters have to be provided to the PDP:

1. A User object holding the user's assigned roles,
2. the requested web service's URI, and
3. the requested web service's operation.

The PEP's answer/decision is either *access granted* or *access denied* (5). To make the decision the PEP first has to read the policy file. Access is granted if at least one permission, matching the requested web service URI and operation, is assigned to any role the user owns, either directly or through role inheritance. Otherwise, the decision results in denied access.

If the web service request was send insecurely without HTTPS, access will only be granted if the required permission was explicitly assigned to the *empty role*.

Constraints. The implementation of RBAC constraints was not finalised during the time this document was written. Therefore, the description of the implementation of context constraints for REACTION will be omitted in this document. The implementation of context constraints will be based on the concept introduced by Strembeck and Neumann [34] who give a formal definition of elements that can be used to model constraints in a universal way. □

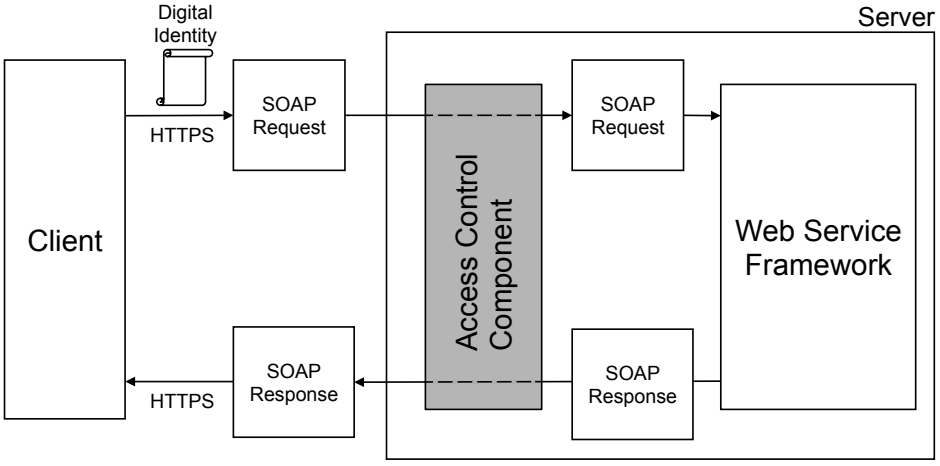

Figure 14: Information flow for a web service request with granted access

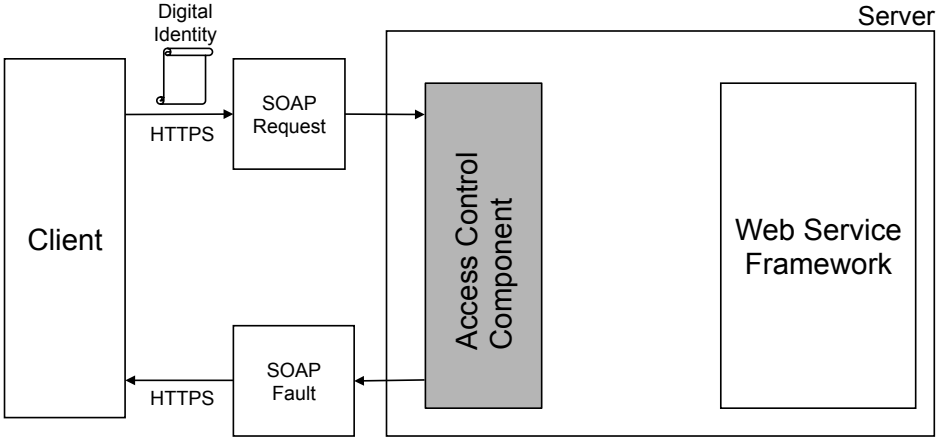

Figure 15: Information flow for a web service request with denied access

7.3.3 Policy Enforcement

The Policy Enforcement Point (PEP) is a component which intercepts a user's access request to a resource and enforces the PDP's decision. In REACTION the PEP is implemented in the form of an interceptor that is located on the server. Since REACTION is a service oriented architecture, access requests will be based on SOAP. Before a SOAP request is processed by the web service framework, the PEP intercepts the request. Based on the user's identity, i.e., the digital certificate used for HTTPS, it is decided whether the user is allowed to use the web service or not. This decision is based on the access control policy.

Figure 14 shows how the policy is enforced in the case of a user having the required permission. When encountering a SOAP request, a policy decision request is sent to the PDP by the access control component. If the user is allowed to use the web service, the request is forwarded to the web service framework for further processing. Subsequently, the web service's response is sent back to the requesting client. Here, the response passes through the interceptor without interference.

If the user does not have the permission to use the requested web service, the interceptor does not forward the SOAP request to the web service framework. Instead, the interceptor answers the user's request with a SOAP fault, informing the client about the denied access, see Figure 15.

7.3.4 REACTION Access Control Language and Policy

In REACTION, the permission assignment is done in a policy. In addition, role hierarchies can be defined in the policy as well. Based on the evaluation in section 7.2.3, OPL was chosen as the access control language used to describe the policy. In order to better serve the needs in REACTION, some changes had been made to the naming conventions of the XML syntax proposed in [3]. The XML-based language used for REACTION is illustrated in the example policy shown in Figure 16. The language is subject to the following structure which can be grouped into four major blocks:

Role definition. Here, all available roles have to be defined with a unique identifier. The roles are defined within a `roles` element. Therein, single roles are defined as a sequence of `role` elements having a unique identifier specified by the `id` attribute. The number of roles is unlimited but at least one must exist. An example role identifier is the string *Specialist*. □

Role hierarchy. In this block role relationships can be defined. A role can inherit from multiple roles. A role inheriting from another role, inherits all the assigned permissions. In the example, the *Specialist* role inherits from both the *Assistant Physician* role as well as the *Qualified Nurse* role.

The optional definition is done within a `role_hierarchy` element. The specific role relations are defined as a sequence of `role_inheritance` elements. Each of these elements has a child element `role` which identifies the target role, i.e., the role that inherits from other roles as well as a sequence of `inherits_from` child elements that refer to the identifiers of the roles, the target role inherits from. □

Permission definition. The permissions are defined with a unique identifier, a web service URI, and a web service operation. Its definition is done within a `permissions` element. Therein, each permission is defined by a `permission` child-element that consists of a unique identifier, the web service URI specified by the `webservice` attribute, and the web service operation specified by the `operation` attribute.

For the web service operation, it is allowed to assign a wildcard '`*`' instead of a string identifying the service's operation. This wildcard signifies that the permission covers all operations offered by the web service considered. The wildcard can be used to reduce administrative efforts. For roles that are allowed to access all operations of a specific web service, it is not necessary to define and assign all the required permissions to that role when using the wildcard. □

Permission assignment. For the permission assignment, each permission assignment consists of a unique identifier, a reference to a role, and a reference to a permission.

The definition is done within the `permission_assignment` element, the permissions can be assigned to the roles. The child-elements are a sequence of `permission_assignment` elements that each have a unique identifier specified by the `id` attribute, a reference to a role specified by the `role_ref` attribute, and a reference to a permission specified by the `permission_ref` attribute.

In the example, the permission assignment *pa1* means that all owners of the role *Specialist* have assigned the permission *p3* which implies that they are authorised to access the *registerPatients* operation of the *Management* web service.

The reference to the role, to which the permission is assigned, can also be empty. In this case, we speak of the *empty role*. If a permission is assigned to the empty role, the permission can be accessed by all users since all users at least own the empty role, see Section 7.3.1. □

```
<?xml version="1.0"?>
<policy>
    <roles>
        <role id=”Specialist"/>
        <role id=”Assistant Physician"/>
        <role id=”Qualified Nurse"/>
        <role id=”Nursing Assistant"/>
        <role id=”Nursing Trainee"/>
        ...
    </roles>
    <role_hierarchy>
        <role_inheritance>
            <role>Specialist</role>
            <inherits_from>Assistant Physician</inherits_from>
             <inherits_from>Qualified Nurse</inherits_from>
        </role_inheritance>
        ...
    </role_hierarchy>
    <permissions>
        <permission id="p1"
          webservice="/webservice/Management"
          operation="changeDosis"/>
        <permission id="p2"
            webservice="/webservice/Management"
            operation=”*"/>
        <permission id="p3"
            webservice="/webservice/Management"
            operation="registerPatient"/>
    </permissions>
    <permission_assignments>
        <permission_assignment id="pa1”
            role_ref=”Specialist” permission_ref="p3"/>
        <permission_assignment id="pa2"
            role_ref=”Qualified Nurse" permission_ref="p1"/>
        <permission_assignment id="pa3”
            role_ref=”Nursing Trainee" permission_ref="p2"/>
    <permission_assignments>
</policy>
```

Roles definition

Role hierarchy

Permissions definition

Role/Permission assignments

Figure 16: An example policy

8 Conclusion

The implementations of identification and authentication mechanisms for the two main scenarios of REACTION are very different from each other. However, the basic principles used for allowing or denying access to sensitive data are almost the same. What makes the two scenarios so different are the environments in which they are operated. In the in-hospital scenario we have a tightly controlled, professionally managed environment while in the primary care scenario only the GP's network is professionally managed and the patients' home networks can be expected to be neither controlled nor professionally managed. But still, access to the GP's network must be given to patients for accessing their own data.

Therefore, the mechanisms used for identification and authentication of users, as well as access control procedures had to take into account these different environments and eventually resulted in different architectures. Both architectures had to balance usability and security. However, the outcome was different for each scenario as the restrictions and the starting positions of the scenarios were also different. The patient portal from the primary care scenario was meant to be accessible from multiple locations which ruled out custom-made software that needs to be installed first. Hence, a web application was chosen for the patient portal along with a user name and password scheme as browsers offer no other way of user authentication without installing additional software. Thus, read access to patients' data from the patient portal was deemed acceptable but write access to any sensitive data seemed too risky, given that a public PC in an Internet café or a PC infected with a trojan horse might have been used to access the patient portal.

The in-hospital application in turn, was implemented from scratch as a stand-alone application. Therefore, integrating sophisticated security mechanisms into the design of the in-hospital application was conceptually easier and opened up the possibility for employing stronger mechanisms than user name and password schemes. The design allowed the use of public key authentication schemes while providing an easily accessible interface to users. For instance, users will only see a user name and password dialogue but are not aware that these are only used to open up containers holding the key pairs used for the actually employed challenge-response authentication with digital certificates. However, using strong authentication alone would have been too broad as an approach as not all authenticated identities should be granted the same privileges. Hence, role-based access control was introduced as an additional security mechanism. Roles allow to define groups of fine grained permissions which can be assigned to any identity. Using this mechanism, roles can be used to support the least-privilege principle which constitutes that any user should not have more privileges than needed to carry out her duty. In the in-hospital scenario, the efforts required to gain unauthorised access to patient data is expected to be considerably higher than in the primary care scenario. Firstly, an unauthorised entity must get hold of a user profile that has the required access rights. This however, is only possible if the entity is able to physically get hold of a mobile device from the hospital, i.e., she cannot simply remotely attack the system as she can with the patient portal. Given that the attacker is able to acquire a mobile device from the hospital, she secondly needs to discover the password used to protect a user profile before the theft of the device is noticed. Otherwise, the profile would be reported as compromised and be blacklisted such that any identity from the profile would be locked out of the system. Since the thief cannot use precomputed dictionaries to find the profile's password, it is expected that the time required to find the password is longer than the time the theft would go unnoticed.

For both REACTION scenarios, a balance had to be found between usability and security of the identity mechanisms employed. While the security aspect can be validated 'on paper', the usability aspect can only be validated by actual users of the system. This will be a task that is subject to the upcoming field trials for the two scenarios. Any feedback from the trials will then be evaluated and taken into account for updated versions of the security components.

References

[1] Anne Adams and Martina Angela Sasse. Users Are Not the Enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[2] Christopher Alm, Michael Drouineaud, Ute Faltin, Karsten Sohr, and Ruben Wolf. A Classification Framework Designed for Advanced Role-based Access Control Models and Mechanisms. Technical Report 851, TZI der Universität Bremen, 2009.

[3] Christopher Alm and Ruben Wolf. The Definition of the OPL Access Control Policy Language. Number MIP-0902. 2009.

[4] Christopher Alm, Ruben Wolf, and Joachim Posegga. The OPL Access Control Policy Language. In *Proceedings of the 6th International Conference on Trust, Privacy and Security in Digital Business*, TrustBus '09, pages 138–148, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] Anne Anderson. *Core and hierarchical role based access control (RBAC) profile of XACML v2.0*. OASIS Standard, 2005.

[6] ANSI. American National Standard for Technology. Role Based Access Control. Technical Report, INCITS 359-2004, February 2004.

[7] Stefan Asanin, Tobias Brodén, Peter Rosengren, Carlos Cavero Barca, and Stelios Louloudakis. D4.4.2 2nd Prototypes of core data, context and event handling management subsystems. REACTION, Deliverable 4.4.2, February 2012.

[8] Stefan Asanin, Akos Levay, Tamás Tóth, Irina Issaeva, Oleg Anshakov, Vasilis Kontogiannis, Malcom Clarke, Matthias Enzmann, Stephan Spat, and Eugenio Mantovani. State of the Art: Concepts and Technology for a unified data fusion architecture. REACTION, Deliverable 4.1, March 2012.

[9] Stefan Asanin, Peter Rosengren, Mattsand Ahlsén, and Peeter Kool. 1st Prototypes of core data, context and event handling management subsystems. REACTION, Deliverable 4.4.1, March 2011.

[10] Atta Badii, Daniel Thiemert, Renjith Nair, Adedayo Adetoye, Stephan Engberg, Morten Harning, Julian Schütte, and Manuel Mattheß. Draft of Virtualisation Design Specification. Hydra, Deliverable 7.2, July 2007.

[11] Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. XML Signature Syntax and Processing (Second Edition). W3C Recommendation 10 June 2008, June 2008.

[12] David F.C. Brewer and Michael J. Nash. The Chinese Wall Security Policy. *IEEE Symposium on Security and Privacy*, 1989.

[13] Paul De Hert, Paul Quinn, and Eugenio Mantovani. Ethical issues. REACTION, Internal Deliverable 9.1, February 2011.

[14] Claudia Eckert. *IT-Sicherheit – Konzepte, Verfahren, Protokolle*. Oldenbourg Verlag, 5th edition, 2007. ISBN 978-3-486-58270-3.

[15] The Economist. Leaked out — Password theft. http://www.economist.com/blogs/babbage/2012/06/password-theft, last accessed 19/07/2012, June 2012.

[16] Matthias Enzmann, Frederik Franke, Thomas Kunz, Eugenio Mantovani, and Paul Quinn. Concepts of Trust and Architectural Implications in Healthcare Environments. REACTION, Deliverable 7.2, February 2011.

[17] David Ferraiolo and Richard Kuhn. Role-Based Access Control. In *15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[18] Shirley Gaw and Edward D. Felten. Password Management Strategies for Online Accounts. In *Symposium On Usable Privacy and Security (SOUPS)*, July 2006.

[19] Hashcat. The table_lookup_attack in hashcat. http://hashcat.net/wiki/table_lookup_attack, last accessed 16/07/2012.

[20] Takeshi Imamura, Blair Dillaway, and Ed Simon. XML Encryption Syntax and Processing. W3C Recommendation 10 December 2002, December 2002.

[21] Burt Kaliski. PKCS #5: Password-Based Cryptography Specification, Version 2.0. RFC 2898, September 2000.

[22] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, and Sumit Shah. First experiences using XACML for access control in distributed systems. In *Proceedings of the 2003 ACM workshop on XML security*, XMLSEC '03, pages 25–37, New York, NY, USA, 2003. ACM.

[23] Tim Moses. *eXtensible Access Control Markup Language (XACML) Version 2.0.* OASIS Standard, 2005.

[24] MSG, MUG, FORTH-ICS, IN-JET, CNET, and FHG-SIT. D2.6 Appendix. REACTION, Deliverable 2.6 Appendix, February 2012.

[25] U.S. National Institute of Standards and Technology NIST. Advanced Encryption Standard (AES). FIPS PUB 197, November 2001.

[26] Institute of Electrical and Electronics Engineers (IEEE). ISO/IEEE Health informatics – Personal health device communication – Part 10417: Device specialization Glucose meter. IEEE Std 11073–10417–2010, May 2010.

[27] Oxid.it. Cain & Abel password cracker. http://www.oxid.it/cain.html, last accessed 16/07/2012.

[28] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, February 1996.

[29] Bruce Schneier. Choosing secure passwords. http://www.schneier.com/blog/archives/2007/01/choosing_secure.html, last accessed 16/07/2012, January 2007.

[30] Julian Schütte, Tobias Wahl, Adedayo Adetoye, and Sebastian Zickau. Implementation of security-related modules and managers. Hydra, Deliverable 7.4, July 2008.

[31] Bluetooth SIG. Core Specification v2.1 + EDR, July 2007.

[32] Bluetooth SIG. Health Device Profile (HDP), June 2008. Version 1.0.

[33] Bluetooth SIG. Core Specification v3.0 + HS, April 2009.

[34] Mark Strembeck and Gustaf Neumann. An integrated approach to engineer and enforce context constraints in RBAC environments. *ACM Trans. Inf. Syst. Secur.*, 7(3):392–427, 2004.

[35] Jesper Thestrup, Helene Udsen, Trine Sørensen, and Louise B. Riley. Scenarios for usage of the REACTION platform. REACTION, Deliverable 2.1, June 2010.

[36] T. Toth, J. Fursse, K. Khosraviani, M. Enzmann, F. Chiarugi, V. Kontogiannis, H. Udsen, S. Spat, B. Höll, K. Neubauer, L. Schaupp, and M. Clarke. Prototype Application Specification. REACTION, Deliverable 2.6, February 2012.

[37] Ruben Wolf, Frederik Franke, and Schneider Markus. iMobileSitter: Sicheres Passwortmanagement in Zeiten von digitalen Schlüsseldiensten und Advanced Persistent Threats. *<kes> Special Mobile Security*, pages 31–33, June 2012.

[38] ZigBee Alliance. ZigBee Specification. ZigBee Standards Organization, Document 053474r17, October 2007.

[39] ZigBee Alliance. ZigBee Health Care Profile Specification, Version 1.0, Revision 15. ZigBee Standards Organization, Document 075360r15, March 2010.