NTNU – Trondheim
Norwegian University of
Science and Technology

Identity Management in M2M Networks

Pranas Butkus

Master in Security and Mobile Computing

Submission date:	June 2014
Supervisor:	Van Thanh Do, ITEM
Co-supervisor:	Jukka K. Nurminen, Aalto University School of Science

Norwegian University of Science and Technology
Department of Telematics

Title: Identity Management in M2M Networks
Student: Pranas Butkus

Problem description:

Currently developed future communication technologies are typically based on multiple devices that communicate with each other to exchange information about an environment or objects that are monitored and afterwards automatically respond with necessary actions. Although several of these future communication technologies, including Machine-to-Machine (M2M) communications, have already caught attention of standardisation authorities, the standards that are currently available focus on static, industry- and business-oriented solutions involving isolated environments with fixed numbers and types of communicating devices.

In the mentioned static model, automated network functioning with minimal maintenance needs can be achieved by pre-configuring devices with identity information before deployment. However, future systems will largely focus on human users and applications that automatically assist people with their tasks and feature dynamic relationships. For example, a user utilising multiple applications might use his/her devices to access separate systems, which recognise the user with different identities. Furthermore, the application set utilised by a user can change over time. Therefore, future communication technologies supporting user-oriented applications will need an Identity Management (IdM) system capable of managing identities of diverse communicating parties and changes in their relationships.

This master's thesis focuses on an IdM system for future communications that addresses challenges related to user-oriented applications, as described above. The project is divided into several steps. First, it involves analysis of related communication and IdM systems with a particular focus on M2M technology and its specifications. Second, it includes identification and elaboration of requirements for the dynamic user-oriented IdM system. Finally, it proposes an IdM system that meets the identified requirements.

Responsible professor: Do van Thanh, ITEM

Abstract

Evolving communication technologies stimulate a rapid growth in utilisation of communication-capable devices and therefore amount of transmitted data. This imposes new requirements for automatic device and data management necessary for successful exploitation of new opportunities. Unfortunately, currently developed systems, including Internet of Things and Machine-to-Machine communications, mainly focus on industrial applications that involve fixed users, proprietary environments as well as ad-hoc devices and things, whereas regular users along with possibilities and challenges created by growing sets of personal user equipment remain ignored.

This thesis addresses the defined problem by analysing currently developed and utilised communication technologies and identity management systems as well as proposing an advanced identity management system that considers user-related needs and enables user-aware automatic device-to-device communications. Our system is unique compared to other automatic communication systems in that it enables global communication of devices owned or used by different parties and supports dynamic connection and relationship establishment based on data administered in a sophisticated identity management infrastructure. Unlike existing identity management mechanisms, our system extends the notion of an identified and authenticated entity to a combination of both user and device. Furthermore, the system introduces an original Single Device Sign-On feature that simplifies user login procedure when accessing a service with multiple devices. As a consequence, this thesis suggests a new direction for evolution of communication technologies as well as user-targeted Internet-based services and applications.

Contents

List of Figures

List of Acronyms

3GPP	3rd Generation Partnership Project.
3GPP2	3rd Generation Partnership Project 2.
AAA	Authentication, Authorisation and Accounting.
ABAC	Attribute-Based Access Control.
AC	Assertion Consumer.
ACL	Access Control List.
ACM	Access Control Matrix.
AcN	Access Network.
AmI	Ambient Intelligence.
ANP	Access Network Provider.
API	Application Programming Interface.
App-ID	Application Identifier.
ARPANET	Advanced Research Projects Agency Network.
AS	Authorisation Server.
BT	Bluetooth.
CN	M2M Core Network.
CONN-ID	Service Connection Identifier.
D'D	D' Device.
DA	Device Application.
DD	D Device.
dD	d Device.
DIP	Device Interworking Proxy.

DNS	Domain Name System.
DS	Device Subsystem.
DSCL	Device Service Capabilities Layer.
EAP	Extensible Authentication Protocol.
EPC	Electronic Product Code.
ETSI	European Telecommunications Standards Institute.
FQDN	Fully Qualified Domain Name.
FTP	File Transfer Protocol.
GA	Gateway Application.
GIP	Gateway Interworking Proxy.
GSCL	Gateway Service Capabilities Layer.
HTML	Hypertext Markup Language.
HTTP	Hypertext Transfer Protocol.
HTTPS	Hypertext Transfer Protocol Secure.
IBAKE	Identity-Based Authenticated Key Exchange.
IBC	Identity-Based Cryptography.
ICT	Information and Communication Technology.
ID-FF	Identity Federation Framework.
IdM	Identity Management.
IdP	Identity Provider.
IdPS	Identity Provider Subsystem.
IMSI	International Mobile Subscriber Identity.
IoT	Internet of Things.
IP	Internet Protocol.
IR	Infra-Red.
JSON	JavaScript Object Notation.
Kma	M2M Application Key.

Kmc	M2M Connection Key.
Kmr	M2M Root Key.
M-Bus	Meter-Bus.
M2M	Machine-to-Machine.
MAC	Media Access Control.
MAS	M2M Authentication Server.
MAS-ID	MAS Identifier.
MSBF	M2M Service Bootstrap Function.
MSBF-ID	MSBF Identifier.
MSCF	M2M Service Connection Function.
MTC	Machine-Type Communications.
MTC-IWF	MTC Interworking Function.
NA	Network Application.
NFC	Near-Field Communication.
NIC	Network Interface Card.
Node-ID	Node Identifier.
NSCL	Network Service Capabilities Layer.
OID	Object Identifier.
OIdP	OpenID Identity Provider.
ONS	Object Naming Service.
OS	Operating System.
PC	personal computer.
PIN	Personal Identification Number.
PLC	Programmable Logic Controller.
PLMN	Public Land Mobile Network.
PoC	M2M Point of Contact.
PROV-ID	Service Provider Identifier.
PSK	Pre-Shared Key.
RADIUS	Remote Authentication Dial In User Service.
RBAC	Role-Based Access Control.

RelP	Relying Party.
REST	Representational State Transfer.
RFID	Radio Frequency Identification.
RO	Resource Owner.
RP	Reference Point.
RS	Resource Server.
SAML	Security Assertion Markup Language.
SCADA	Supervisory Control and Data Acquisition.
SCL	Service Capabilities Layer.
SCL-ID	SCL Identifier.
SCS	Services Capability Server.
SDSO	Single Device Sign-On.
SMS	Short Message Service.
SP	Service Provider.
SS	Service Subsystem.
SSO	Single Sign-On.
TEE	Trusted Execution Environment.
TISPAN	Telecommunications and Internet Converged Services and Protocols for Advanced Networking.
TLS	Transport Layer Security.
UA	User Agent.
UE	User Equipment.
UICC	Universal Integrated Circuit Card.
UML	Universal Modelling Language.
UMTS	Universal Mobile Telecommunications System.
UPID	User-Provided Identifier.
URI	Universal Resource Identifier.
URL	Universal Resource Locator.
W-LAN	Wireless Local Area Network.

WiMAX	Worldwide Interoperability for Microwave Access.
WPAN	Wireless Private Area Network.
xDSL	x Digital Subscriber Line.
XML	Extensible Markup Language.
XRI	Extensible Resource Identifier.

Chapter 1 Introduction

Less than half a century ago, in 1969, one of the first packet-switched networks called Advanced Research Projects Agency Network (ARPANET) was established with the goal of providing universities and research laboratories with a long-distance data exchange mechanism. Besides accomplishing this goal, ARPANET also initiated development of various networking protocols that have afterwards resulted in creation of the Internet utilised not only by academia, but also businesses, governments and everyday users, thus improving lives in many aspects.

Although the early version of the Internet enabled global communication, it provided a regular user with a rather limited functionality, i.e. remote communications via email and access to static web sites in the World Wide Web. The key factor that made the Internet as powerful as we know it today was creation of digital user identities and Identity Management (IdM) mechanisms that enabled treating on-line and physical user presence alike. By providing user-friendly features for identity creation and access to data, e.g. by using federated IdM systems and Single Sign-On (SSO) functionality, origination of IdM systems was the key factor that stimulated emergence of web-based services, social networks and other automated user-related functionality.

Currently, as the number of various network-connected devices rapidly increases, the minds behind evolution of communication networks and Information and Communication Technology (ICT) are putting effort into creation of autonomous device communications and utilisation of these devices to gather sufficient data to enable even more capable, automatic and intelligent applications. The most prominent of the new technologies addressing these aspects are Machine-to-Machine (M2M) and Internet of Things (IoT). M2M systems introduce the concept of autonomously functioning networked devices with no or only minimal intervention of human personnel. This technology mainly focuses on devices themselves and capabilities required for their autonomous functioning. A related technology, IoT addresses the amount of data these connected devices generate and thus emphasises the need for an infrastructure and applications, i.e. an advanced version of the Internet, capable of handling this data.

Although the mentioned technologies seem promising, they principally aim at industrial applications based on isolated hierarchies of proprietary communicating devices. Furthermore, most of these devices are rather primitive and serve only a single pre-defined purpose, i.e. sensors gather specific kind of data, actuators perform certain physical operations and Radio Frequency Identification (RFID) tags embed a *device* concept into ordinary things by merely enabling their identification needed for discovery of their meta-data. As a result, M2M and IoT enable systems with pre-defined parties that are involved in communication and management and linked together using static relationships.

Our vision of evolution in communication technologies differs from the previously mentioned approaches in the following aspects. First, the technology should serve a regular human user and involve his/her devices as well as custom user applications. Second, it should enable dynamic, on-demand relationships among communicating parties, e.g. user's devices and local service provider servers, rather than be limited to fixed and pre-configured links. Thus, in order to enable this functionality, currently developed technologies are insufficient and a different approach is needed.

This thesis shows that the described evolutionary step of device-to-device communications can be achieved by applying the means proven successful before, i.e. creating a suitable IdM infrastructure. We address the problem of user-oriented, dynamic and intelligent communications by presenting an IdM system that involves both user and device identities, thus providing sufficient data for automatic and dynamic user-aware interaction of devices. The proposed system addresses various user-related IdM challenges, including multiple identities, shared hardware equipment and continuously increasing number of devices carried by a user. By noticing the improvements in usability created by federated IdM systems and SSO service, we further enhance the IdM system by proposing a Single Device Sign-On (SDSO) mechanism that enables a user to connect to a service with multiple devices he/she carries by performing authentication procedure from one of the devices.

It is important to note that although our system was initially aimed at M2M communications, literature analysis that was performed as part of the thesis process revealed that current M2M concept is rather limited and essentially addresses the mentioned static systems. Conversely, we aim at enabling global dynamic device-to-device communications. Therefore, unless we specifically refer to M2M technology and specifications, we use the term M2M to describe autonomous device-to-device communications.

This thesis is structured as follows. Chapter 2 presents the background of identities, identity management mechanisms and their roles, as well as briefly discusses future communication technologies under development. Afterwards, Chapter 3 further covers the current situation of IdM and communication systems by presenting the most relevant systems implemented as industrial solutions or proposed in scientific literature. Subsequently, Chapter 4 explains the proposed system's idea, the problems it tackles and targeted use cases. Afterwards, Chapter 5 presents the actual solution. Chapter 6 evaluates the system from various perspectives and provides a discussion. Finally, Chapter 7 provides final remarks and conclusion of the thesis.

Chapter 2 Background

This chapter presents the background information related to IdM and future communication technologies. This information is used in further chapters of this thesis. Section 2.1 analyses the importance of identities, IdM and IdM systems. Afterwards, Section 2.2 provides an overview of currently developed future communication technologies, specifically M2M, IoT and Ambient Intelligence (AmI). These technologies are believed to provide the basis for ubiquitous computing and advanced automatic applications. Chapter 3 continues the description of the technologies presented in this chapter by discussing implementations of these technologies and their standardisation processes.

2.1 Identity Management

2.1.1 Utilisation of Identifiers

Information and communication systems largely utilise various identifiers necessary for recognition of a communicating party, as well as consistent session management. For example, one of the popular computer networking models, the TCP/IP model, contains 4 layers responsible for distinct functionality related to communications, as explained in Chapter 1 of [Hun02]. Each of the layers introduces a certain identifier, e.g. a Media Access Control (MAC), Internet Protocol (IP) and email addresses for identification of a Network Interface Card (NIC), network-host combination and a human user, respectively. These identifiers are necessary for identification and data transmission over a communication channel.

Although sufficient for low-level communication or user-controlled applications, identifiers often do not carry enough data to enable higher-level automatic applications. Thus, additional information related to communicating entities or peers is needed for connection establishment and further communication in higher-level application context.

2.1.2 Identities

Data that describes relevant features of a subject, i.e. a person or a thing, and stored in digital space is called *digital identity*, as defined by Windley [Win05]. According to the author, digital identity features are comprised of attributes, describing subject's characteristics, preferences and traits, which are similar to attributes but are more inherent to the subject and experience very rare or no modifications. In this document, we simplify the definition of a digital identity and use *identity* to refer to a set of attributes characterizing a communicating entity.

It is clear that data used to describe human users and devices differ rather significantly, in such aspects as data format, number of data sets and their relation to each other, data sensitivity, as well as utilisation and management of data, which may be regulated by law. Furthermore, behaviour of users and devices differs rather significantly. For example, a user typically controls a device, and the latter simply performs requested operations, i.e. it is dependent on the user. Therefore, scientific literature typically uses term *identity* for user entities, i.e. researchers analyse *user* identities and treat device-related data merely as device meta-data [CY11, RC11]. On the other hand, recent scientific works, such as Hydra project [AH08], are starting to address the increasing intelligence and actuality of devices, thus they treat device-related data as device identities. To address the mentioned similarities between user and device entities from the perspective of communication, we address both user- and device-related data sets as identities. However, due to inherent differences, we make a distinction between device identities and user identities, and further analyse these identities separately.

Identities enable entity identification, i.e. recognition of a party by detecting its unique characteristics. This process enables personalisation/customisation of a service and communication, as well as access to and manipulation of personal or other entity-related data. In this respect, attributes are used to represent identifiers, which are partitioned into strong and weak identifiers [BSSB05]. A strong identifier is sufficient to uniquely identify the entity in the system, and its value often does not have other purposes. On the other hand, a weak identifier provides important information about an entity, but its value may be shared with multiple other entities in the same system. Thus, the strength of an attribute is determined by the uniqueness of its value and the overall number of entities in the system. Furthermore, a set of weak identifiers may enable reliable identification of a unique entity. Besides attributes, other information units, such as certificates, can be used to perform identification.

All identity data is created, managed and protected by a certain management authority, generally called an IdM system. The remainder of this section thoroughly analyses IdM system actions and models.

2.1.3 Entity Authentication

Merely providing identity information, e.g. an identifier, without a proof that this information really belongs to the claimed entity, is often insufficient. Thus, multiple systems require communicating parties to perform entity authentication, which is one of the primary functions of IdM systems [Win05]. Authentication is a procedure performed to verify that an entity is what it claims to be. This operation is critical in many systems that involve confidentiality, integrity and other security-related mechanisms. Authentication is typically performed in the beginning of communication. However, security-critical systems often require repeated authentication either periodically or before performing important operations.

Scientific literature and the industry utilise multiple entity authentication methods, which depend on specific systems and entity types. As mentioned above, we analyse authentication procedure of two separate types: user and device identities. During user identity authentication, a human user is typically asked for the following [BL07]:

1. Something a user knows. In this case, a user possesses knowledge of a secret which can be used as a proof of identity. The most commonly used secret type is a password that a user provides together with his/her unique user name, which identifies the entity. There are other ways created to identify users utilising their knowledge, including recognition of graphical images, drawing patterns on screen [SS12] or even on the back side of mobile devices [DLvZN$^+$13]. However, such inventive authentication methods are incapable of replacing passwords in a widespread use, because, in comparison to passwords, they provide little advantage in security and usability and are inferior to passwords in authentication system's deployability [BHOS12].

2. Something a user has. In this case, the system checks user's possession of certain items, i.e. tokens, data containers or other secrets that a user does not keep in his/her memory. Everyday devices, such as a smart phone or a Bluetooth (BT) token [CN03], may also be utilised as tokens if the secret information is stored and kept in a secure manner. Compared to the previous case, possession of an item frees the user from having to learn and remember secret phrases. Furthermore, it enables more frequent authentication to repeatedly verify user's presence, which would be too intrusive if passwords were used for authentication. However, it is generally doubted whether a token really proves the identity of the user, since these items can be easily shared among multiple users, lost or even stolen [CN03]. Thus, this type of authentication is more suitable for authentication of actual devices rather than human users that carry them.

3. Something that is characteristic to the user. In this case, the user is reliably identified by determining his/her unique features, i.e. genetic traits, e.g.

fingerprints, voice pattern, eye retina or DNA, and comparing this data to equivalent data obtained during user's registration [SP12]. However, this method raises concern about potential unintended use and theft of digitised biometric information, since, unlike passwords or physical tokens, biometric information is bound to a user, it does not change and thus cannot be revoked [BL07]. Furthermore, it is possible to make copies of this information, e.g. photographs or gelatin fingerprint patterns, which may then be used to fake body parts involved in authentication [BL07, Pan10]. It is possible to prevent this kind of attack by supervising devices and premises involved in authentication, but it is not technically feasible for all systems and services.

This user-presented information obtained directly from the user or by performing certain digital operations, e.g. scanning a fingerprint image and afterwards calculating a pattern, is called credentials. After they are obtained from the user, credentials are sent to a certain security authority or policy enforcement point for verification.

An alternative method to identity a user is by analysing his/her behavior, e.g. gait, file system browsing or other behavioural patterns, as specified by De Luca et al. [DLvZN+13]. However, as the authors specify, these techniques are susceptible to attacks, because the behaviour can be recorded and replayed, thus they are rarely used in secure systems.

Device authentication is similar to user authentication, although less complicated. We classify it into two categories:

1. Something a device has. This case involves secrets stored in the device that are used to prove its identity and is generally used to enable user authentication with a token described above. As in the case of human users, the secrets may be transferred and checked in a certain authentication authority. Alternatively, a device may use a secret to derive another piece of information that is accepted as a reliable proof by the authenticator. Challenge-response and certificate-based authentication mechanisms are based on this concept. It is important to emphasise that although device authentication items, e.g. a X.509 certificate, are effectively issued to users for user authentication, they are often used in an automatic matter and do not ensure user's presence at exact moment, even though a login or periodic presence verification procedure is used [CN03]. Thus, secrets stored in a device should be treated as a means to authenticate devices and not users.

2. Something that is characteristic to a device. This option enables determining device identity by its behavioural or physical context credentials, e.g. geographic or low-level operational features, such as frequency of transmitted signals

[ARE+05]. However, although suitable for authentication, these credentials are considered context-based rather than identity-based, thus they are not suitable for detection of a specific identity, not to mention its verification.

2.1.4 Entity Authorisation and Accounting

Although authentication enables reliably detecting identities of communicating parties, it is often insufficient for access to a certain resource or communication in a system. Typically, a system contains resources or functionality that is accessible to specific and limited sets of users, e.g. private data of a specific user. To enable control over the mentioned features, the system has to check if the authenticated entity is allowed to access the requested data and functionality. Thus, authentication is followed by authorisation, i.e. a process that involves analysing entity's rights in the system [Win05]. This procedure is performed in a policy decision point and typically compares security policy for a requested resource with permissions and entitlements of the authenticated entity that requests it. As specified by Rotondi et al. [RP12], there are several access control mechanisms, including the following:

1. **Access Control Lists (ACLs)** enable explicitly specifying individual actions that an entity, which is called a subject, is allowed to perform with specific resources, which are called objects. A generalised, more systematic approach of this technique is Access Control Matrix (ACM), which enables specifying access rights in a matrix, where matrix elements are subject-object pairs. However, both ACLs and ACM approaches are complicated to manage when the number of subjects or objects in a system is large.

2. **Role-Based Access Control (RBAC)** helps solving the rights management problem mentioned above by introducing an additional *role* layer. This technique allows associating access rights to roles rather specific subjects. Thus, a role can be treated as a subset of actions. Afterwards, one or more roles are assigned to each subject, thus allowing the latter to perform operations defined by one or more access right subsets.

3. **Discretionary Access Control and Mandatory Access Control** focus on the issuer of access rights. In Discretionary Access Control, the resource owner, typically a human user, specifies who or what can access his/her resources. Conversely, Mandatory Access Control involves a central administration that determines the rights of subjects to access system's objects.

4. **Attribute-Based Access Control (ABAC)**, unlike previously described models, grants access to resources based on attributes of identities rather than identities themselves. However, this method does not enable detecting specific identities and is thus not suitable for the systems we analyse in this thesis.

To further increase security of the system, operations of an entity are often recorded. This procedure is called accounting, and is more often used for logging human user operations in order to gather proof for investigation of security breaches and prevent non-repudiation.

Servers responsible for authentication, authorisation and accounting are typically called Authentication, Authorisation and Accounting (AAA) servers.

2.1.5 IdM System Models

As mentioned above, identities are issued and managed by IdM systems. Such a system also performs authentication and authorisation of identities in a specified scope. However, this does not mean that an IdM system serves at most one Service Provider (SP). Based on agreements and shared standards for identity data, as well as its exchange mechanisms, a single IdM system may be utilised by multiple SPs. This is especially common in user-oriented domains. User IdM system models, i.e. those operating on user identities, are traditionally classified into three categories: isolated, federated and centralised [GJK$^+$09]. Besides the differences in IdM models, types of identities also differ depending on specific IdM models, which are explained together with the models below.

1. **Isolated IdM model** requires that each service contains its own IdM system. In this case, the IdM system maintains separate identities along with isolated identifiers for each user, thus every user has a separate identity per every service he/she uses.To utilise such a system, a user creates a *virtual identity*, representing only a subset of the entire user's identity (all the data known about the user) necessary for a user's actions in a particular domain of a service [HZW11].

   Although this model is used in most of the systems, it has a disadvantage of not being scalable from perspectives of both the user and SP. As the number of a system's users increases, IdM at a SP's side requires more resources, including hardware, bandwidth and human personnel. From the customer's perspective, a person that utilises services provided by multiple providers has to manage credentials for all the accounts. This imposes additional effort needed for creation of an account and accessing it later in order to use the service. Furthermore, since most of the systems authenticate users by requesting passwords (although alternatives exist, as specified above), this creates issues related to password reuse, data leaks and limited human memory capabilities [BHOS12].

2. **A federated system model** enables sharing identity information among multiple parties that trust each other and have standardised procedures to

map identity fields. These communicating parties, forming a *federated group*, accept the same identifier provided by a user. In practical implementations, a number of domains acting as Identity Providers (IdPs) control identity data and authenticate users. The same or other parties in the federated group, e.g. SPs, request trusted IdPs to authenticate a user and provide user's identity information. In this case, a user creates an identity by providing personal details that are afterwards stored in one or multiple IdPs. Afterwards, when a user wishes to access a service, a certain trusted IdP authenticates the user and provides the service with a subset of stored identity attributes defined with relationship agreements.

Federated IdM system model reduces the effort needed for a user to start using a service and manage his/her credentials. Furthermore, since authentication in this model is performed by an IdP, the model enables SSO service. SSO simplifies utilisation of multiple services by requiring one-time user authentication, after which a user is provided seamless access to multiple potentially unrelated services that use the same IdP. In Section 3.2, we present several currently utilised IdM systems that provide SSO feature and are based on this model.

3. **A centralised IdM system model** strictly defines the roles and relationships among systems: there is one IdP that manages user authentication and identity information, and one or more SPs that use the identity information. The IdP has to be trusted by both the customers and the service providers. This model also enables SSO feature and reduces user's effort to use a new service, if there are other services in the same domain that are already used. In this model, a user is allowed to create one or more virtual identities, introduced together with isolated IdM model. However, virtual identities issued by an IdP are not linked to each other, although they are typically based on the same identity in the IdP [HZW11].

Besides user authentication, current services and systems impose new requirements for IdM systems. For example, some systems enable access rights delegation, which enables third parties to perform actions on the authenticated user's behalf. Furthermore, increasingly popular smart phones stimulate the trend of mobile applications that deal with challenges imposed by mobility, e.g. user or network mobility and multi-homing. Thus, they create a need for new modern solutions in identification management systems. Several modern systems addressing these issues are analysed in Section 3.2.

2.2 Communication Technologies

In this section, we specify the other aspect of the thesis' topic, that is modern communication technologies. We do not provide specific details about implementations, but rather define the general ideas of these technologies.

2.2.1 M2M Communications

M2M communications is a broad term describing any technology that enables networked devices to exchange information and perform actions without manual assistance of human personnel [WWR+12]. Besides performing tasks specific to their system's functionality, these devices act as independent network nodes, capable of initiating communication by sending messages to other devices as well as responding to incoming requests.

An M2M system potentially comprises a wide variety of both simple and intelligent devices, such as sensors, actuators and controllers. These devices can be connected together by utilising various wired and wireless network technologies, e.g. Ethernet, Wi-Fi, ZigBee and Universal Mobile Telecommunications System (UMTS). M2M is expected to utilise currently available, open and standardised device and communication technologies to create intelligent applications and services that are scalable, reliable and, most importantly, cheap to deploy and maintain [Law04]. This technology is also related to IoT technology, a vision that refers to connecting all common devices and non-electronic items, i.e. things, to the Internet, as discussed below. A system with such a large number of connected nodes increases the effort of network management, making it extremely expensive and even impossible to perform it manually. Thus, automation provided by M2M becomes essential.

The scope of application provided by M2M technology is rather wide. Some of the areas expected to utilise M2M technology identified by Potter et al. [PHS13] include smart metering, automotive applications, residential, commercial and large-area automation as well as E-health. These applications operate in diverse environments and create different challenges. Thus, as the authors note, there is no single optimal solution covering all of these application areas.

Technologies similar to M2M have been present for a long time. The concept of M2M technology originates from telemetry, designed for automatic measurement and transmission of data using wired or wireless technologies [GBK+11]. The difference between telemetry and M2M is that the latter utilises more common, open, standardised and ubiquitously used technologies. For example, a predecessor of M2M, Supervisory Control and Data Acquisition (SCADA) system also utilises a similar network that includes sensors and actuators [Law04]. However, in SCADA, the controlled devices are connected in a wired-manner to a central server, which is responsible for polling the sensors and hence acquiring data. Furthermore, SCADA is based on proprietary technologies, making the system impractical due to limitations on system's scalability and flexibility as well as high costs, difficulties and large time resources needed for system's deployment and maintenance.

2.2.2 Internet of Things

IoT promotes the ideology that everything, from devices to ordinary things, should be connected to the Internet [KKK$^+$13]. As noted by Cisco [Eva11], the term also emphasises evolution of the Internet, since the current Internet is designed for human user-oriented communications and user-based traffic flows. Thus, new solutions are needed in order to support the continuously increasing growth of numbers of everyday things connected to the Internet, which has already overtaken the number of connected human users. Furthermore, as the author of the term IoT, Kevin Ashton [Ash09], describes, the infrastructure has to become capable of processing enormous load of device-generated data, thus IoT is related not only to communications, but also endpoints, i.e. smart servers and applications.

IoT is still in its early development phase and, as noted by Koreshoff et al. [KRL13], its definition is not stable yet. Furthermore, the precise position of IoT technology is unclear. Although presented as a powerful communication technology connecting devices and things in the global Internet structure, IoT seems to be targeted more towards simple devices and things, with applications enabling asset tracking and quick item meta-data retrieval using RFID tags, as well as environment monitoring using sensors [Ash09, Per12, WJ12]. In the terms of the latter use case, IoT closely relates to M2M. In fact, Song et al. [SKSS14] define IoT as an underlying technology providing connectivity for M2M devices and protocols.

2.2.3 Ambient Intelligence

Another technology currently in development that is believed to enable ubiquitous computing is AmI. AmI is a technology that enables various device-based functionality, when devices are integrated in physical environments and directly or through user's devices provide him/her with functionality [CNH$^+$08]. For example, in a home environment, a system detects user's presence, and the latter can control the surrounding devices using voice commands [GHL05]. Another example is a AmI-powered conference building, where the premises are equipped with screens, Near-Field Communication (NFC) readers and other devices enabling attendants to easily register to the conference and sessions using their smartphones, obtain real-time conference progress information and start their presentations by simply approaching the screen in front of a presentation room [CNH$^+$07].

Unlike the previously analysed technologies, AmI actually involves users and minds their identities when providing services. On the other hand, AmI is also related to the previously discussed technologies due to utilisation of the same devices, including RFID and NFC-based equipment [CNH$^+$08, CNH$^+$07].

However, envisioned AmI applications are limited to local environments, and the largest effort of the technology's research is based on detecting context. For example, current scientific effort includes works related to general detection of context using RFID and NFC [CNH$^+$08, CNH$^+$07], user's identity detection based on biometrics [Piu13], or interacting with user using voice commands and learning to understand user's behaviour and habits to improve application features [GHL05]. Thus, this technology is focused on local user-to-environment and user-to-device communication rather than device-to-device communication. Since our work focuses on device-to-device communication based on user identities, AmI is not directly related to our work.

Chapter 3

State of the Art

This chapter analyses the state-of-the-art IdM systems utilised in ICT as well as communication technologies related to ubiquitous device-to-device communications. Since the goal of this thesis is to propose an IdM system for autonomous user-oriented device communications, this chapter focuses on IdM related to communicating parties and communications over the Internet performed among entities.

As mentioned in Section 2.1.2, there are significant differences among identities of human users and devices. Thus, the analysis of current implementations of IdM systems presented in the document is divided into two parts. The first part presents systems that manage identities for devices and items, e.g. the *things* in IoT. These systems are typically utilised by commercial organisations for business purposes, e.g. asset tracking or equipment management. The second category includes user-oriented IdM systems that manage human user-related information. These systems are typically used by user-oriented services, such as web services and applications. The further

3.1 Device-Oriented IdM Systems

In this section, we analyse identities and IdM systems related to devices. As mentioned in Section 2.1.2, information related to devices is often treated simply as meta-data rather than actual identities, although the current increase in device intelligence and capabilities are reflected in new projects that treat devices as independent peers with identities.

Quite surprisingly, identification of devices and acquisition of their further information is largely based on RFID technology. It enables item identification and meta-data acquisition based on RFID tags that are attached to the mentioned items, e.g. baggage at an airport [RC11]. The applications are illustrated in the analysis of scientific work presented further.

3.1.1 Object Identifier for Meta-Data

Roussos and Chartier [RC11] describe utilisation of an Object Identifier (OID) mechanism that identifies items using an identifier that follows Universal Resource Identifier (URI) format and hence enables access to item meta-data stored on the web. OIDs follow a tree structure, thus enabling definition of complex relationships, including parents, siblings and children. The system is intended to provide information about items, i.e. relatively static data. Furthermore, OIDs may be reused for identification of different items, if temporary object identification is sufficient.

Due to utilisation of URIs, the system depends on Domain Name System (DNS). However, as the authors note, DNS services are managed locally by network administrators, and no users from outside the domain, e.g. a company, can create or manage OIDs for items. Thus, this solution is restricted to proprietary systems.

3.1.2 Cooltown

Another RFID-based device-related identity management system is Cooltown, specified by Kindberg et al. [KTV05, KBM$^+$02]. The system uses Infra-Red (IR) and RFID to transfer identifiers of users, devices and places, i.e. links to web-pages representing them, enabling so-called *web-presence* [KBM$^+$02]. The web-pages are either hosted in the items themselves, e.g. in connected devices, or external web-servers. These web-pages provide information and services related to the represented entities. Furthermore, since entities are web-present, the approach actually treats entity meta-data as identities.

In the broad sense, the system provides an AmI functionality, presented in Section 2.2.3, where a user may interact with locally available services presented by other entities in the environment using his/her device, e.g. a mobile phone. Similarly to OID-based approach, Cooltown is based on DNS services, thus creation of new identifiers is rather restricted.

3.1.3 Object Naming Service

Object Naming Service (ONS) built on top of DNS is yet another similar identification technology that is based on DNS functionality for storing *Name Authority Pointer* records containing information about the location of meta-data of objects referred to by identifiers [RC11]. ONS was originally designed in EPCGlobal system to map product class level information to a service point and description. Due to relation to DNS, the device meta-data may be obtained by sending regular DNS queries.

Although ONS enables finding the exact location of meta-data storage, it is merely a discovery service and thus storage and acquisition of the actual meta-data exceeds the scope of this technology. Therefore, in order to find and obtain this data, operation of ONS must be combined with additional protocols, which may complicate development and management of such systems. Furthermore, ONS does not specify the way for obtaining object identifiers. Since Electronic Product Code (EPC) is used to identify products, it potentially identifies the objects by using RFID tags and bar-codes. However, as indicated by Roussos and Chartier [RC11], ONS is now an insecure, deprecated technology that has been replaced by EPC Discovery Service.

3.2 User-Oriented IdM Systems

This section discusses application-level IdM systems for management of human user identities. We start by discussing features of OpenID in Section 3.2.1. Afterwards, Section 3.2.2 analyses Security Assertion Markup Language (SAML) protocol. Then, Section 3.2.3 describes OAuth v2.0. Finally, Section 3.2.4 discusses a recent protocol OpenID Connect.

3.2.1 OpenID

General Description

OpenID is an authentication protocol providing a user with a SSO feature for accessing web-based services [Opea]. It is a decentralised IdM system, i.e. it allows having multiple OpenID identity servers capable of providing identity information to any service supporting OpenID authentication.

OpenID, with the currently newest version 2.0, is designed as an IdM solution for access to websites and web applications. It involves three parties: a User Agent (UA), controlled by a human user, a Relying Party (RelP), which provides the service, and an OpenID Identity Provider (OIdP), which authenticates the user and delivers his/her personal information to RelP [Opea]. OIdP is identified by a URI. To use OpenID technology, a user has to register in a selected OIdP and obtain an identifier called User-Provided Identifier (UPID), which is needed later for authentication. A user may create several identity profiles at the same OIdP and afterwards select a specific identity to provide to a RelP, thus keeping control over private data. A user may also establish his own private OIdP server, which enhances control over personal data. However, this requires technical competence that is higher than possessed by typical users.

OpenID does not ensure any trust mechanisms for relationship management between RelPs and OIdPs. Therefore, few service providers wish to become RelPs, which is considered the main reason why OpenID did not experience significant adoption [BHOS12].

Operation of OpenID

Authentication process in a website that supports OpenID login is initiated by a user, i.e. by first loading the website in a UA, i.e. a web-browser. The user then types in his/her identifier in the form of a Universal Resource Locator (URL) or an Extensible Resource Identifier (XRI). The latter option is available since OpenID version 2.0. A user can also provide the identifier of an OIdP instead of the identifier of his/her identity [Opea, CMT08], in which case he/she can choose an identity profile during authentication in OIdP.

Afterwards, RelP performs OIdP discovery based on the provided identifier to find the endpoint URL for OIdP. RelP connects to the endpoint and establishes a secure channel to exchange secret information and enable message signing. Afterwards, RelP redirects the UA to a corresponding OIdP for authentication and optionally identity profile selection.

Subsequently, the UA is redirected to RelP with an assertion of either a successful or failed authentication. This assertion may also include subject-related information. RelP validates and analyses the response, and either allows or denies access to the service.

3.2.2 SAML

General Description

SAML is an authentication protocol that allows exchanging security assertions between parties to enable SSO service [OAS08]. These parties are security domains with trust relationships established between them, thus SAML enables enterprise authentication. The main principle of the protocol's operation is similar to OpenID discussed in Section 3.2.1. However, unlike in OpenID, collaboration between a relying party and an identification provider using SAML is performed only if the parties have a trust relationship and potentially additional information utilisation contracts.

SAML involves three parties: SP, UA and IdP. A SP provides a service valuable to the user. A UA is used by a human user, i.e. the subject, to obtain access to a service provided by a SP. Typically, UA is a desktop or mobile web-browser. An IdP provides SP with identity information related to the subject. In a typical scenario, when a user requests access to a service using his/her UA, UA is redirected to the IdP where the user has to prove his/her identity. Upon successful authentication, IdP sends identity information, including authorisation and authentication-related attributes, as assertions back to SP.

The current version of SAML protocol is 2.0, created after merging an earlier version 1.1 with Shibbolleth and Identity Federation Framework (ID-FF), developed by Liberty Alliance [Ora10]. Earlier protocol versions 1.0 and 1.1 are now deprecated, thus further discussion addresses protocol version 2.0.

SAML protocol is comprised of the three following components [OAS05a]:

1. **Messages** that encapsulate data, e.g. security assertions;
2. **Bindings** that specify the means for transporting SAML messages;
3. **Profiles** that specify complete combinations of bindings that can be used to perform authentication and authorisation.

SAML protocol messages are based on Extensible Markup Language (XML) format. The fundamental SAML message type is security assertions, which contain statements about a subject's attributes and rights. A single assertion message contains one or more statements of the following types:

1. **Authentication Statement** that provides information about the authentication process, e.g. time and method and result of subject authentication.
2. **Authorisation Decision Statement** that indicates whether access to a requested resource has been granted.
3. **Attribute Statement** that provides additional information about the authenticated subject.

Assertions also contain information about the issuer of the message, general information about the subject and potentially indicate conditions of token's validity. For example, a token may be valid at a specific period of time.

SAML also includes Authentication Request and Artifact Resolution message types. Authentication Request specifies messages used for initiation of subject authentication in the IdP, which are sent from the SP. Artifact Resolution messages enable SP and IdP exchange message references, which are afterwards resolved in order to obtain the content of the messages.

Besides message formats, SAML protocol specifies bindings and profiles. Bindings [OAS05b] define how the messages are sent, and profiles [OAS05c] describe a combination of bindings used for communication among specific parties. For example, Hypertext Transfer Protocol (HTTP) Redirect Binding enables adding the SAML message into URL query string of a HTTP message. However, due to URLs length limit, this option is suitable only for short messages, such as Authentication Requests.

Alternatively, POST Binding enables putting a transferred message into a HTTP form that is forwarded to another party (SP or IdP) through a UA. Furthermore, HTTP Artifact Binding profile enables using UA to transfer message references as Artifact Messages, which are used by the receiver to directly contact the sender and obtain the actual message content.

Operation of SAML 2.0

SAML communication sequence begins when a subject uses his/her UA to access a resource in a SP [OAS05c]. This invokes a security check for a given resource at the SP. If the particular SP already contains a user-related security context, it simply checks the context for subject's rights to access the resource and afterwards returns the resource. Otherwise, SP tries to obtain information about the user from an IdP it already knows about or discovered using the Identity Provider Discovery protocol.

After obtaining information about the IdP, a SP creates a request for IdP that includes authentication request message and *RelayState* parameter describing the location of SP. This request is forwarded to IdP by the UA. After receiving the request, IdP checks if it has a security context created for the requested user identity and, if not, requests the user to prove his/her identity. The SAML protocol standard does not specify authentication mechanisms for the IdP, thus passwords, allowing utilisation of certificates or any stronger type of authentication. After successful authentication, IdP creates a security context related to the user identity, meaning that the user is logged in the IdP and future assertion requests are handled without the need for the user to re-authenticate, thus enabling SSO service.

After authentication, IdP validates the request of SP and sends a response back through the UA. The response contains security assertions and *RelayState* parameter value initially issued by the SP. When forwarding the response to SP through UA, the user might be asked for consent to transfer his/her information to the SP.

Assertion Consumer (AC) validates the response, creates a security context for the user at the SP for this and future uses and redirects the UA to the requested resource.

3.2.3 OAuth 2.0

General Description

OAuth 2.0 (further called OAuth2) is an authorisation framework. Unlike previously discussed technologies, it does not focus on authenticating a user and providing parties with the user's identity information. Instead, it defines a protocol for authorisation of a third party service, i.e. Client, to act on a resource owner's behalf when performing certain actions on a resource stored in a Resource Server [Int].

OAuth2, as a recent technology, addresses current needs, such as mobile applications based on web-services. Unlike OAuth v1.0, which specifies its own security mechanisms, OAuth2 relies on security provided by Hypertext Transfer Protocol Secure (HTTPS) protocol, which depends on Transport Layer Security (TLS). Furthermore, OAuth2 enables obtaining long-time possession of tokens by distinguishing short-time tokens and refresh tokens.

As discussed by Dennis [Den], OAuth2 has an advantage over OpenID and SAML 2.0 protocol because it does not use HTTP POST messages and thus fully supports mobile clients, i.e. applications. In certain modes of SAML 2.0 and OpenID, assertions and authentication results, respectively, are sent in HTTP POST messages. Mobile applications launched using URL obtained as SAML response, do not have access to POST messages, thus cannot access information sent in these messages. Although workarounds exist, this feature of OpenID and SAML constrains mobile application development. Thus, OAuth2 addresses this problem by using only HTTP redirects in UA for message forwarding among parties and entirely avoiding utilisation of POST messages.

OAuth2 specification [Int] defines the following four parties involved in the process of OAuth2-based authorisation:

1. **Resource Owner (RO)**: an entity that possesses (contains rights to) certain resources and is willing to grant a third party Client with access to the resources.
2. **Resource Server (RS)**: a server that contains the resource.
3. **Client**: a third-party application (web-based, mobile-based or other) that desires to perform certain actions on a resource in RS.
4. **Authorisation Server (AS)**: a server responsible for authenticating the RO and issuing authorisation tokens to Clients allowing certain operations on a resource.

RS and AS roles can be performed by a single entity. It is noted that relationship and interaction between these parties is beyond the scope of SAML specification.

Operation of OAuth2

OAuth2 authorisation procedure begins when a Client sends an Authorisation Request to RO. The message can be sent directly to the RO or indirectly through AS. At this step, the owner may be prompted to complete authentication procedure. Afterwards, RO sends an Authorisation Grant back to the Client. Authorisation Grant is generally an explicit owner's agreement indicating that the Client is allowed to use the resource. For example, it can be a combination of a user-name and a password or an authentication code received from an AS associated with the RS.

The Client uses the Authorisation Grant to request the AS for an Access Token. The AS performs user (Resource Owner) authentication, validates the Authorisation Token and issues an Access Token to the Client. Access Tokens typically have limited lifetime. To enable token refreshment without the need for user re-authentication, AS can also provide the Client with a Refresh Token that can be used to obtain new Access Tokens.

Having the Access Token, the Client application can request permission to perform certain operations with the resource on user's behalf. In order to do so, it sends a request with information about desired actions as well as the Access Token to the RS, where the resource is located. Afterwards, RS validates the Access Token and performs the requested operations. Validation method is not defined in the specification, but it typically involves collaboration between RS and AS.

As it is noticed by Dennis [Den], unlike authentication response in SAML 2.0, OAuth2 Authorisation Grant and Access Token, obtained by the Client after user's authentication and authorisation, does not provide the Client with user identity information. Thus, an additional request to the AS is needed to validate the Token and to obtain certain user identity data as a resource. On the other hand, this enables invalidating the Access Token on the server side if the token is compromised and thus forbidding the Client's further access to a specific resource.

3.2.4 OpenID Connect

General Description

As discussed in Section 3.2.3, OpenID and SAML do not support mobile applications. Although OAuth2 addresses this problem, there are multiple important aspects that are not defined by OAuth2 specification, which leads to differences and thus non-interoperability of implementations. Furthermore, OAuth2 is designed for third party authorisation, i.e. delegation of rights, rather than authentication. As noted in OpenID Connect specification [SBJ$^+$], OAuth specification includes methods for obtaining Access Tokens needed for accessing resources, but omits definition of access to user identity information.

OpenID Connect (current version 1.0) [Opeb] addresses these and other issues of predecessor protocols and is described by its creators as Application Programming Interface (API)-friendly technology that enables protocol support for native and mobile applications. It is "*a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication* *performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner*" [Opeb]. OpenID Connect is an extension of OAuth2 providing authentication functionality, thus it largely depends on OAuth2 for authorisation and flow of messages. OpenID Connect also defines most of the out of scope parts of OAuth2, thus providing a complete authentication framework specification.

OpenID specification defines the roles of OpenID connect members based on OAuth2 definitions. For example, a RelP is described as OAuth2 Client that requests Claims (information asserted about specific entities). Similarly, an OpenID Provider is defined as OAuth2 Authorisation Server capable of providing a RelP with requested Claims.

Operation of OpenID Connect

OpenID Connect Core specification [SNB+] defines three message flows, specifically Authorisation Code Flow, Implicit Flow and Hybrid Flow that involve different endpoints for issuing tokens. In this section, we analyse Basic Client's implementation [SBJ+], which uses Authorisation Code Flow, that uses `Token Endpoint` to issue all the tokens.

Authentication procedure begins when a RelP prepares and sends an Authentication Request, which specifies a request to authenticate and/or return identity data about a user, to an Authorisation Server. If needed, the latter authenticates the user and asks the latter if RelP is allowed to access user's personal data. Authorisation Server then provides the RelP with a response that includes a `code` (an equivalent of Authorisation Token in OAuth2). RelP uses this data structure to request a `Token Endpoint` for access to user data. After receiving this request, `Token Endpoint` provides a RelP with ID and Access Tokens, as well as potentially a Refresh Token [SBJ+]. ID Token contains information about the authentication event. Among the required parameters, such as the issuer and subject identifiers as well as expiration time, it may include details about authentication processes that were performed.

Identity information access functionality is managed by `UserInfo Endpoint`. After RelP receives an Access Token, it can use this token to request `UserInfo Endpoint` for user identity-related information. Identity information set that is returned by `UserInfo Endpoint` depends on Access Token and previously described Access Token acquisition step. In order to specify requested identity information set, a RelP specifies requested identity fields in the `scope` parameter that is included in the request for Access Token sent to `Token Endpoint`. `scope` may include request for user profile information, email, address and phone number request values. However, it always includes `openid` value identifying that the request is an OpenID Connect request.

As mentioned, OpenID Connect specification also fills multiple gaps that are present in OAuth2. One of the most criticised shortcomings of OAuth2 is the lack of definition of Access Token validation performed in the Authorisation Server. OpenID Connect defines validation procedure not only for Access Tokens, but also for ID Tokens and other related security parameters. Similarly to OAuth2, OpenID Connect uses TLS for securing message transmission.

3.3 Communication for Ubiquitous Computing

This section discusses various industrial approaches of communication systems used for ubiquitous computing. We start with a rather thorough discussion of European Telecommunications Standards Institute (ETSI) M2M architecture specification in Section 3.3.1, and briefly analyse other related efforts afterwards.

3.3.1 ETSI M2M System Architecture

One of the important M2M communication standards developed by ETSI [Eur13d] defines M2M system's functional architecture. This section thoroughly analyses ETSI architecture from the perspectives of communicating components, their identification and communication.

Architecture Components

In ETSI architecture, an M2M network consists of a number of different hierarchically-ordered components that communicate with each other to perform high-level system's functionality, as presented in Figure 3.1. In the top-most level, ETSI-specified system consists of *Device and Gateway Domain* and *Network Domain.* The Device and Gateway Domain is comprised of the following elements:

1. **M2M Device(s)**, responsible for running M2M Applications (referred to as Device Applications (DAs)) using M2M Device Service Capabilities Layer (DSCL). For example, a DA may be responsible for measuring environment conditions and periodically sending data to other devices in the domain or to the M2M Core Network (CN) in Network Domain, by utilising M2M Area or Access Network, respectively. M2M devices may also provide connectivity to M2M legacy devices which are not compliant to ETSI standards. In this thesis, we refer to M2M Devices as D nodes.

2. **M2M Gateway**, which functions as a proxy between M2M devices and CN. Similarly to M2M Devices, a Gateway also runs M2M Applications (referred to as Gateway Applications (GAs)) on top of M2M Gateway Service Capabilities Layer (GSCL). In simple scenarios, it can function as a mere repeater, relaying

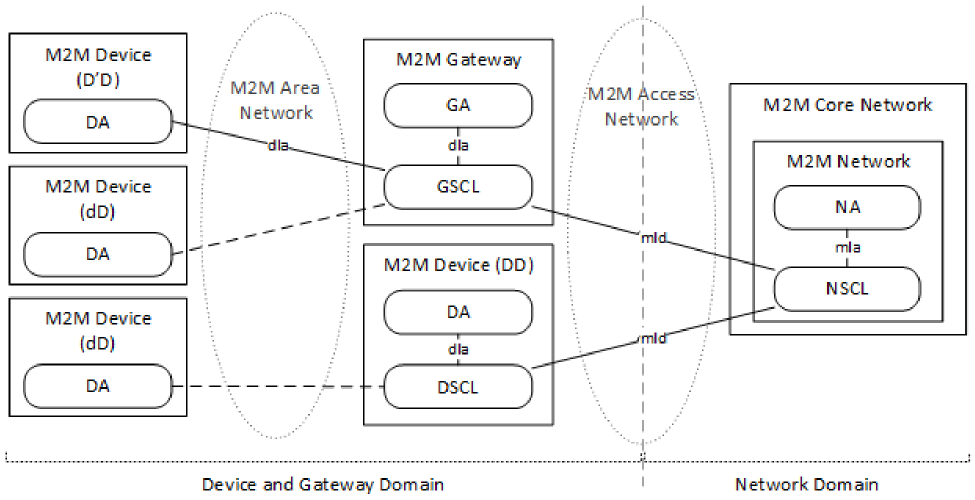

Figure 3.1: ETSI M2M functional architecture [KSB+13, Eur13d]

the packets between M2M devices and CN. However, it often performs more sophisticated operations, e.g. aggregation and compression of data received from M2M Devices, and afterwards transmission of this processed data to M2M Core Network. Such gateway capabilities enable reducing the number and size of messages sent from one domain to another. There can be multiple Gateways in the Device and Gateway Domain. Furthermore, Gateways can also provide connectivity to legacy M2M devices that are not compliant with ETSI architecture. We refer to M2M Gateways as G nodes.

3. **M2M Area Network**, which enables connectivity between M2M devices and M2M Gateway. It may be based on Wireless Private Area Network (WPAN) technologies, e.g. Bluetooth, ZigBee and ISA100.11a, or use local network technologies, such as Wi-Fi, Programmable Logic Controller (PLC) and Meter-Bus (M-Bus). Utilisation of short-range networks allows minimising consumption of energy needed for communication, thus prolonging lifetime of energy-restricted M2M nodes.

Network Domain consists of:

1. **M2M Access Network**, which enables communication among M2M Devices and/or Gateways and CN. It is typically a large area network based on a long-range communication technology, e.g. x Digital Subscriber Line (xDSL), satellite link, Wireless Local Area Network (W-LAN) and Worldwide Interoperability for Microwave Access (WiMAX).

2. **M2M Core Network**, which is primarily responsible for providing M2M Devices and Gateways with IP connectivity to the global Internet. Additionally, it runs service and network control functions, provides roaming and interconnection with other networks. CN types defined in the specification include, among others, 3rd Generation Partnership Project (3GPP) CNs, ETSI Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN) CN and 3rd Generation Partnership Project 2 (3GPP2) CN. In this thesis, we also refer to this component as N node.

3. **M2M Service Capabilities**, which is a layer of abstraction that provides common functions and hides network and technology specifics. These functions can be used by M2M Applications. To differentiate this layer from similar Service Capabilities layers in M2M Devices and Gateways, we refer to it as Network Service Capabilities Layer (NSCL).

4. **M2M Applications**, which utilise M2M NSCL to run service logic. These applications are responsible for high-level system's logic and user interface utilised by consumers to access the system's services. We refer to applications running in Network Domain as Network Applications (NAs).

The Network Domain also includes two layers of management functions: Network Management Functions Layer and M2M Management Functions Layer.

As mentioned above, some DAs are running in devices that do not have a Service Capabilities Layer (SCL) or are even ETSI-incompatible. Thus, M2M devices are classified into three categories. The first category comprises devices that contain a SCL and are ETSI-compatible. The second category contains ETSI-compatible devices that do not have a SCL. Such devices use *dIa* Reference Point (RP) to connect to a Gateway and utilise the latter's GSCL. Finally, the third category is comprised of legacy devices that do not have a SCL and are ETSI-incompatible. These devices can be connected to the network through a D or G node by using Gateway Interworking Proxy (GIP) or Device Interworking Proxy (DIP) capabilities, respectively, provided by the supporting device's SCL. ETSI does not specify the connection between a legacy device and the node that connects it to the network. Like in [Eur13d], we refer to the devices of the three categories as D Device (DD), D' Device (D'D) and d Device (dD), respectively. Connections between these devices and other architecture components are shown in Figure 3.1.

Identities in M2M Network

M2M systems utilise identities to establish robust and secure connections between network components and to communicate afterwards. This section discusses IdM system specified in ETSI functional M2M architecture [Eur13d].

Identifiers in M2M Network The overall functioning of ETSI M2M system is based on a set of identifiers defining different components of the system. Based on ETSI specification, this set is divided into two identifier models. The first model consists of identifiers that are used by the features specified in the specification. More specifically, these are M2M components and their logical relationships, defining the ETSI M2M platform. The specification focuses on the latter model of identifiers, and the provided instructions related to these identifiers are normative. Since an M2M system greatly depends on underlying communications network (at least for the Access Network part), there is another hierarchy of identifiers that addresses identification and communication at lower level, concerning mostly SP and Access Network (AcN) parts. However, these lower-level communications are out of scope of the specification and the provided details are merely informative. Therefore, in this thesis we focus on the first model, and mention only the important aspects of the second model.

The following are the components that comprise the normative model of identifiers:

1. **Application Identifier (App-ID)** uniquely identifies an M2M Application registered at a certain SCL. If there is more than one instance of the same application registered at the same SCL, App-IDs must be unique. App-ID is used by other parties for interaction with the application.

2. **Node Identifier (Node-ID)** identifies a node, i.e. the logical representation of components in a M2M Device (D Node), Gateway (G Node) or Network Device, registered with a particular SP. A node is comprised of a SCL, M2M Service Bootstrap Function (MSBF) and M2M Service Connection Function (MSCF). A Node-ID is created during M2M Bootstrap procedure or when pre-provisioning a D/G Node with SP configuration parameters. If a device registers with several SPs, then there can be multiple nodes deployed on a M2M device or gateway, but they all must have unique Node-IDs.

3. **SCL Identifier (SCL-ID)** uniquely identifies a SCL. According to the specification [Eur13d], SCL-ID may have the same value as Node-ID.

4. **Service Connection Identifier (CONN-ID)** identifies the connection between a DSCL/GSCL and NSCL. Such a connection is used for communication between a D/G Node and the Network. A connection between the parties is not permanent, i.e. after it is terminated, it can be re-established or even replaced with a new connection, thus the CONN-ID of a connection between two parties is not static and can change in time.

5. **Service Provider Identifier (PROV-ID)** identifies a SP. The value of this identifier is unique and static.

6. **MSBF Identifier (MSBF-ID)** is a unique and static value that identifies a MSBF. MSBF-ID is assigned by a SP.
7. **MAS Identifier (MAS-ID)** is a unique and static value identifying a M2M Authentication Server (MAS) and is provided by a SP.

In addition to the mentioned model, the specification provides an informative part about lower-level identification used for addressing and communication purposes, and instructions for mapping between the two models (see Figure 3.2). The lower-level model consists of two layers: M2M Service Device Level and M2M Transport Device Level. M2M Service Device Level is further divided into M2M Application Identifier, M2M Service Subscription Identifier and M2M Service Device Identifier. M2M Transport Device Level identifier set consists of M2M Access Network Subscription Identifier, M2M Access Network Device Identifier and M2M Access Network Address.

Distinction of the layers enables a logical and potentially physical separation of functionality provided by a high-level SP and lower-level communication part, based on capabilities of Access Network Provider (ANP). It also implies that identifiers in different layers are utilised by different parties, i.e. SP and ANP.

The lower-level model of identifiers addresses the two cases when SP and ANP are the same entity and when they are separate entities associated with a pre-established relationship. Furthermore, these two layers can refer to a single or two separate physical devices (a service device and a transport device). For example, a D'D contains only Service Device Level identifiers, whereas the Gateway the device connects to provides the Transport Device Level identifiers.

Some of the lower-level identifiers can be omitted in the implementation of the network. For example, a DD acts as both a Service and Transport Device, which means that the hardware information is the same in both levels. In this case, Access Network Device Identifier is sufficient and M2M Service Device Identifier is redundant.

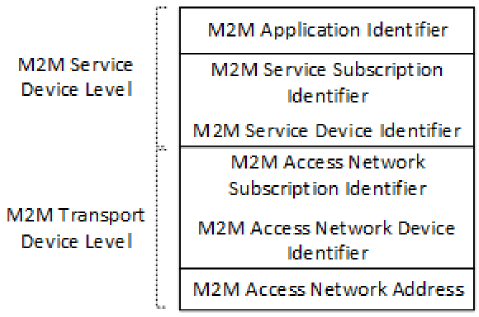
M2M Service Device Level
M2M Transport Device Level
M2M Application Identifier
M2M Service Subscription Identifier
M2M Service Device Identifier
M2M Access Network Subscription Identifier
M2M Access Network Device Identifier
M2M Access Network Address

Figure 3.2: Low-level identification system of ETSI M2M architecture [Eur13d]

Furthermore, when the SP is the same entity as ANP, the M2M Service Subscription Identifier carries the same information as the M2M Access Network Subscription Identifier, thus one of them can be omitted.

Connectivity in the Access Network may be enabled by diverse technologies, e.g. UMTS, a satellite connection and xDSL, as mentioned in Section 3.3.1. The choice of a communication technology may influence the identifier formats used in the system's implementation.

Identification and Connection of D/G nodes In order to provide secure M2M communications, node identification and authentication processes are of prime importance. In ETSI M2M system, this is addressed with a hierarchy of keys, identifiers and a set of procedures that are utilised to identify a D/G node during connection establishment between the node and the M2M system.

The top-level key used in M2M identification process is the M2M Root Key (Kmr). This key allows authentication and key agreement between a D/G Node and the SP. Kmr depends on Node-ID and PROV-ID, thus every M2M node has a separate Kmr shared with the SP. Furthermore, a Kmr is used to generate M2M Connection Keys (Kmcs) and afterwards M2M Application Keys (Kmas). Kmc is involved in authentication of a M2M connection. Similarly, Kma is utilised in authentication and authorisation of an M2M application. In addition to that, Kma allows protecting the data traffic of an application. Keys Kmc and Kma are derived using CONN-IDs and App-IDs, respectively.

A device may obtain a Kmr needed for further key generation in several ways. Kmr may be preconfigured, i.e. stored in a device's Secured Environment Domain, e.g. in a Universal Integrated Circuit Card (UICC) module. Alternatively, the device may be provisioned with the material required to generate the keys, including Kmr, during service bootstrap. As the specification describes, Node-ID, SCL-ID and a list of NSCLs identifiers can be produced as additional elements of M2M service bootstrap.

There are multiple ways to perform the bootstrap procedure, depending on equipment and communication technologies that are present in the network and whether Access Network provides assistance. In the Access Network-assisted scenario, a node can obtain the information required for key derivation and to locate other network members by utilising network credentials. Independent scenarios, however, are based on pre-provisioned information set by the manufacturer in D/G devices. For example, Identity-Based Authenticated Key Exchange (IBAKE)-based automated bootstrapping procedure requires pre-provisioned identifiers, as well as a pre-shared secret (key or a password). Thus, in this case the same information would be separately and securely provisioned into the D/G devices and the MAS before deploying the system.

ETSI specification also defines a temporary service connection that is established between a D/G node and the service provider in order to gain connectivity and access to MSBF which is then used to perform service bootstrap. However, in order to create a temporary connection, the connecting node has to be pre-configured with service provider address as well as identity and secret parameters.

After a D/G performs M2M bootstrap and service connection procedures, it contains Kmr and Kmc keys, as well as *mId* RP. This information can be used by the node to perform service registration procedure, which involves registration of a node's SCL in the NSCL. In addition to that, when applications are registered in the node's SCL and Kmas are generated, the node can communicate with other nodes in the network via the Network node. Thus, it can either request or provide functionality enabled by M2M applications.

Addressing of Applications

Communication of devices and applications is based on a feature of ETSI M2M architecture called M2M Point of Contact (PoC). PoC is used by M2M network to locate the device's SCL and afterwards the required application by looking up the information related to the targeted SCL that is stored in NSCL and translating it into a URI or IP address. These addresses are then used to perform communication with the device and application using Representational State Transfer (REST) protocol. The further part of this section contains a description of a D/G node SCL and application registration procedure, as well as processes performed to locate and reach the desired application.

Initially, when a D/G node is connecting to the network, it has to perform bootstrap, network and service connection as well as service registration procedures, as described above. During service registration, a SCL (DSCL or GSCL) has to register to a NSCL. When performing this procedure, the registering node creates a *SCL* resource, i.e. a data structure, in NSCL to provide the latter with information of the node's SCL. This resource contains another resource, *M2MPoC*, which holds information specifying how to reach the registered SCL, i.e. the D/G node that contains it.

After SCL registration, M2M applications can register in the local SCL. Similarly to SCL registration in NSCL, this procedure involves creation of another resource *application* that stores information about an application, but in this case the information is stored in the local SCL, i.e. located in the same node as the applications. The SCL can then also register applications in the remote SCL, i.e. NSCL. The ETSI specification does not specify what and when decides to register the application in NSCL, thus it is assumed that it depends on the implementation specifics of the application and the system.

After the application registration is complete, NSCL knows about the SCL and the applications that have registered to it. Thereafter, it might be requested by some other party, e.g. an application in another D/G node, to send a REST request to one of these applications. To successfully locate and address a destination application, NSCL performs a two-step approach using PoC information:

1. NSCL locates the SCL where the application is registered by analysing the PoC information. It can be either DSCL in a DD or GSCL. In the case of a DA running in D'D or dD, the NSCL would try to locate the GSCL to which the mentioned D nodes are attached. Information about the SCL's location is taken from the *M2MPoC* resource stored in NSCL.

2. NSCL transfers the request to the located SCL. The request contains identification (App-ID) of the target application. The local SCL then locates the target application and delivers the request to it.

It is emphasised that for a SCL to reach NSCL, PoC functionality is not needed, because the former is expected to already contain information about the NSCL, which is initially used for SCL registration. D/G nodes are pre-provisioned with this information (e.g. in a UICC card), or provided with it during bootstrap procedure.

The scheme of location detection and addressing of an application also supports advanced connection features, such as multi-homing and mobility. In the case of multi-homing, an SCL simply provides an NSCL with several *M2MPoC* resource records. In case of mobility, an SCL is provided with capabilities of refreshing the location and accessibility values in *M2MPoC* resources of NSCL. Furthermore, the specification allows service capabilities to decide which of the registered channels to reach a node registered in a SCL, thus the best connection of those specified with *M2MPoC* resources can be selected. Finally, the architecture is capable of detecting situations when a D/G node is not reachable, e.g. when no response can be received. In such a situation, the connection specified by *M2MPoC* entry in NSCL is marked as "not reachable", which makes the system use alternative ways for contacting the device, e.g. by sending wake up messages using Short Message Service (SMS) [Eur13d].

3.3.2 MTC Standard

Machine-Type Communications (MTC) is a type of M2M communications that uses Public Land Mobile Network (PLMN) technology as the Access Network. A mechanism of such communications is specified by 3GPP [3GP11a, 3GP11b]. According to the specifications, MTC is a type of communication that involves at least one device that does not necessarily require human interaction. However, as identified by Song et al. [SKSS14], whereas the presented ETSI M2M architecture focuses on the functional M2M aspects and specifies a fully connected functional platform, standards provided by 3GPP are related to signalling and communication among distinct devices.

MTC specification defines transport, subscriber management and M2M device triggering based on 3GPP access technologies, e.g. UMTS. Unlike ETSI specification, MTC treats all devices uniformly as User Equipment (UE). Thus, enhancements of MTC created for M2M communications are also available for regular communication devices, e.g. mobile phones.

MTC defines communication among UA located in UE and an MAS located outside the operator domain. To support communication between the two parties, MTC specifies two additional components:

1. **Services Capability Server (SCS)**, which connects AS to the network. SCS may be controlled by the mobile operator or the provider of MTC service that provides the applications.

2. **MTC Interworking Function (MTC-IWF)**, which is an additional control node for MTC, located in the home network. It is mainly responsible for device identification and triggering. As mentioned in [SKSS14], this component uses E.164 MSISDN or External Identifier to find the equivalent International Mobile Subscriber Identity (IMSI) of a device and decide on the best triggering method, such as sending an SMS that includes an Application Port Identifier indicating a targeted application.

Unlike ETSI M2M architecture specification that defines two sets of identifier systems, MTC uses only two identifiers, i.e. internal and external identifiers. The internal identifier, i.e. IMSI, is defined inside the core network and has local scope of validity. The external identifier consists of the domain and local identifiers, specifying entity in the operator's domain.

3.3.3 oneM2M

The capabilities of M2M technology have encouraged standardisation processes performed by a number of organisations, including the already mentioned ETSI and 3GPP, which lead to potential incompatibilities and isolated vertical structures. In order to address this problem, ETSI together with other organisations have formed a global M2M organisation, *oneM2M*, established in order to enable global M2M communications by defining a common M2M service layer [one12, SKSS14]. It also addresses coordination of development processes associated with the current M2M service layer standards and common features based on needs of vertical market aspects.

With a common service layer, oneM2M seeks to define methods for various functions, including M2M protocols, APIs, security, privacy, reachability, discovery and management. oneM2M expects to minimise deployment expenses, simplify application development, reduce the time-to-market and prevent overlaps of standards.

Chapter 4 Vision and Requirements

The future of communications involves participation of multiple various devices finding and interacting with each other to achieve individual or common goals. This chapter presents our vision of an IdM system that supports dynamic communications among multiple devices and thus enables advanced user-oriented applications. An analysis of incapabilities and drawbacks related to current device communication technologies discussed in Chapters 2 and 3 is presented in Section 4.1. Afterwards, Section 4.2 thoroughly describes the vision of our system. Section 4.3 describes the indicated imposed on the envisaged system. Finally, Section 4.4 presents several use cases of how our envisaged system could be used.

4.1 Problem Description

Currently, there are two prominent approaches to intelligent device-to-device communications: M2M and IoT. As described in Section 2.2.1, M2M is a technology that enables communication among devices with no or minimal human intervention. Although M2M communications technology is attracting large interest, the current M2M technology specifications and implementation cases [Eur13d, 3GP11a, Eur13c, Eur13b, Eur12, Eur13a] are static, proprietary and limited to specific application areas. Therefore, M2M systems consist of ad-hoc devices that communicate with other pre-defined devices, which leads to a problem of multiple isolated vertical systems, described by Clayman and Galis *silos* [CG11]. Furthermore, these systems omit the concept of an owner or a system's user, because it is assumed that each of the private systems are utilised and managed by separate isolated users.

Another approach to enable intelligent communication among devices is IoT. As noted by Cisco [Eva11], IoT is an evolutionary step of the Internet that is now becoming a communication media not only for human users, but also for intelligent devices. That is, IoT is responsible for carrying large amounts of machine-type information, not intended to be processed or utilised directly by users. Unlike M2M, IoT enables connecting any and every device with each other if needed, thus the scope of peers any device can reach is not limited. Furthermore, IoT devices are not dedicated to a single application and can be aware of the context that includes human users.

Although IoT potentially allows global device communications, the infrastructure consisting of the Internet, servers, applications and user-held devices is currently geared towards human users, their decisions and data utilisation, whereas devices perform low-level operations involving information delivery and processing. The mentioned advanced device communication technology requires devices to be able to obtain the necessary information about each other including their features, capabilities, intentions and ownership. In other words, communications should include semantic information related to communicating peers.

Unfortunately, currently even a reliable consistent identification of a communicating device is not available. Although MAC addresses provide a certain type of identification, it merely identifies the network interface and thus is only useful in a local scope and merely for a lifetime of a physical connection. However, a device might have several network interfaces, and use different or several of them to connect to a certain peer, thus determining identity of connecting device is complicated.

4.2 Our Approach

In order to solve the mentioned issues and enable global, autonomous and dynamic interaction among devices, we envisage a need for global identification of communicating entities and particularly an IdM mechanism primarily dedicated for use by devices in an automated manner. As mentioned above, the identification system should allow devices to learn each other's identities, i.e. functionality, capabilities, goals and ownership, perform discovery of devices and establish customised identity-based connections with selected devices.

The vision analysed in this chapter, as well as its implementation presented in Chapter 5, aims at describing a system that utilises capabilities of IoT to build a platform of recognisable communicating entities, i.e. devices that are uniquely identified and contain their own identities, similarly to human users. Furthermore, identification of devices involves information about their owners. The latter feature enables device authentication that is aware of the device's user, thus enabling automatic, user-oriented applications and device communications.

In the remainder of the document, we use term *M2M* to refer to autonomous intelligent communication among devices, without referring to current M2M system implementations or ETSI specification.

Although the discussed technology is designed for utilisation by devices, it is eventually targeted towards enhancing functionality and experience of human users, e.g. regular consumers that carry one or more intelligent devices with them. Therefore, the system should also involve user-related information. However, besides including user information in authentication processes, the IdM system also has to address the following aspects related to user's behaviour:

- **Device sharing**. In many environments, including home and workplaces, there are certain devices that are utilised by different users at different time. Thus, the devices should be user-aware and allow services and other peers to get user-related information, which could afterwards be used for device authorisation or adaptation of communications.

- **Multiple identities**. Users may desire or be demanded to use certain identities in specific environments. For example, *"bring your own device"* policy increasingly used in workplace environments allows employees to bring their own devices for performing work-related activities. However, companies may wish to restrict possibilities to perform tasks not necessary for work, e.g. using social networks. Similarly, a person might not want to use work-related credentials when he/she is accessing services from home. Thus, a device should be able to represent different identities related to the same user that are used in specific areas. Therefore, there is a need for a mechanism that enables selection of active identity.

- **Multiple devices**. A user may be equipped with multiple devices at a given moment. Before he/she can access a certain service using more than one device, it is necessary to authenticate the devices. However, typing in the login credentials in each device is impractical. The system has to address this issue by providing a mechanism for simplified login procedure for service access using multiple devices.

The configuration of a system oriented towards a human user is illustrated in Figure 4.1. Here, user John utilises his set of devices, i.e. a smart phone, a tablet personal computer (PC) and a laptop PC, to reach multiple services available to him. When John wishes to use a certain service, he sets the devices to provide the service with one of John's identities that the service is able to recognise. In the presented figure, any device is used with one or more of the identities. Furthermore, a given identity allows authenticating to multiple services, which is indicated in the figure using rounded rectangles representing association of identities and services.

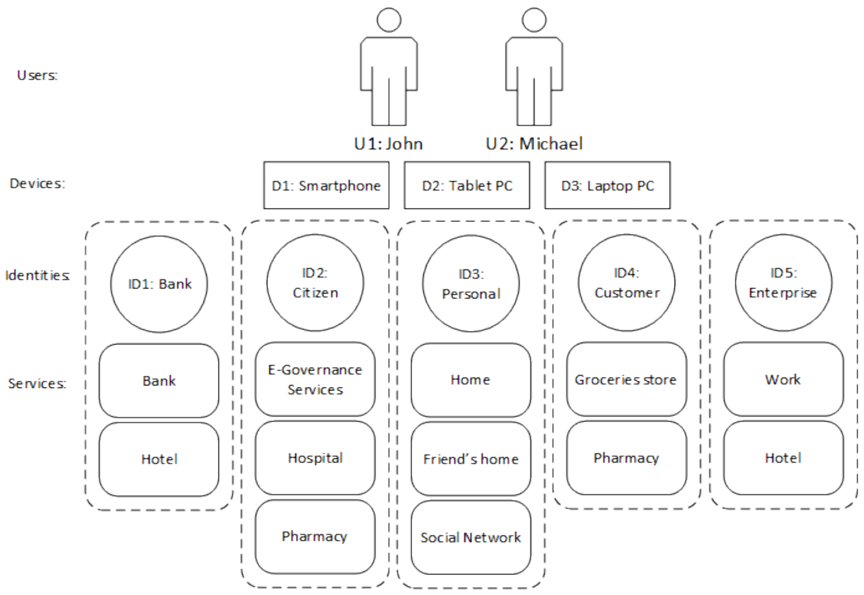

Figure 4.1: Users accessing services using devices configured with multiple identities

What is more, another user Michael, working in one or more shared environments with John, is allowed to utilise the same devices that John is using, thus making devices shared and required to know the exact user and his/her identity.

4.3 Challenges

The aspects of the envisaged IdM system presented above create multiple challenges. Most of them are related to multiplicity and varying number of entities as well as dynamic relationships. In this section, we elaborate the list of challenges in more detail. As mentioned, the technology is intended to enhance device-to-device communications needed for advanced user-oriented applications, thus the challenges presented further reflect actions from a user's perspective.

The following are the challenges imposed on the envisaged IdM system:

1. **Authentication using a user-related identifier**. Although this document discusses a communication system for users and user-oriented applications, the system is primarily based on communications among devices. To enable automatic communications among devices and providing users with services, devices should be aware of their users and present their identifiers to other peers for authentication and further communication.

2. **Creation and management of identities**. As already mentioned, a user has multiple identities that he/she wants to use when working in different environments and accessing various services. The system should provide a way to create new identities as well as to present them for services during registration and later authentication. Additionally, the system should support management of relationships among different identities, e.g. user-device relationships.

3. **Authentication of multiple devices**. This challenge can be treated as the next step of *"single user, multiple services"*, that is currently approached with SSO feature with SAML, OAuth and other technologies. Although in our situation there is still a one user accessing multiple services, the user utilises multiple devices, potentially simultaneously, and expects a seamless access to a certain service with all of his/her devices after the initial authentication, performed on one of his/her devices. Thus, the IdM system has to support convenient authentication of multiple user's devices, i.e. a feature we call SDSO.

4. **Discovery of identities suitable for authentication**. The number of services accessed by a typical user is continuously growing. This also infers that the number of identities created by a user for different sets of services will only increase. Thus, eventually manual discovery of available services and selection of suitable identity might become practically infeasible. Identity selection should be performed semi- or fully automatically. Thus, the developed system should provide a mechanism to discover services, learn about their accepted identities and prompt the user for identity selection only after the list of identities is minimised to those that the service accepts.

5. **Privacy Protection**. Utilisation of the envisioned technology might cause a threat to user's privacy. Even more, the imposed threat may be larger than potential threat caused by classical services, i.e. when one identity is used per service. The scale of the threat is influenced by the following reasons.

   First, an identity might be used by multiple services. However, some identity data fields might be intended for use only by one or several specific services. Hence, there is a need for identity data access management mechanism that is aware of services that access identity data.

   Second, although a user presents the same identity to multiple services, usually he/she does not want the services to be able to recognise that it is the same user. In other words, services should not be able to communicate to each other and share details about the user, e.g. user's behaviour, without explicit user's consent.

   Third, details about the set of user's devices and each individual device can reveal additional details threatening user's privacy. For example, from a set of devices, the service might be able to predict what the user intends to do, e.g. work, go on vacation, exercise or other. Although this information is at first glance innocent, it may be exploited for malicious intentions. Thus, disclosure of such information has to be controlled.

6. **Utilisation of existing technologies**. The problems explained in this chapter are already present. Therefore, the provided solution is already needed and thus has to utilise currently available technologies. However, the system should also enable potential future extension based on evolution of ICT.

4.4 Envisaged Use Case

This section presents a use case that illustrates a realistic situation of the system's utilisation from the user's perspective.

In our envisaged scenario, a user John has several identities installed in his devices (a smart phone, a tablet PC and a laptop PC, as shown in Figure 4.1). He uses these devices to access different services. When John goes to his friend Michael, he wants to obtain access to the WiFi network of Michael's home and access the Internet from his devices.

In the classical scenario, John would ask Michael for the network's password. However, typing network password into all of his devices is not practical. Besides, the password itself can be very complex for security reasons, which increases the rate of errors during authentication. Thankfully, Michael's network access is managed from a cloud IdP server which also stores John's identity. The latter identity is specified in Michael's home network as Michael's friend, thus the system may provide John's devices with access to the network if they can prove that they belong to and are currently controlled by John. To obtain access to the network, John takes one of his devices, e.g. his smart phone, selects the identity and performs authentication. Authentication to the identity requires John to type in his personal user name and password. When the device is authenticated by the IdP, Michael's home network provides access to this device, obtains a list of other John's devices that he potentially has at the moment and suggests authenticating additional devices. John selects additional devices from the list by using the same smart phone and sends a response back to the server. After this procedure, the specified John's personal devices obtain access to the network.

Analogous steps may be performed when John wants to obtain access to the network of his workplace. He then simply selects a network, authenticates with his enterprise identity and specifies additional devices to connect.

When going to stay in the hotel, John can use one of his identities, depending on the circumstances of his stay. If it is a personal trip, the stay at the hotel and access to the hotel's network is charged directly to John, thus his personal details, including address, bank account number or a credit card information are presented to the hotel's system. If John is staying at the hotel for business reasons, he might want to provide the hotel's system with his enterprise identity. This identity may store bank account number only for internal use, e.g. for paying salary. Thus, in this case the enterprise IdP would provide another virtual identity, where John's bank account number is replaced by his company's account number. The further authentication process matches the one described in the case of authentication to Michael's home network.

Chapter 5 IdM System Proposal

This chapter presents the proposed IdM system that addresses future communications based on autonomous devices and geared towards user-oriented services. Analysis of the system starts with Section 5.1, which presents an abstract view of the IdM system's components and entities it involves. Afterwards, Section 5.2 describes the identification mechanism used in the system. Section 5.3 presents the system's functionality and responsibilities. Subsequently, Section 5.4 presents the system's static view by focusing on internal system's structure. Finally, Section 5.5 analyses the system from dynamic perspective by presenting system's actions and operations performed by different components. In order to avoid ambiguity, system's analysis includes Universal Modelling Language (UML) diagrams, where applicable.

5.1 System's Entities

This section describes entities that are distinguished in our IdM system and directly or indirectly participate in M2M communications. In our system, these objects are unambiguously identified with identifiers and described with identities, as it is presented further in Section 5.2. The remainder of this section is divided into 2 parts. The first part discusses system's logical entities, which define the data involved in the IdM model. The second part addresses the physical infrastructure and presents physical entities that enable our IdM system.

5.1.1 Logical Entities

As already mentioned, our proposed IdM system addresses a situation where multiple potentially shared devices communicate with each other or certain services to obtain desired data or functionality provided by certain parties. Based on this simple definition, we identified the following system's entities:

1. **User**: a human user that utilises one or more devices to access services, such as

a social website, WiFi connectivity or advanced context-aware and AmI-based functionality, e.g. a smart conference room or a smart payment system in a shopping mall.

2. **Device**: typically a piece of hardware equipment communicating with other hardware over a certain connectivity channel in order to obtain information or a service. In our system, devices are controlled by human users. However, a single device may be shared, i.e. used by several users during different time periods. Therefore, decisions of whether to allow a device to access certain functionality or resources are based on information about the current user of the device.

3. **Domain**: an area where a device or a user operates, or, in other words, an institution that provides the identity and defines its utilisation scope. It can be a workplace, a specific service or home environment. Together with information about the user and/or device, a domain provides additional information enabling access-related decisions made on the side of Service Provider. Furthermore, a domain itself may define specific Service Providers that a user and/or device can reach, and perform authorisation checks in IdP before forwarding authorisation to the service. This enables two-step authentication from both the side of a Domain and a Service Provider.

4. **Service Provider**: a party that provides services to requesting users that utilise certain devices.

As described above, system's entities are related to each other. For example, a device, e.g. a smart phone, can be controlled by a certain user. Furthermore, the device and the user both represent a certain organisation, i.e. domain. We suggest that this information, specifically user, device and organisation, is included into consideration when deciding whether this combination is authorised to access requested features in a service. Thus, we suggest using identifier parts to specify distinct users, devices and domains. These parts are afterwards combined into a single user-device-domain combination that is an identifier representing a specific user utilising a certain device and coming from a certain domain. Existence of such an identifier also means that the domain allows a certain user to utilise a particular device to access services. This combined identifier, as further specified in Section 5.2, is the key-structure that is presented by a device to a service when requesting access.

As mentioned in Chapter 4, there may be multiple relationships among users, devices and domains. A single user might contain several devices and devices may be shared among multiple users. Furthermore, as the mentioned *"bring your own device"* policy used in workplaces illustrates, devices can be utilised in different operational areas, i.e. domains, and utilisation of identities related to these domains may be restricted based on internal domain policies. Thus, a single device may have multiple user-device-domain combinations associated with it even for the same human user.

5.1.2 Physical Entities

Our proposed system is a distributed system and follows federated IdM model. Its infrastructure consists of several physical parts that relate to logical entities defined above. The following are the physical system's entities representing the general view of the systems infrastructure, as presented in Figure 5.1:

1. **Device Subsystem (DS)** is a middle-ware layer stored in a user's device and providing a set of functionality to Operating System (OS) and user applications. DS is potentially located in a secure element or trusted execution environment to prevent unauthorised modification of data or corruption of valid operations.

2. **Service Subsystem (SS)** is a part of service infrastructure that enables using the protocol and infrastructure of the proposed system for authentication of a device and user. Furthermore, this subsystem enables authentication of additional user's devices, as discussed later in this chapter. From the perspective of user's devices, services and therefore SSs may be either local or remote, e.g. WiFi connection or a social website, respectively.

3. **Identity Provider Subsystem (IdPS)** represents the part of IdP responsible for storage of all the identity data as well as authentication of users/devices and services. Unlike SS, IdPS is considered a remote entity, i.e. not in the location of a device and services (when the latter do not include IdP functionality). It can be either a private or public entity, installed by a private user at home or a public party in a cloud, respectively.

4. **User** represents an actor that uses one or more devices to access and utilise services. In turn, services learn about the user's identity and obtain related identity data by communicating with IdP.

IdPS is a new element in device-to-device communications. It may seem controversial to trust a third party for storing such sensitive data as identities and authentication secrets, but it is needed in order to provide the following benefits and solve certain problems, as discussed further.

In the broad sense, IdPS is required for automated communications and functionality, since an identifier alone is not always sufficient to perform reliable and effective communications. For example, a communicating service and a device of a specific domain may decide they can trust each other if 1) the parties can prove their identities and 2) the service and IdP storing domain's identities are connected with

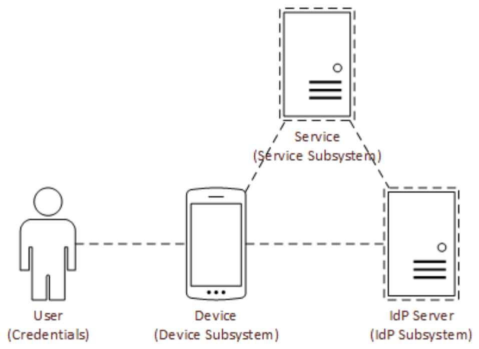

Figure 5.1: High-level structure of the system

a static relationship. However, as mentioned in Chapter 4, devices should be allowed to dynamically establish connections when no initial static relationships are defined. Thus, additional information about entities has to be provided in such a way that it is easily discoverable and available for devices during authentication and further communication. These storage, search and discovery features are provided by IdPS.

Storing identity data in a remotely accessible server also enables freeing devices from having to store identity data in their own memory, which would be rather space-consuming if a device contains multiple identities. Furthermore, our design enables global entity discovery by specifying certain criteria, if the IdPs themselves are discoverable. This also enables device mobility, i.e. changing communication addresses, and multi-homing. Information related to device addresses and access technologies may be stored in IdPSs as part of device identities.

Furthermore, as already mentioned, the system's design is intended to enable regular home users to create private subsystems of identifiers for possessed devices and self-established identities, which can then be provided to local or remote services that trust user's IdPS, e.g. a friend's home network. However, in order to enable remote services to access user's private IdPS, the user establishing an IdPS has to obtain a public IP address enabling direct access to the IdPS. Although technically possible, the current situation indicates that a rather small number of private Internet users have public IP addresses. Thus, providing IdM functionality as a service in the cloud enabling public access seems a more rational choice for private users. However, we allow a user to establish an IdPS of our system personally for his/her local services at home, work or other private space.

Additionally, having identities in the cloud improves privacy, as long as the IdP itself can be trusted. More specifically, our system requires that both the user and the service are authenticated before the service obtains user's identifier and identity information. If the service cannot prove its identity or it is not trusted by IdP, as well as when it appears that the device is asking for access to a different service than the one requesting user's authentication, the service does not learn the user's identifier and/or identity data.

Finally, as already mentioned, IdP is intended to be established as a cloud service and thus provide typical cloud benefits, such as scalable capabilities of computing and storage, as well as global public access.

The exact functionality, structure and collaboration of subsystems are discussed more thoroughly in Sections 5.3, 5.4 and 5.5, respectively.

As mentioned above, the system is distributed, because the design allows having multiple IdPSs managed by multiple IdPs. Similarly, there can be multiple services managing separate SS, as well as multiple users with their own devices that contain DS. The system enables each component to communicate with other system's components of different types. Thus, SSs are allowed to communicate with one or more IdPS instances, making the system a federated IdM system.

5.2 Identification System

As discussed in Section 5.1, the proposed system consists of several communicating entities. In order for these entities to discover each other and obtain the necessary information related to peers, they hold their identifiers and present them to other parties on demand. This section discusses the identifiable entities and format of their identifiers. Furthermore, it describes how the IdM system supports identity data management.

5.2.1 Requirements for Identification

IdM system presented in this document enables identification of logical entities specified in Section 5.1, which directly or indirectly communicate with each other. Before specifying structure of identifiers, we analyse the requirements imposed on entity identification:

1. It is clear that our identifiable entity set comprises a hierarchical structure. For example, an IdP holds identity data and provides identification of users and devices associated with one or more domains. Furthermore, a domain may contain one or more devices and provide them with identifiers. Similarly, a domain may involve one or more users, e.g. employees, who use their company's devices. Finally, a device may be used by one or more users. An identification system should reflect these hierarchical relationships.

2. In order to enable global communication, identifiers have to be globally discoverable and have consistent and unique meaning across different domains.

3. An identifier has to include all the necessary information for identification of a specific communicating unit. For example, a communicating user's identifier should uniquely define the user, device and domain. Otherwise, if part of this information, e.g. the device identification part, was stored as part of a user's identity, obtaining this information would require additional requests to IdP, which would happen rather often due to importance of device identification.

4. It is desirable that identification mechanism enables easy access to identity data related to different identities.

5.2.2 Identification System

In order to meet the requirements specified above, our IdM system contains an a tree-structure of identifiers based on URL format. As discussed in Section 5.1, a standalone identifier represents the combination of partial identifiers. The universal identifier format follows the scheme `idpID/domIDPart/devIDPart/userIDPart` that constitutes a valid HTTP URL (see Figure 5.2) and contains the following fields:

1. **idpID**: identifier of an IdP. This is a standalone identifier representing a valid URL. The `idpID` part is the authority component, i.e. the address of an IdP. It is either a symbolic value, i.e. a Fully Qualified Domain Name (FQDN), or an IP address.

2. **domIDPart**: partial identifier of a domain specifying an organisation or scope where devices and users are registered. This element can be treated as the root element of an identification hierarchy in a company. However, `domIDPart` is a partial identifier, meaning that it needs an `idpID` part pre-pending it in order to represent an addressable and unique identifier. There may be several different domains with the same `domIDPart` registered in different IdPs.

3. **devIDPart**: partial identifier of a device. It is created and assigned to a device by a certain domain. Typically, this identifier is formatted accordingly to domain's regulations. `devIDPart` is static, i.e. it installed in a device and does not change if the device is used by different users identified in the domain.

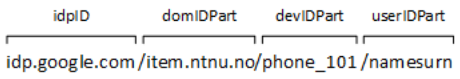

Figure 5.2: An example identifier used in the system

Similarly to `domIDPart`, `devIDPart` is not a standalone identifier and needs to be combined with `idpID` and `domIDPart` to constitute a complete and unique identifier.

4. **userIDPart**: partial identifier of a user. Together with the previously mentioned identification components, this identifier uniquely identifies the user coming from a specific domain that is utilising a device.

As discussed above, identification in our IdM system is based on a family of identifiers that are made of one or more sub-components, i.e. partial identifiers. A certain combination of partial identifiers constitutes a complete identifier that may indicate different entities and contexts, as specified below:

1. **Full user-device identifier udevID** specifies the user and the exact device that is connecting to a service on user's behalf. It follows format `idpID/-domIDPart/devIDPart/userIDPart`. The `devIDPart` and `userIDPart` are both registered in domain identified by `domIDPart`, which stores and manages their identity data in an IdP identified by `idpID`.

2. **IdP identifier idpID**, as discussed above, completely and uniquely identifies and provides location information to IdP in the web. This sub-component is the only one that can be used as a standalone identifier in our system.

3. **Domain identifier domID** identifies a domain as an entity registered in an IdP. It follows format `idpID/domIDPart`.

4. **Device identifier devID** identifies a device that belongs to domain `domID`, which has identities registered in IdP `idpID`. `devID` follows format `idpID/-domIDPart/devIDPart`. It is a user-independent identifier that merely defines hardware equipment that is related to a specific domain.

5. **User identifier userID** identifies a user identity that belongs to domain `domID` and follows format `idpID/domID//userID`. Unlike the full format identifier, user identifier lacks the `devID` part, because it is not relevant in this structure.

Different identifiers, i.e. combinations of the mentioned identifier sub-components, are managed with user accounts. In order to start using the proposed IdM system, with a certain device, a user has to log into the device with one of his/her accounts, as described in Section 5.5.1. Afterwards, the device obtains a specific `udevID` combination to be presented to other communication parties, e.g. services.

If a different person in the *same* domain starts using the *same* device, the new active `udevID` differs from the one used by the previous user only by its `userIDPart` sub-component. However, if a user (the same or another) logs into an account that specifies a different domain, the mentioned sub-components may be different, depending on configuration.

The presented identification system is designed for use by devices in automatic and dynamic communications. However, in certain situations a user may be requested to specify his/her user identity or an exact device that is going to be used. For example, during user login in a device, he/she needs to provide user identifier, specified in `userID` format, as described above. However, URL-based identification in user level may seem too complex for regular users. Remembering different sub-components of an identifier, typing them in a correct sequence and remembering the differences of identifiers, e.g. `udevID`, `devID` and `userID`, is error prone and may lead to rejection of the technology.

To improve user experience, we introduce certain simplifications. First of all, it is worth noting that during login procedure, the user does not use `udevID` to specify user identity. `udevID` is composed after authentication, based on selected `devID` and `userID` and is processed only by lower-level components, e.g. DS and SS. Therefore, its complexity does not influence user's experience. Second, `devID`s are bound to certain domains and installed before their utilisation. In addition to that, the system may restrict that a particular device may contain at most one `devID` issued by a certain domain. In this case, the `userID` can be simplified to *userIDPart@domID* format, which follows the structure of an email address that is familiar to the majority of Internet users. This identifier is used locally at user's level and is transformed to complete `userID` and back by DS to send in the network and vice versa. We can reduce user's effort even more by creating a rule that `domIDPart` is globally unique, regardless of IdP, with identity claimed using certificates, e.g. X.509. In this case, `userID` may be simplified to *userIDPart@domIDPart*, i.e. excluding `idpID`.

Additionally, a higher-layer software may be used to control user's local account and improve user experience. For example, such software may prepend `userIDPart` typed in by the user with one of `devID`s associated with user's account in the device.

5.2.3 Identity Information

The identification scheme discussed above allows uniquely identifying IdP, domain, device and a user. A change of at least one of the components in the identifier results in an identifier that potentially refers to a different identity with different properties. This identity information has to be available for access by other entities before they establish connections. However, as explained above, multiple combinations of components lead to a rather large set of identities in each device, and keeping the identity information in a device itself might be impossible or complicated due to memory limits of a device. Furthermore, polling devices in order to decide if certain application-based communication can be performed drains its already limited energy resources and is especially complicated in a system involving billions of devices.

The IdM system proposed in this thesis solves the mentioned problem by storing identity information in an IdP and enabling identity's global addressing with an identifier that follows a URL format, as specified in Section 5.2.2. Such an identification mechanism enables various parties to access identity data based on the identifier, which might be directly requested from a communicating entity or found in an IdP by performing discovery. For example, identity information located using `devID` or `udevID` contains information about an entity's PoCs, physical location, peers that are communicating with the entity and other information. Just like information in the World-Wide Web, identity data in our system is composed by linking data from different entities, e.g. domain, user and one or more devices.

Although entity identification is based on URL structure, it does not strictly define the application-level protocol that is used, e.g. HTTP. This is because communicating parties may have different capabilities for parsing data and even for obtaining it. Therefore, a given IdP may support several protocols for providing data, including Hypertext Markup Language (HTML) document, a text file obtained using File Transfer Protocol (FTP) or other. Furthermore, the data may be provided in XML or JavaScript Object Notation (JSON) format. However, specific information stored in an IdP as well as its storage and delivery formats exceed the scope of this thesis.

5.3 System's Functionality

This section discusses the proposed IdM system's functions and involved parties. The system's functions are specified in a use case diagram presented in Figure 5.3. As the diagram shows, the system involves three types of actors, i.e. roles, and three distinct subsystems responsible for performing certain use cases.

The subsystems provide specific sets of functionality and are utilised by different actors. A `User` utilises the DS installed in his/her device to log into a certain user identity and access services available directly to the user or indirectly, through lower-level utilities provided for the device, based on this user identity. The second subsystem, IdPS, stores identity data related to participating entities, i.e. domains, users and devices, thus enabling easy discovery and communication of devices, authentication of users, devices and services as well as management of device access to these services. Identity data stored in IdPS is managed by one or more `Identity Managers`. It is important to note that `Identity Manager` is not the same as an identity provider. The latter provides the infrastructure for managing identities,

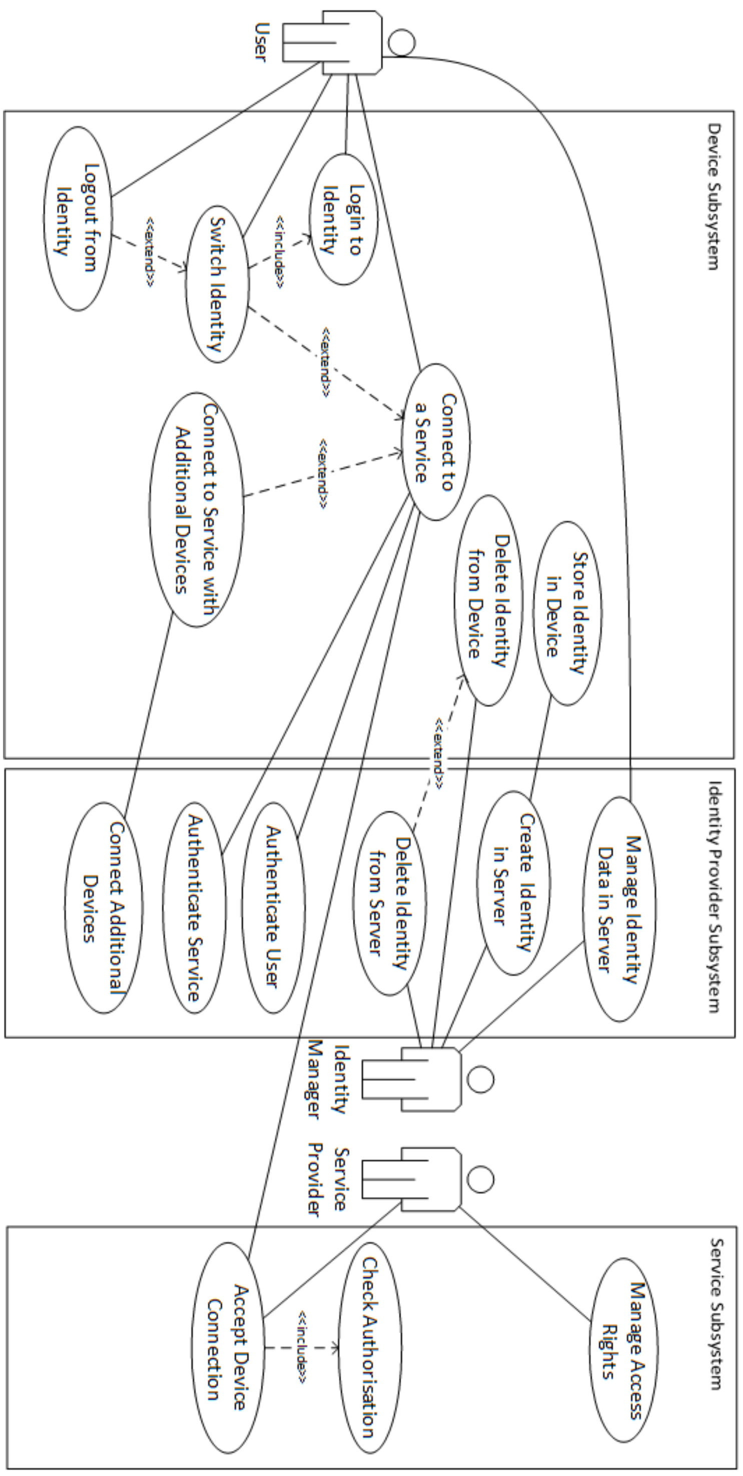

Figure 5.3: System's use case diagram

whereas the former acts as an administrator related to a certain domain and uses the infrastructure. Finally, SS, which is a part of a service compatible with our system, manages access rights of identities and performs authorisation. Service access rights depend on the identity information it obtains from an IdPS. Their management may be based on additional access control mechanisms, such as those discussed in Section 2.1.4. Access rules managed by both `Identity Manager` and `Service Provider` enable controlling access to services from two points, i.e. IdPS and SS. This aspect is discussed more thoroughly below.

Before analysing use cases in detail, it is needed to explain several aspects of the diagram. First, although the diagram involves three distinct actors, a certain party may be responsible for more than one of these actor roles. For example, a university may provide and manage both the IdPS and services that utilise this IdM system. Similarly, a user himself may establish an IdPS and store as well as manage his/her identities privately. This information may be provided in requests to access certain services that trust the provider, e.g. a friend's home network. Furthermore, although the system includes multiple use cases, some of them are not specifically defined in this document, because they can be implemented in many different ways that are platform- or application-dependent. Nevertheless, these use cases are included in our vision of the IdM system's capabilities and responsibilities. Although we provide implementation examples for certain use cases, the exact implementation details are beyond the scope of this document and need further definition.

The remainder of this section discusses each of the use cases in more detail. We start with use cases in DS.

Use case "Login to Identity" enables a `User` to set an active identity in his/her device. Based on the active identity, the device presents a certain identifier to services for authentication and afterwards service access. This thesis does not specifically define how a `User` provides credentials for login process in a device. In the most typical case, the `User` knows his/her user name and password combinations for each of the identities he/she uses and inputs a specific combination during login. Alternatively, biometric, behavioural or some other password-related technology may be used for user recognition, although a user would still need to specify the `userID` for selection of a specific identity. Based on user-specified login information, DS sets an active identity identified by a certain `udevID`.

When a `User` does not wish to utilise the identity that is currently active in a device, he/she performs the action specified by use case "Logout from Identity", which is typically performed by pressing a button in device's menu, executed automatically by the device after an inactivity period or using another technique. If a `User` logs out of an identity in a device, a corresponding DS unit has no active identity set and thus cannot operate with other system's components until a certain identity is set again by performing the login procedure.

Another use case performing actions with active identities is "Switch Identity". This use case is performed when there is a need to change identity to a suitable or desirable one, typically when SS requires an identity issued and managed by a specific IdPS. Similarly to virtual identities specified in [HZW11], in our system a `User` may have multiple identities that represent different, potentially overlapping attribute sets. Thus, when switching into a specific identity in a device, a `User` has to perform the full login procedure as specified for use case "Login to Identity", including provision of credentials or certain secret information. Optionally, if there is another identity currently set as active, a logout procedure is performed before login procedure is allowed.

One of the most important use cases of the system is "Connect to a Service". This use case enables manual connection to a service with a user's device. However, this use case may be extended with additional steps defined by use case "Connect to Service with Additional Devices". This use case enables SDSO functionality and allows a `User` to connect other devices he/she is currently using to the same service using the same identity but without the need to type the full credentials in each of the devices. Actions that enable this functionality on the side of IdPS are defined in use case "Connect Additional Devices".

Functionality of the mentioned connection to a service procedure is based on several key-aspects. First, it ensures service access provision based on the presented `User`'s identity, i.e. access is granted to the `User`, regardless the device he/she uses, rather than a secret that is related to the service itself, e.g. a pass-phrase used in WiFi with Pre-Shared Key (PSK) security mode. Furthermore, devices are able to use this authentication scheme to access both local and remote services. For example, a device may access a traditional WiFi network that uses EAP-TLS authentication technique, or a remote service, e.g. a synchronisation service or storage of measurements of environmental conditions. Finally, although based on identity related to the `User`, authentication is provided both to the user and to his/her one or multiple devices. It means that a device may still perform its own independent functions that are not directly related to `User`, but utilise his/her identity to obtain access to establish connections to services or other devices. Connection-related use cases are thoroughly analysed later in this chapter.

Use cases related to identity creation and modification are performed by an `Identity Manager` and involve actions in DS and IdPS. User, device and domain identities are created by performing "Create Identity in Server". This use case also includes input of identity information that is afterwards stored in IdPS. For user identities, identity data also includes user's credentials needed for logging into one or more devices. However, to actually log into a user identity in a device, the latter has to be configured to support the identity. Thus, some additional information has to be installed in the device, including security data for communication with the corresponding IdPS and domain. As explained below, this information defines a device identity in an exact device. The latter information is stored by performing "Store Identity in Device" use case. Actions performed during creation of identities are thoroughly described in Section 5.5.3.

Similarly, a device may be considered not suitable any more for logging in with user identities from a certain domain. In this case, a device identity related to the corresponding domain may be deleted by a performing a procedure defined as "Delete Identity From Device". The corresponding device information is also deleted from the server that stores it with a use case "Delete Identity from Server". Similarly, this use case also enables deleting user and domain identities.

In addition to previously defined identity creation and deletion use cases, IdPS contains one more management use case "Manage Identity Data in Server". This operation enables administering data related to user, device or domain entities, to preserve its validity, up to date status and avoid violation of security precautions, e.g. keeping only the minimal data set necessary for intended operations. Depending on IdPS implementation specifics, this use case may be performed either by an `Identity Manager` working as a centralised management authority or the user himself. A `User` might be allowed to perform this use case only from a personal device by logging into it using the corresponding identity first. Alternatively, IdPS might provide a website accessible from anywhere, thus a user could use any device with a web browser to perform management. However, this option still requires the `User` to prove his/her identity by providing login credentials in the web page. Further specifics of the use case exceed the scope of this document and remain for definition during implementation.

The final category of use cases involves IdPS and SS operations related to user/device authorisation and connection establishment. First, "Manage Access Rights" in SS describes all the activities related to specifying rules for making authorisation decisions based on identity data obtained from IdPS. The access rules are implementation-specific and thus beyond the scope of this document. Furthermore, as already mentioned, both IdPS and SS involve operations related to checking user's identity. More specifically, IdPS is responsible for authenticating user's device and the service he/she is trying to access. Therefore, use case "Authenticate Device" checks if identity presented by a device really represents the current user. Authentication is performed based on login credentials provided by the user earlier or during connection to the service. Additionally, "Authenticate Service" checks if the service contacting the IdPS is what it claims to be and it actually is the one that the device is trying to access. Finally, when IdPS confirms validity of a user's identity to the specified service and optionally provides the service with additional identity data, a SS is able to check if the user utilising a particular device is authorised to perform any actions with a service. This procedure is indicated by use case "Check Authorisation", which is a part of broader use case "Accept Device Connection" that defines the sequence of messages required for connection establishment and sent between the service and user's device.

5.4 System's Structure

This section describes the static structure of the proposed IdM system, including its members and their relationships. The description includes a more thorough analysis of system's physical entities presented in Section 5.1. For clarity reasons, some of the information from earlier sections is also repeated in this section. The static system's structure represented in a UML class diagram format is shown in Figure 5.4. This diagram is closely related to previously discussed use cases and identification system.

The system consists of several main components. The DS, SS and IdPS represent the already described components of the IdM system. The actor `User`, introduced in Section 5.3, is also presented in the structure as one of the components involved in the system. This actor differs from other actors, since he/she has or remembers certain information, i.e. credentials required to login to his/her identities on a device. Other actors merely act as entities taking management decisions. They do not play a role in the system's static model and are not included in the class diagram.

As explained in Section 5.1, the system is distributed and federated, thus many-to-many relationships are possible among different subsystems, as it is reflected in the class diagram. The remainder of this section analyses each of the components more thoroughly.

DS represents the system's side related to a user's device. As the diagram shows, each device contains an `IdentityAgent` designed to work as a middle-ware IdM layer and provide authentication-related functionality to underlying OS as well as higher-level subsystems, e.g. third-party applications. DS stores installed device identities, i.e. identifiers along with security tokens for secure connection with corresponding IdPs and domains, as well as tracks the currently active user identifier.

The `Application` class included in DS is not directly provided by our system. The class is included in the diagram only to specify that applications utilising the system are required to be aware of it and developed using provided API. More specifically, applications have to be capable of invoking connection process by utilis-

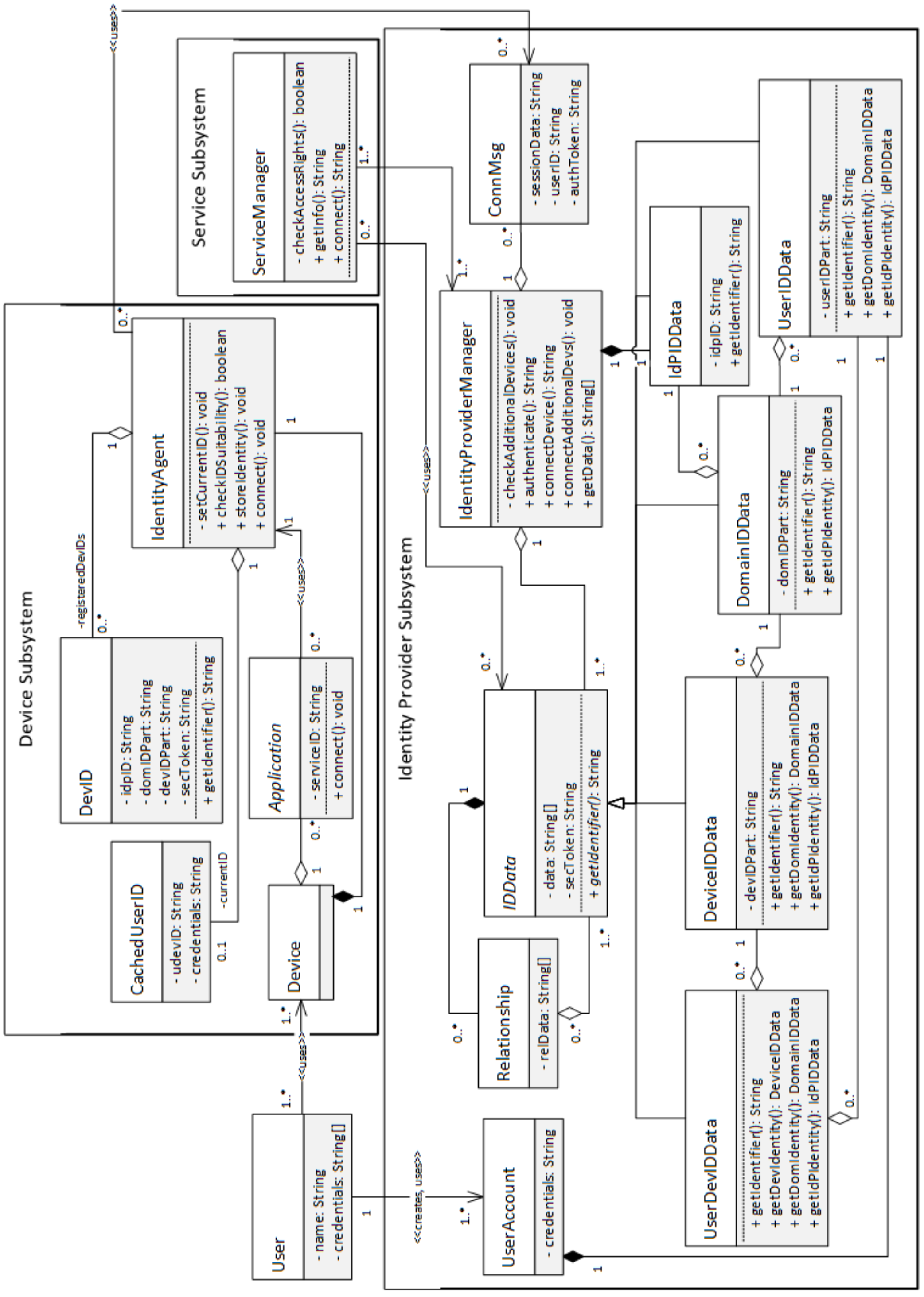

Figure 5.4: System's class diagram

ing capabilities of system's `IdentityAgent` and know identifiers of services they want to access. Other application features are implementation-dependent, thus `Application` class is defined abstract in the diagram.

IdPS acts as an independent subsystem that stores identity data for different users, devices and domains, as well as itself. This information is kept in memory using `IDData` and its children classes. Furthermore, IdPS stores relationships among these identities for more complex identity data usage scenarios. Finally, the subsystem uses class `UserIDData` to store credentials associated with user accounts related to user identities. IdPS is typically stored remotely in the cloud and is globally accessible. However, independent private IdPSs used with private services are also allowed. It is important to note that although multiple IdPSs are allowed, any collaboration among them is beyond the scope of this document.

IdPS collaborates with SSs located together with services by providing them with requested identity data. Only trusted services that have pre-signed contracts with IdP are allowed to obtain data from its IdPS. However, we do not specify this trust mechanism and leave it to be defined during implementation. One of the possibilities to manage the trusted service providers is to store a list of pairs containing a service identifier and a pair-wise secret at the IdP. This secret could then be used to perform a Diffie-Hellman key exchange to establish a secure channel between SS and IdPS.

Furthermore, IdPS communicates with devices of multiple users in order to allow them to log into their identities. Trust among IdP and devices is ensured by using information in devices that is installed as part of device identities, as mentioned in Section 5.3. Specifically, device identities include security information for communication with IdPSs and domains that issued these identities. Specific implementation for this security information is beyond the scope of this document. Implementation options include, but are not limited to, X.509 certificates, secret for Diffie-Hellman key exchange or a pre-shared token. These options determine the specifics of data structures in communicating DS and IdPS.

Finally, SS is a component that is deployed together with the service and performs authorisation of a user's device based on identity data related to a user and his/her device (identified by `udevID`). As mentioned in Section 5.1, the proposed IdM solution targets services accessible by a device both locally, such as WiFi connectivity service, and remotely, e.g. a social network or a cloud data storage service. Thus, depending on a specific service, a certain SS can also be treated as local or remote.

5.5 System's Actions

Previous sections of this chapter described the proposed IdM system's static features. This section continues the description of the system by analysing it from dynamic perspective, i.e. presenting its actions. Among other features, this section discusses an innovative SDSO feature that enables users to obtain access to a service from multiple devices by performing full authentication on only one of the devices. Analysis of the dynamic features starts with discussion of the basic authentication principles in Section 5.5.1. Afterwards, Section 5.5.2 describes collaboration among different system's components during device authentication and connection to a service establishment. Finally, Section 5.5.3 thoroughly describes system's management actions and particularly focuses on creation of identity data in the IdM infrastructure.

5.5.1 User Login and Device Authentication

The proposed system enhances autonomous device-to-device communications based on the information about users utilising them at particular moments. To learn and use information about the current user, the system involves two authentication stages: user login in a device and device authentication to other communicating devices, e.g. services. As already explained in Section 5.2, user and device identifiers are related, thus a particular user's identity that is specified during user login also determines the full identifier `udevID` that a device provides to other peers.

In order for a user to start utilising services connected to the system's infrastructure, he/she has to log into the device by providing the latter with credentials that a device can use when communicating to other parties. The IdM system is designed for on-line communications with connection to an IdP available when needed. Thus, a device authenticates a user on-line, i.e. by communicating with an IdP responsible for the identity a user provides. However, when logging into a device, a user may use a certain `userID` only if the device contains an identifier `devID` that belongs to the same domain. As discussed in Section 5.5.3, device identity installation process is performed by a responsible administrator or the user himself, depending on whether the IdP is public or private.

It is important to emphasize that user login procedure is typically performed when requesting access to a service. A user may then be requested to select an identity and provide login credentials. Typically, a user's account is protected with a user name and password combination, thus a user has to type in his/her credentials in order to authenticate. However, when a user logs into a device, the user identity is set as active and its credentials are cached for later use when connecting to other services that accept identities from the same domain.

Device authentication differs from user login procedure, because in this case it is the device that provides identifier (`udevID`) to other parties and the user is not involved. Furthermore, device authentication may be performed multiple times during a single connection, e.g. when performing security-critical functions. To prove identity, a device needs to provide its own secret, or show knowledge of it, and also include authentication data of a user.

For the device secret, we propose using Identity-Based Cryptography (IBC), which enables secure identity verification without a need for public key management, because an already publicly available identifier, e.g. an email address, may be used. In our case, system's identifiers, e.g. `udevID` and `devID`, may serve as public identifiers. Furthermore, communicating IdPS and DS may both store a shared secret used for authentication and verification of device identity part. The user's credentials may additionally be transferred from device to the IdP, when requested, by protecting them using the same shared secret.

5.5.2 Connection to a Service

The major attention of our system is focused on a device authentication mechanism that enables controlling access to services based on information about devices and their current users. The proposed system involves two related procedures for establishing a connection to a service: general connection and authentication. The latter is part of general connection procedure, but for simplicity and clarity reasons it is distinguished and analysed separately. The further description thoroughly analyses various aspects related to connection to a service and subsystem collaboration during connection establishment. From the perspective of SDSO feature, we initially discuss actions performed by the first device connecting to a service and controlled manually by the user. Afterwards, we discuss automatic connection of additional user's devices.

General Connection Procedure

General connection to a service ensures that during successfully performed procedure, a connecting device finds a service, learns about IdPs supported by the service, authenticates to the service using device and user identity data, and eventually obtains a service connection that it may use to utilise the service. Collaboration of different parties during the general connection procedure is presented as a sequence diagram in Figure 5.5. Due to clarity reasons, we separate description of complex authentication procedure from general connection steps. Thus, this explanation focuses on connection procedure, whereas authentication is analysed separately below. Furthermore, although the overall connection procedure also involves SDSO-related steps, they are discussed individually after description of manual connection and authentication procedures.

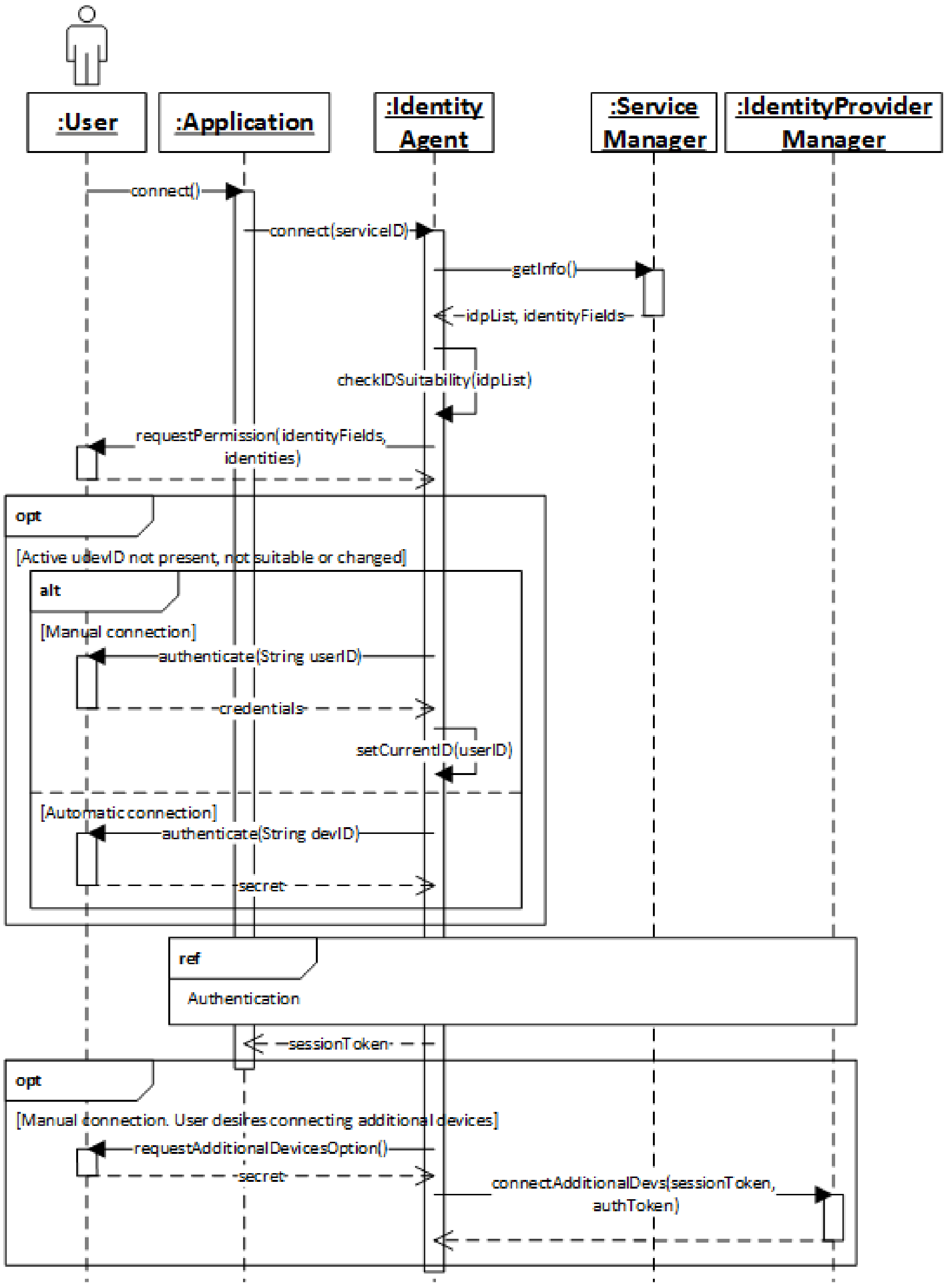
:User
:Application
:Identity Agent
:Service Manager
:IdentityProvider Manager
connect()
connect(serviceID)
getInfo()
idpList, identityFields
checkIDSuitability(idpList)
requestPermission(identityFields, identities)
opt
[Active udevID not present, not suitable or changed]
alt
[Manual connection]
authenticate(String userID)
credentials
setCurrentID(userID)
[Automatic connection]
authenticate(String devID)
secret
ref
Authentication
sessionToken
opt
[Manual connection. User desires connecting additional devices]
requestAdditionalDevicesOption()
secret
connectAdditionalDevs(sessionToken, authToken)

Figure 5.5: Sequence diagram of general connection procedure

Figure 5.5 illustrates a situation when a human user utilises a device and instructs a certain application to connect to a locally or remotely available service that is pre-defined in the application or dynamically discovered. Alternatively, an application may not depend on user's requests for connection and connect to a certain service automatically. The latter option is often suitable for pre-defined, i.e. known by application beforehand, or local services.

Afterwards, the application contacts `IdentityAgent` installed in the same device with a request for connection to a service specified by `serviceID` or alternatively other service contact information. Based on the mentioned service information, `IdentityAgent` finds and communicates with the service's `ServiceManager` to learn about relationships between the service and different IdPs and information about requested identity fields.

When `IdentityAgent` obtains the service-related information, it uses the list of IdPs associated with the service to check whether device's active identity, if any is set, is suitable for the service. Afterwards, `IdentityAgent` informs the user about the service and identity data fields it requests, as well as potentially prompts the user to switch to another identity based on specific requirements.

The optional login procedure shows two alternatives. The first one, i.e. manual connection, is performed when the connecting device is manually controlled by the user and is the first user's device accessing the service. The second optional procedure, automatic connection, enables easily connecting additional user's devices after a user performs manual connection with one of them, as explained in the scenario above. Since automatic connection is explained below, we exclude it from this description and currently focus on manual connection.

After login, the new identity is set as the active identity. However, the credentials are not yet checked by the device or IdP. They are used in authentication procedure performed as the next step of the sequence. In this step, these credentials are sent to an IdP in Extensible Authentication Protocol (EAP)-manner, as explained in the description of authentication procedure below.

When the authentication procedure is complete, `IdentityAgent` sends a `sessionToken` obtained from `IdentityProverManager` during authentication back to the application, which uses the token for further communication with the service without involving the `IdentityAgent`.

Authentication Procedure

The authentication sequence executed during connection to a service is presented in Figure 5.6. As mentioned above, this sequence is not performed independently, but instead is a part of general connection procedure presented in Figure 5.5. Although for simplicity this procedure is called an authentication procedure, it also includes potential authorisation steps to check user's validity to use a service.

In the broad sense, procedure of authentication to a service resembles EAP-based authentication performed when connecting to a wireless network. As shown in Figure 5.6, the procedure begins when an `IdentityAgent` located in an authenticating device contacts a `ServiceManager` in a selected service. However, unlike in EAP, the `IdentityAgent` also informs the service either about its active `udevID` or just `idpID`. Provision of a certain identifier is necessary, because a service may have relationships with multiple IdPs and thus could not independently determine the IdP needed to contact for authentication of a device.

An option to initially provide the service with `idpID` instead of full `udevID` enables preserving security and potentially anonymity, because an IdP may hence hide the real identity of the user and/or device. In such a scenario, the service obtains the complete `udevID` of a connecting device only after a corresponding IdP verifies the service's identity and that the latter matches the service that a device intends to access, as described in later steps of this procedure.

The IdP may also analyse data records stored by an `Identity Manager` to check if the user, device or their combination is allowed to use the particular service. This evaluation is performed together with verification of user's and service's identities. If connection to the service is not allowed, authentication procedure fails.

After receiving the connection request message, `ServiceManager` analyses the included identifier (`udevID` or `idpID`) and contacts a corresponding IdP's `IdentityProviderManager` to authenticate the device. In order to perform authentication, `IdentityProviderManager` requests the device's `IdentityAgent` for certain identity-related information, e.g. credentials. These requests from `IdentityProviderManager` to `IdentityAgent` and responses transmitted in the opposite direction are transferred among the parties directly or indirectly, i.e. forwarded by `ServiceManager`. The exact identity information, credentials, methods for their protection and transfer techniques depend on implementation. One of the examples to perform the procedure is to use:

- EAP-TLS between the device and the IdP,
- EAPOL between the device and the service endpoint,
- Remote Authentication Dial In User Service (RADIUS) between the service and IdP.

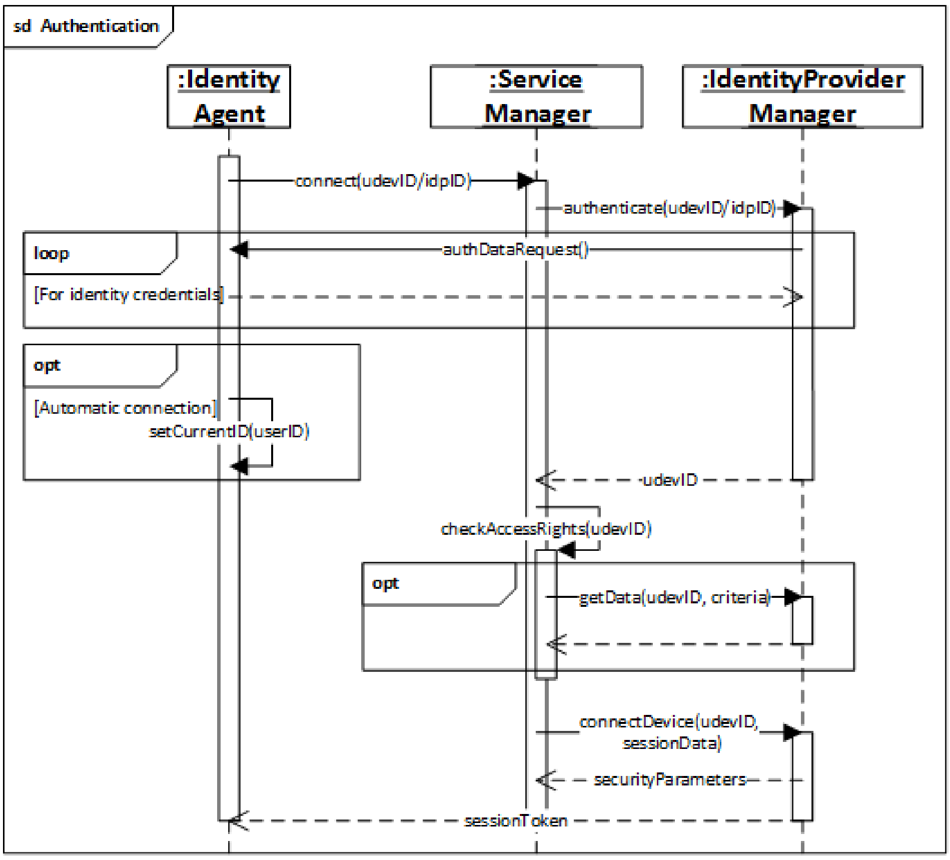

Figure 5.6: Sequence diagram of authentication procedure

In this step, `IdentityProviderManager` may also request `ServiceManager` to prove its identity and check if the service requested by the device matches the service requesting device authentication. After `IdentityProviderManager` validates identity of the device, it returns the IdP authentication result back to `ServiceManager`. This result includes identifier `udevID`.

The diagram also includes an optionally performed fragment that enables setting an active identity in `IdentityAgent`. However, this action is not performed in the first device that is connected to a service. This fragment, as well as other SDSO-related deviations from actions performed by first connecting device, is presented in the description of automatic connection below.

When `ServiceManager` receives a proof of device's identity, it performs its own procedure of device authorisation based on device's `udevID` and additional parameters received from `IdentityProviderManager`. At this point, `ServiceManager` may also request `IdentityProviderManager` for additional information related to identity of `udevID`. The further authorisation mechanism, depending on implementation and specifics of the service, may include RBAC, access matrix or other access control models.

If authorisation succeeds, `ServiceManager` requests `IdentityProviderManager` to connect the authenticated device to the service and provides additional service data that needs to be sent to the device. DS may perform additional actions with this data, if needed.

Afterwards, `IdentityProviderManager` enables communication among the service and device by presenting the necessary data to both parties. In this step, the `ServiceManager` obtains security parameters for secure communication with the device, based on service data presented to the IdPS and `udevID` of the device. Similarly, device obtains a `sessionToken` that includes the necessary information related to device and service authentication, identity data accessed by the service as well as security information for communication with the service. As mentioned above, this token is afterwards returned to the requesting application.

Connection of Additional Devices

The presented IdM system focuses on a situation when a user utilises multiple devices and many or all of them have to be connected to a specified service. Connecting each of the devices by following the manual connection and authentication procedure specified above would be cumbersome and impractical. Thus, the proposed IdM system includes a SDSO feature for seamless automatic connection of additional devices after one of the user's devices is connected manually by performing manual connection procedure. SDSO ensures that after this procedure is complete, other devices that a user picks for utilisation do not require the user to perform the full authentication and obtain service access in a simplified way.

In the description of this procedure, we refer to the first connected device that a user utilises to perform manual connection as initial device. Other devices that obtain access to a service by performing a simplified automatic connection are called additional devices. It is also important to note that, just like the initial device, additional devices are connected only after a user initiates connection, e.g. by launching an application. This feature enables avoiding overhead due to authentication and connection of devices that, although specified, are never used to access a service.

Automatic connection procedure is enabled by additional connection records stored in `ConnMsg` class objects in DS and managed by `IdentityProviderManager` (see Figure 5.4). The procedure involves both general connection and authentication sequences specified in Figure 5.5 and Figure 5.6, respectively. However, the performed steps are different than explained above. The further description analyses the differences and additional actions included in automatic connection procedure.

As shown in the lower part of Figure 5.5, connection procedure involves additional steps necessary for connection of additional devices to the service, after the initial device is successfully authenticated and connected. These steps define that at the end of successful connection to a service, `IdentityAgent` asks the user if he/she desires to use the service with additional devices (displayed as an invocation of `requestAdditionalDevicesOption` method). A user may then agree to use additional devices and merely specify a secret that is used to derive an `authToken`. This token is used later during connection of additional devices, as specified below. The secret may be also obtained in ways other than requesting a textual code from a user. For example, the secret might be obtained from a user's personal UICC card inserted in a device.

Afterwards, `IdentityAgent` sends the derived `authToken` and `sessionToken` to `IdentityProviderManager`, which creates a `ConnMsg` structure that enables simplified connection of other user's devices. `IdentityProviderManager` stores this data for a limited time period.

During connection of an additional device, a user switches to a device different than the one used when performing authentication manually. Furthermore, he/she desires to use the same identity to access the same service. Thus, he/she again performs connection and authentication procedures specified above, but this time certain actions are performed differently. Specifically, during user login step of the connection procedure, the *automatic connection* alternative is performed. In this case, rather than providing full `udevID` and authentication credentials, a user simply specifies a `devID` or `domID`, as discussed above, and a secret. This data is afterwards used in authentication procedure (see Figure 5.6) in the loop of request and response messages sent between `IdentityAgent` and `IdentityProviderManager`. In this case, after receiving an authentication request from a `ServiceProvider`, `IdentityProviderManager` requests `IdentityAgent` of a device to provide a `devID` and performs `checkAdditionalDevices` method to check if it is storing any `ConnMsg` structures that are associated with users allowed to utilise this device. Upon a hit, it sends the device a request for an `authToken`, which is obtained by using an analogous `secret` in the same way as it was obtained in the initial user's device. After successful authentication using a secret, `IdentityProviderManager` sends the device `udevID` and potentially credentials that the device can use for manual access to other devices. Other device and service authentication steps performed in the EAP-like request-response message sequence that are not related to SDSO are also performed during this automatic authentication alternative.

After the request-response sequence is performed, the device possesses the authenticated user's identifier and credentials, thus it may use them to set the current identity. Further authentication and connection steps are the same as in manual connection performed in the initial device.

In the case of additional device connection, the user is not requested if he/she wishes to use the service with additional devices. If there are multiple additional devices, this procedure is performed separately in each of them, when they are picked up by the user to access the service.

This procedure shows that actions of `ServiceManager` do not depend on the specific connection procedure, i.e. manual or automatic. The mentioned aspect is enabled by storing all the sensitive data and critical functionality in centralised and trusted IdPs. This feature simplifies creation and complexity of services and reduces risks related to insecure or distrusted services misusing secret data.

5.5.3 System's Management Operations

The last dynamic aspect addressed in this thesis is establishment of the system, i.e. the process of IdPS deployment, configuration of relationships and creation of domain, device and user identities.

As already mentioned, IdPS may be installed locally by a private user or an enterprise for internal use, i.e. for access by local services. However, the general intention is to provide the system as a public cloud service. This document does not specify exact IdPS implementation or deployment specifics, but the system must communicate using the defined protocol and have a reachable IP address or FQDN that can be used as `idpID`.

When a user or an organisation wishes to utilise a certain IdP for management of device and user identities, it first creates a domain, i.e. registers for IdM service and obtains a `domID`. This operation is performed by a user or a responsible administrator of a company, acting as `Identity Manager` actor presented in Section 5.3. During domain registration, IdP creates an administrator account associated with the created `domID`. This account may represent one of the user identities identified with a distinct `userID` defined in the domain. Depending on implementation, IdP may allow creating additional user accounts with administrator rights in the same domain during or after the procedure of domain registration. An IdP must ensure that domains are isolated, i.e. management operations and data access policies are controlled only by domain-related administrators.

In the proposed system, a domain created in an IdP is also issued an identity. Thus, other parties may request additional domain-related information, which is managed by `Identity Managers`. In the most limited case, a domain is associated with information about its users and devices, which is required for search and discovery of potential communication peers.

After a domain is created, it is possible to create identities for users and devices associated to the domain. As already mentioned, identifiers of such new identities would include `domID` representing the domain, and therefore address of IdP.

Creating a user identity is a simple procedure that is performed by an `Identity Manager` and involves typical account creation steps present in other identity-based systems. However, in our system, user identity is associated with an identifier `userID` representing a hierarchical structure. The `userIDPart` is domain-specific, i.e. `Identity Manager` may specify it explicitly or create a rule, e.g. that an identifier should consist of person's name, surname and birth year. Additional user identity information may be provided and managed by `Identity Manager` or the actual user, by connecting to the same system using his/her credentials. These credentials are created as part of the user identity and used to prove user's presence when logging into a device or into the IdP website for management of his/her data.

The type of user's credentials is implementation-specific and depends on security requirements as well as infrastructure of services that use this data. Similarly, distribution of user's credentials also depends on implementation and type of operations performed by domain. The simplest and most common type of credentials is a username and password combination, which can be sent to a user using e-mail, or distributed as a printed copy by an identity management office. However, the system may use biometric user's information for identification, in which case users provide data before user identities are created or enabled for access.

Creation of device identity involves two steps:

1. creation of identity in the IdP server, which is similar to creation of user identity,
2. deployment of device identifier `devID` along with other related security data in the actual device.

During device identity creation, an `Identity Manager` creates a certain device-related data structure that is stored in the server and potentially used for device discovery as well as establishment of a connection with the latter. `Identity Manager` may manually enter this data in the server. However, a more convenient approach is to have the DS itself store most of the necessary device-related information, e.g. device type, and provide this information to the server when the device is directly connected to it. Direct connection during device identity creation also enables performing both of the mentioned identity creation steps, i.e. creating an identity structure in the server and storing `devID` and related data in the device, to be performed at the same time.

The data set deployed in the device is used by the latter to prove its identity when connecting to services and IdP servers. Effectively, the second step of device identity creation defines permission for users to log into the device using a `userID` that belongs to the same domain as the installed `devID`. Similarly to `userID`, `devID` may be defined manually by an `Identity Manager` or generated by IdP using a specified rule.

Similarly to domain isolation, device identity structures stored in a device must be separated and protected from modification by unintended parties, as well as malicious or careless users. Thus, rights for data modification in a device must be protected with another secret, e.g. a password or an UICC card. Alternatively, similarly to identity creation, device data modification may require connecting a corresponding device to a specific IdP in order to perform and synchronise changes in both IdP and the device.

Chapter 6

Discussion

This chapter discusses and evaluates the proposed IdM system described in Chapter 5. It analyses the system's features, identifies its weaknesses and aspects that need further investigation. In addition to that, it proposes available options to solve certain challenges that were not addressed in the thesis. Along with the thesis outcome, i.e. the proposal of an IdM system, this chapter addresses the thesis work process itself by identifying and discussing findings related to this work. The analysis begins in Section 6.1, which compares our system to other IdM systems presented in Chapter 3. Section 6.2 evaluates the system based on 7 identity laws proposed by Cameron [Cam05]. Section 6.3 analyses some of the system's characteristics related to trust, usability and anonymity. Finally, Section 6.4 discusses the most important issues related to practical implementation and usage issues related to the proposed system.

6.1 System's Comparison to Other Solutions

This section compares the proposed system to other systems discussed in Chapter 3. As in the mentioned chapter, here we address device-based and user-based IdM systems separately. The provided comparison shows that even though differences exist, the systems share certain similarities. This means that although the proposed system is designed to serve a different purpose than other analysed systems, it depends on these technologies and is built using their strong points.

6.1.1 Device-Based Systems

Object Identifier The most similar aspect of OID and our system is identification system. OID uses URIs resolvable by DNS servers, whereas our system utilises resolvable URLs, which effectively comprise a subset of URI set [BLWF+05]. However, our system's identification system is much more complex, defining different members that are involved and enabling identifier distinction by different contexts. Furthermore, our system is targeted towards user equipment, i.e. intelligent devices, whereas OID utilises additional equipment, e.g. RFID tags, and thus enables identification and acquisition of data based on any kind of items, regardless of their intelligence, if any, and communication capabilities. In addition to that, unlike OID, which enables temporary identification of an item and thus promotes reuse of identifiers for different items, our system considers an identifier closely associated with a device, thus identifier reuse is not intended. Finally, whereas OID merely enables identification and data acquisition, our system's functionality is much more complex, involves authentication, sophisticated identity data management, discovery of devices and enables utilisation of enhanced security mechanisms.

Cooltown Similarly to OID, Cooltown uses IR and RFID to identify items, thus unlike our system, it enables providing data and functionality related to identified things regardless of their computing or communication capabilities. On one hand, this aspect of Cooltown enables identification of users, devices and places, which is similar to entities identified in our system. On the other hand, unlike Cooltown, our system defines a structured approach of entity identification.

Object Naming Service Unlike ONS, our system enables both discovery of identity data storage and acquisition of this data. Furthermore, in contrast to device meta-data management technologies, our system is focused on human user and his/her devices regardless of technology used. Finally, our system defines the format of identifiers as well as procedures of identity and identifier creation and entity authentication, which is beyond the scope of ONS.

6.1.2 User-based Systems

Before addressing user-based systems individually, it is worth discussing the common differences between these systems and our proposal. First, unlike other systems, our system involves both user and device identities. Second, it introduces an additional party, identity manager, which is different from identity provider and performs IdM in an IdP's server. Third, whereas other mentioned systems typically involve independent third-party applications to transmit messages, our system includes a native, shared and dedicated component DS, which is installed in user's device. Finally, our system serves a wider purpose, i.e. not only IdM, but also discovery, thus unlike other services, it enables search of devices and users, based on defined policies. Other aspects depend on particular systems, as specified below.

OpenID Compared to OpenID, our system uses a similar approach for accessing services with identities that are provided by multiple different IdPs. However, services involved in our system are not limited to web-services and web-applications that are accessible only by using web browsers. Furthermore, unlike OpenID, our system includes a communication precondition for a relationship established between collaborating SP and IdP. This relationship also defines data used by SP and terms of its storage and usage. Just like OpenID, our system uses resolvable URLs to specify identifiers. OpenID v2 also allows using XRI. However, due to criticism regarding this identification scheme in comparison to URLs [BLW], we omit utilisation of XRIs in our system. Finally, OpenID enables a user to select one of the identities stored in an OIdP. This is possible if the user provides OIdP identifier in the service login form. This is similar to our system's approach, where an IdP may store multiple identities of the same user. However, our system makes the identity provision procedure more automatic and provides the identity of currently logged in user. In this case, `idpID` is provided only to avoid revealing `userID` before the SS is authenticated, as specified in Section 5.5.

SAML Just like SAML, our system enables federated IdM, i.e. requires that parties providing and using identity data initially establish relationships. However, SAML defines several profiles that include HTTP POST messages, thus it is designed to enable SSO to web-services by using web browsers, whereas our system does not impose any limitations on service or client software.

OAuth2 Similarly to OAuth2, our system uses an authorisation mechanism for resource access control. However, OAuth2 defines a wide scope of available resources and operations for their manipulation, whereas in our system, resources are identity data and services. Furthermore, in our system, authorisation decisions are made by two parties, i.e. IdP and service. Just like SAML, OAuth2 is similar to our system because of a requirement to establish relationships among parties before they actually communicate. Furthermore, OAuth2 omits using HTTP POST messages. Thus, just like our system it enables a wide scope of services and clients.

OpenID Connect From the systems defined in Chapter 3, our system is mostly similar to OpenID Connect, because both of them enable user authentication and provide identity management layer for access to identity data as a resource. Since OpenID Connect is built on OAuth2, it shares most of its features and thus similarities with our system, as mentioned above.

6.2 Satisfaction of 7 Laws of Identity

Cameron [Cam05] defined a set of laws for IdM that are necessary for preserving user's privacy when using identities to access web-based services. We evaluate our system based on these laws, as presented further:

1. **User Control and Consent** requires that a user has control over his/her identity data in IdM system and authorises the system to reveal user's information to other parties by explicit consent. This requirement is rather straightforward

and simply defines that a user should know when and what data is sent to a party. Our system addresses this aspect by informing the user about acceptable identities before login, and the specific set of identity data during login, based on relationships between IdP and SP.

2. **Minimal Disclosure for Constrained Use** specifies that only the minimal set of information that enables certain functionality, e.g. a service, should be disclosed. Our system does not directly address this aspect. However, since a SP has to establish a trust relationship with an IdP before it can obtain identity data, these relationships may be used to define what data subsets are needed for certain operations. Furthermore, upon login and/or request to data associated with a user or device, IdP may inform the user about service's requirements for data.

3. **Justifiable Parties** defines that identity information should be disclosed only to parties that serve a certain purpose in the system, defined with relationships. Our system addresses this requirement by performing user, device and service authentication. In this case, identity data and `userID` itself, if the initial connection message sent by DS to SS includes only `idpID`, is revealed to the service only after it is authenticated.

4. **Directed Identity** principle states that a universal IdM system should support both omni-directional and unidirectional identifiers representing an entity for other public and private entities, respectively. Our system fails to meet this requirement, since it treats all entities the same, and uses one type of identifier, i.e. `udevID` to access services. However, since already mentioned, initial provision of `idpID` rather than `udevID` may be treated as provision of a public identifier, until the service is authenticated and thus trusted with the private `udevID`. Furthermore, the proposed system enables having multiple different identities and controls access to these identifiers and identity data. Thus, it effectively provides the same privacy features as demanded by this law.

5. **Pluralism of Operators and Technologies** requires a system to enable collaboration of multiple IdM technologies delivered by different IdPs. Our system fails to meet this requirement, because it does not address collaboration of different IdPs, although this functionality may be introduced in future work. However, from a user's perspective, he/she may use several IdPs, potentially based on different communication technologies. On the other hand, in order to ensure a uniform functionality of several IdPs, specifying the protocol for interaction is insufficient. It is also necessary to define identity fields, e.g. name, address, account number, device communication frequency, etc. Lack of such definitions would complicate IdP and service implementation, as well as establishment of relationships among them. The current work does not address

these issues. However, they may be easily addressed by introducing an identity data specification layer with field types and formats for use by all IdPs, SSs and DSs.

6. **Human Integration** defines that an IdM system should enable such attack-proof means for human-machine interaction that a human user would become a part of the IdM mechanism. Effectively, involving a human user into device-to-device communications is one of the primary goals for our system. The identification system and authentication mechanisms presented in this thesis show that our proposed system meets this requirement.

7. **Consistent Experience Across Contexts** principle states that a system should enable separation of contexts based on different operators and technologies as well as enable consistent user experience across these contexts. Our system entirely meets this requirement and even exceeds it, because besides operators and technologies (we assume communication technologies), it also involves domains, i.e. organisations issuing the identities, into consideration during definition of contexts.

6.3 Utilisation Characteristics

This section addresses additional IdM features that are considered important in related literature.

6.3.1 Trust

As identified by Windley [Win05], storing identity data in IdP server is greatly related to trust of IdP, which is not limited to merely knowing the exact organisation that provides IdM service. In addition to that, the users of the system, including identity managers and the actual subjects described by identities, need to be ensured that the IdP they utilise will not go out of business. Such a situation would result in inability of accessing services and potentially loss of managed data. Furthermore, a user cares if IdP would block his/her account if the credentials are stolen or it is hacked in some way.

Our system does not address the issue of an IdP going out of business. Thus, in order to protect the data and functionality, the system needs additional means for exporting the managed identities. This would enable easily moving the identity hierarchy of a domain to another IdP. However, `devIDs` installed in devices include `domIDs`, which depend on `idpIDs`. Thus, if identities are moved to another IdP, data installed in devices has to be updated. This aspect is not addressed in the current system's proposal and requires further investigation.

On the other hand, our system enables blocking compromised accounts by deleting identities and potentially creating new ones, with the same information, but different identifier. Identity deletion on IdP side is sufficient, because even if the user's device provides non-existent credentials, login procedure is performed on-line and involves a corresponding IdP, thus authentication would fail. However, for usability purposes, e.g. in order to avoid suggesting that a user chooses non-existent identities, inactive identity data should be deleted from a device in a certain way. Since modification of data in a device is protected by administrator password or additional mechanisms, such a procedure would be rather complex. However, just as installation of new identities, deletion of old identity data is expected to be performed rather infrequently. Thus, such a situation should not introduce significant discomfort.

6.3.2 Usability

We consider system's usability from the user's perspective. In order to increase the chances of the system being widely adopted, it needs to be user-friendly. In other words, it should not require much effort from users to learn and afterwards use the system. One of the strategies to reduce the mentioned effort is to make the system's operation similar to other IdM systems.

Our system is focused on a user that has several identities and authenticates using certain credentials. This situation is already familiar to Internet users, since they have multiple different accounts associated with diverse services. In the most common case, our system demands that a user provides a user-name (user identifier) and password combination, which is also common in various web-based systems as well as mobile applications. Furthermore, authorising services to access user information is also a frequent feature, e.g. seen in third party application authorisation in Facebook. Thus, when prompted with service requests to access identity data, users should not be surprised and are expected to understand performed actions and their implications.

However, new features, such as additional device login, may puzzle users without initial preparation. Thus, in order to perform new operations, a proper interactive helper is needed, that would introduce the feature and assist the user when performing certain steps of these features.

6.3.3 Anonymity

In certain cases, a user may want to stay anonymous, i.e. provide only general information that would not enable identification. However, just like other IdM systems discussed in this document, our system is geared towards provision of data that is pseudo- or completely identifying the user and thus enable personalised services. Typically, services that enable anonymous connection do not authenticate users or utilise their identity data. Thus, even if a user is required to provide a pseudonym when he/she accesses a service, this does not constitute an identity and thus is not considered a problem related to IdM.

6.4 Practical Application Issues

6.4.1 Prototype Implementation and Out-of-Scope Parts

An apparent drawback of this work is lack of a system's prototype implementation. Although the system looks promising theoretically, it is essential to implement its prototype and test it with real users. This also means that out-of-scope parts of the systems must be filled with specific design solutions. For many of these parts, we provide recommended or possible implementation options, thus creation of a prototype should not be a problem.

However, multiple out-of-scope parts indicate that such an identity management and communication system greatly depends on other systems and technologies that provide security, communication mechanisms and software for functionality. This means that introduction of such a system may require collaboration with multiple institutions to make adjustments or updates in the associated technologies. However, it also means that this thesis provides an idea of an IdM system based on identified requirements, rather a complete specification. Parts left for decisions during implementation show that this document does not limit implementation possibilities and leave development of specifications for this kind of systems to specification institutions.

6.4.2 Legal Identity Management Issues

Identity data identifies a user and reveals user's personal details that, if stolen, may result in damage of certain type. Thus, it must be protected. As noted by Windley [Win05], IdM systems are often regulated by law that often requires that IdPs define utilised methods for obtaining and storing the data, the purpose of collecting data and the peers that will have access to this data. However, previous technologies consider either public general data stored in a public IdPs, or private, more sensitive data, e.g. bank account number and home address, in ad-hoc IdPs controlled by certain institutions for provision of specific internal services and serve specialised purposes.

However, in this work, we go a step further and propose storing private identity data in the cloud, i.e. in a public IdM system. Although this data is managed by trusted parties, it is stored in a remote location. Furthermore, in order to enable automatic access to this data, it cannot be completely hidden from the IdP. Thus, our system is facing a potential problem associated with trust and legal restrictions that may impede successful spread of the technology. In order to avoid that, additional security mechanisms or contracts by public IdPs may be needed, which this document does not address.

6.4.3 Secure Data Storage in Users' Devices

Another concern emerges from current implementations of sensitive data protection in users' devices.

Currently, Android OS "forgets" user's credentials when the screen of a device is locked and retrieves them again for easy access when the screen is unlocked. The unlock mechanism requires that the owner of a device proves his/her presence by typing in a Personal Identification Number (PIN) code or drawing a pattern that is used to recompute the secret to unlock credentials [Ele12]. Although this security aspect ensures data protection, it limits possibilities of creating a completely automatic system that works with a locked screen. In order to enable desired functionality of access to sensitive data without the need to constantly unlock the screen, Trusted Execution Environment (TEE) technology [Gil14] could be used. However, it still requires changes and adoption by mobile OSs before it can be used.

In addition to that, mobile OSs treat devices as personal devices, thus PIN- or pattern-based screen unlock mechanisms assume there is only one user/owner and thus one correct user-provided secret. There are no possibilities to isolate sets of credentials by creating separate user accounts with different PIN codes for each user. Thus, our system would require even more changes in different mobile OSs.

Chapter 7 Conclusion

This thesis revealed current issues related to development of future communication technologies, particularly lack of orientation towards regular users, and addressed these problems by proposing a sophisticated IdM system that utilises a complex identification system and enables management of identity hierarchies in a private or a could IdM server. Furthermore, the system addresses user-related IdM, which led to a proposal of a SDSO mechanism. This section discusses potential future work that would allow improving the system as well as presents the final remarks of the thesis.

7.1 Future Work

7.1.1 Practical Implementation

In order to evaluate the currently proposed system and be able to improve it, the next step of the remaining work is to implement and test the system in practice. A prototype implementation based on the proposal presented in this thesis should be tested by real human users, who could afterwards provide feedback regarding the effort needed to understand the system's concept, create identities and use identifiers to access services.

Furthermore, current proposal involves multiple out-of-scope aspects, which may require further elaboration and specification. Implementing a practical prototype would allow trying different options, which would in turn give more information about the needs of the system and thus enable improving it further.

7.1.2 Application Programming Interface for Device Subsystem

One of the aspects not elaborated in this document is interaction between third party or OS applications and the DS. Therefore, it is important to define an API specification that would allow developers more quickly learn the technology and use it in development of applications and services.

7.1.3 Legal Aspects

As indicated in the discussion provided in Chapter 6, IdM procedures are subject to legal regulations. Our system complicates the situation even more by introducing universal IdPs that do not fit in the regulations designed for more limited, ad-hoc IdM systems. Thus, envisaged future work includes an analysis of existing regulations in different countries and identification of system's aspects that contradict existing rules and require modification of existing or introduction of new legal rules.

7.1.4 Survey of Involved Actors

Users often express their concern regarding privacy when an IdM system or a service asks them for personal details. Such a reluctance to provide personal information may prevent our system from reaching an adequate number of users, enterprises and services that would enable large-scope benefits of the system. Thus, in future work, it is needed to survey different parties, including regular human users, identity managers and service providers, regarding their expectations about the proposed IdM system and willingness to use it. The survey should address two different situations: when the IdPS is run privately and when it is deployed in a cloud.

7.2 Final Remarks

Scientists and the industry unanimously agree that the number of communication-capable devices used by both industry and regular human users is only going to increase. This will inevitably result in a multiplicity of devices and abundance of communicated data, most of which will be redundant and impose requirements for automatic processing. This will not only require changes in communication techniques and infrastructures, but also rethinking service models, applications and especially user-performed work-flow.

To the best of our knowledge, this thesis is the first work that addresses automatic user-based device-to-device communications and SDSO feature for easy connection of multiple user's devices. With this proposal, we hope to attract scientific community's attention towards the unexplored and rather relevant area of user-related automatic device communications. As a result, we expect that the proposed solution encourages new user-related communication system proposals, specific protocols and their implementations, as well as emergence of applications based on these systems. Such systems would contribute to evolution of communications and the Internet, thus even more improving user welfare by enabling advanced automatic and ubiquitous user-related functionality.

However, it is important to note that the IdM system presented in this thesis serves a different purpose than other currently developed industry-related technologies that enable communications in proprietary static infrastructures. The latter systems address different problems and enable use cases that diverge from objectives of our system. Therefore, it is highly possible that the future of communication technologies will experience coexistence of both industry- and regular user-oriented technologies and applications, thus providing specialised solutions serving distinct needs.

References

[3GP11a] 3GPP Organizational Partners. Service requirements for machine-type communications (stage 1), 3GPP TS 22.368 V11.3.0. Available at http://www.3gpp.org/ftp/Specs/archive/22_series/22.368/22368-b30.zip, September 2011.

[3GP11b] 3GPP Organizational Partners. System improvements for machine-type communications, 3GPP TR 23.888 V1.6.0. Available at http://www.3gpp.org/ftp/Specs/archive/23_series/23.888/23888-160.zip, November 2011.

[AH08] H. Akram and M. Hoffmann. Laws of identity in ambient environments: The hydra approach. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM '08. The Second International Conference on*, pages 367–373, Sept 2008.

[ARE$^+$05] Nidal Aboudagga, Mohamed Tamer Refaei, Mohamed Eltoweissy, Luiz A. DaSilva, and Jean-Jacques Quisquater. Authentication protocols for ad hoc networks: Taxonomy and research issues. In *Proceedings of the 1st ACM International Workshop on Quality of Service &Amp; Security in Wireless and Mobile Networks*, Q2SWinet '05, pages 96–104, New York, NY, USA, 2005. ACM.

[Ash09] Kevin Ashton. That 'internet of things' thing. Available at http://www.rfidjournal.com/articles/view?4986, 1 2009. [Online; accessed 15-May-2014].

[BHOS12] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: a framework for comparative evaluation of Web authentication schemes. Technical Report UCAM-CL-TR-817, University of Cambridge, Computer Laboratory, March 2012.

[BL07] Michelle Boatwright and Xin Luo. What do we know about biometrics authentication? In *Proceedings of the 4th Annual Conference on Information Security Curriculum Development*, InfoSecCD '07, pages 31:1–31:5, New York, NY, USA, 2007. ACM.

[BLW] Tim Berners-Lee and Stuart Williams. TAG recommends against XRI. Available at http://lists.w3.org/Archives/Public/www-tag/2008May/0078.html. [Online; accessed 27-March-2014].

[BLWF⁺05] Tim Berners-Lee, W3C/MIT, R. Fielding, Day Software, R. Masinter, and Adobe Systems. Uniform resource identifier (URI): Generic syntax. Available at http://www.ietf.org/rfc/rfc3986.txt, 01 2005. [Online; accessed 03-June-2014].

[BSSB05] Abhilasha Bhargav-Spantzel, Anna C. Squicciarini, and Elisa Bertino. Establishing and protecting digital identity in federation systems. In *Proceedings of the 2005 Workshop on Digital Identity Management*, DIM '05, pages 11–19, New York, NY, USA, 2005. ACM.

[Cam05] Kim Cameron. The laws of identity. Available at http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf, 5 2005. [Online; accessed 2-June-2014].

[CG11] Stuart Clayman and Alex Galis. Inox: A managed service platform for inter-connected smart objects. In *Proceedings of the Workshop on Internet of Things and Service Platforms*, IoTSP '11, pages 2:1–2:8, New York, NY, USA, 2011. ACM.

[CMT08] Sébastien Canard, Eric Malville, and Jacques Traoré. Identity federation and privacy: one step beyond. In *Proceedings of the 4th ACM workshop on Digital identity management*, DIM '08, pages 25–32, New York, NY, USA, 2008. ACM.

[CN03] Mark D. Corner and Brian D. Noble. Protecting applications with transient authentication. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 57–70, New York, NY, USA, 2003. ACM.

[CNH⁺07] G. Chavira, S.W. Nava, R. Hervas, J. Bravo, and C. Sanchez. Combining rfid and nfc technologies in an ami conference scenario. In *Current Trends in Computer Science, 2007. ENC 2007. Eighth Mexican International Conference on*, pages 165–172, Sept 2007.

[CNH⁺08] G. Chavira, S. W. Nava, R. Hervás, V. Villarreal, J. Bravo, S. Martín, and M. Castro. Services through NFC technology in AmI environment. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, iiWAS '08, pages 666–669, New York, NY, USA, 2008. ACM.

[CY11] Yuan Cao and Lin Yang. Gisl: A generalized identity specification language based on xml schema. In *Proceedings of the 7th ACM Workshop on Digital Identity Management*, DIM '11, pages 3–12, New York, NY, USA, 2011. ACM.

[Den] Zach Dennis. Choosing an SSO strategy: SAML vs OAuth2. Available at http://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/. [Online; accessed 10-March-2014].

[DLvZN⁺13] Alexander De Luca, Emanuel von Zezschwitz, Ngo Dieu Huong Nguyen, Max-Emanuel Maurer, Elisa Rubegni, Marcello Paolo Scipioni, and Marc Langheinrich. Back-of-device authentication on smartphones. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, CHI '13, pages 2389–2398, New York, NY, USA, 2013. ACM.

[Ele12] Nikolay Elenkov. Storing application secrets in Android's credential storage. Available at http://nelenkov.blogspot.no/2012/05/storing-application-secrets-in-androids.html, 6 2012. [Online; accessed 3-June-2014].

[Eur12] European Telecommunications Standards Institute. ETSI TR 102 935 V2.1.1 (2012-09) machine to machine communications (M2M); applicability of M2M architecture to smart grid networks; impact of smart grids on M2M platform. Available at http://www.etsi.org/deliver/etsi_tr/102900_102999/102935/02.01.01_60/tr_102935v020101p.pdf, September 2012.

[Eur13a] European Telecommunications Standards Institute. ETSI TR 102 857 V1.1.1 (2013-08) machine to machine communications (M2M); use cases of M2M applications for connected consumer. Available at http://www.etsi.org/deliver/etsi_tr/102800_102899/102857/01.01.01_60/tr_102857v010101p.pdf, August 2013.

[Eur13b] European Telecommunications Standards Institute. ETSI TR 102 857 V1.1.1 (2013-08) machine to machine communications (M2M); use cases of M2M applications for eHealth. Available at http://www.etsi.org/deliver/etsi_tr/102700_102799/102732/01.01.01_60/tr_102732v010101p.pdf, August 2013.

[Eur13c] European Telecommunications Standards Institute. ETSI TR 102 898 V1.1.1 (2013-04) machine to machine communications (M2M); use cases of automotive applications in M2M capable networks. Available at http://www.etsi.org/deliver/etsi_tr/102800_102899/102898/01.01.01_60/tr_102898v010101p.pdf, April 2013.

[Eur13d] European Telecommunications Standards Institute. Machine-to-machine communications (M2M); functional architecture. Available at http://www.etsi.org/deliver/etsi_ts/102600_102699/102690/02.01.01_60/ts_102690v020101p.pdf, 10 2013.

[Eva11] Dave Evans. White paper: The Internet of Things. How the Next Evolution of the Internet Is Changing Everything, 04 2011.

[GBK+11] V. Galetic, I. Bojic, M. Kusek, G. Jezic, S. Desic, and D. Huljenic. Basic principles of machine-to-machine communication and its impact on telecommunications industry. In *MIPRO, 2011 Proceedings of the 34th International Convention*, pages 380–385, 2011.

[GHL05] Alfonso Gárate, Nati Herrasti, and Antonio López. Genio: An ambient intelligence application in home automation and entertainment environment. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, sOc-EUSAI '05, pages 241–245, New York, NY, USA, 2005. ACM.

[Gil14] Kevin Gillick. Globalplatform made simple guide: Trusted execution environment (TEE) guide. Available at http://www.globalplatform.org/mediaguidetee.asp, 2014. [Online; accessed 03-June-2014].

[GJK$^+$09] Matteo Gaeta, Juergen Jaehnert, Kleopatra Konstanteli, Sergio Miranda, Pierluigi Ritrovato, and Theodora Varvarigou. Federated identity management in mobile dynamic virtual organizations. *Identity in the Information Society*, 2(2):115–136, 2009.

[Hun02] Craig Hunt. *TCP/IP Network Administration (3rd Edition; O'Reilly Networking)*. O'Reilly Media, Inc., 04 2002.

[HZW11] Chunye Hu, Jie Zhang, and Qiaoyan Wen. An identity-based personal location system with protected privacy in iot. In *Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on*, pages 192–195, Oct 2011.

[Int] Internet Engineering Task Force. The OAuth 2.0 Authorization Framework. Available at http://tools.ietf.org/html/rfc6749. [Online; accessed 18-March-2014].

[KBM$^+$02] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic. People, places, things: Web presence for the real world. *Mob. Netw. Appl.*, 7(5):365–376, October 2002.

[KKK$^+$13] Byoungoh Kim, Taehun Kim, Han-Gyu Ko, Dongman Lee, Soon J. Hyun, and In-Young Ko. Personal genie: A distributed framework for spontaneous interaction support with smart objects in a place. In *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '13, pages 97:1–97:10, New York, NY, USA, 2013. ACM.

[KRL13] Treffyn Lynch Koreshoff, Toni Robertson, and Tuck Wah Leong. Internet of things: A review of literature and products. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pages 335–344, New York, NY, USA, 2013. ACM.

[KSB$^+$13] Damjan Katusic, Pavle Skocir, Iva Bojic, Mario Kusek, Gordan Jezic, Sasa Desic, and Darko Huljenic. Universal identification scheme in machine-to-machine systems. In *Telecommunications (ConTEL), 2013 12th International Conference on*, pages 71–78, 2013.

[KTV05] Eija Kaasinen, Timo Tuomisto, and Pasi Välkkynen. Ambient functionality: Use cases. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies*, sOc-EUSAI '05, pages 51–56, New York, NY, USA, 2005. ACM.

[Law04] G. Lawton. Machine-to-machine technology gears up for growth. *Computer*, 37(9):12–15, 2004.

[OAS05a] OASIS. Assertions and protocols for the OASIS security assertion markup language (SAML) v2.0. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, 3 2005. [Online; accessed 17-March-2014].

[OAS05b] OASIS. Bindings for the OASIS security assertion markup language (SAML) v2.0. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf, 3 2005. [Online; accessed 18-March-2014].

[OAS05c] OASIS. Profiles for the OASIS security assertion markup language (SAML) v2.0. Available at http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf, 3 2005. [Online; accessed 18-March-2014].

[OAS08] OASIS. Security assertion markup language (SAML) v2.0 technical overview; committee draft 02. Available at https://www.oasis-open.org/committees/download.php/27819/sstc-saml-tech-overview-2.0-cd-02.pdf, 3 2008. [Online; accessed 17-March-2014].

[one12] oneM2M. Leading ICT standards development organizations launch oneM2M. Press Release, July 2012. Available at http://www.onem2m.org/press/oneM2M%20Launch%20Release.pdf.

[Opea] OpenID Foundation. OpenID Authentication 2.0 - Final. Available at http://openid.net/specs/openid-authentication-2_0.html. [Online; accessed 14-March-2014].

[Opeb] OpenID Foundation. Welcome to OpenID Connect. Available at http://openid.net/connect/. [Online; accessed 01-April-2014].

[Ora10] Oracle. The Liberty ID-FF Convergence (Sun Java System Access Manager 7.1 Federation and SAML Administration Guide): The Liberty ID-FF Convergence. Available at http://docs.oracle.com/cd/E19462-01/819-4674/gdore/index.html, 2010. [Online; accessed 17-March-2014].

[Pan10] R.R. Panko. *Corporate Computer and Network Security.* Prentice Hall PTR, 2010.

[Per12] George Percivall. Connecting islands in the internet of things. In *Proceedings of the 3rd International Conference on Computing for Geospatial Research and Applications*, COM.Geo '12, pages 4:1–4:1, New York, NY, USA, 2012. ACM.

[PHS13] C.H. Potter, G.P. Hancke, and B.J. Silva. Machine-to-machine: Possible applications in industrial networks. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 1321–1326, 2013.

[Piu13] V. Piuri. Biometric technologies for ambient intelligence in the internet of things. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages lxxi–lxxii, Aug 2013.

[RC11] G. Roussos and P. Chartier. Scalable id/locator resolution for the iot. In *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pages 58–66, Oct 2011.

[RP12] D. Rotondi and S. Piccione. Managing access control for things: A capability based approach. In *Proceedings of the 7th International Conference on Body Area Networks*, BodyNets '12, pages 263–268, ICST, Brussels, Belgium, Belgium, 2012. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[SBJ$^+$] Nat Sakimura, John Bradley, Michael B. Jones, Breno de Medeiros, and Chuck Mortimore. OpenID Connect Basic Client Profile 1.0 - draft 28. Available at openid.net/specs/openid-connect-basic-1_0.html. [Online; accessed 01-April-2014].

[SKSS14] Jaeseung Song, Andreas Kunz, Mischa Schmidt, and Piotr Szczytowski. Connecting and managing M2M devices in the future internet. *Mob. Netw. Appl.*, 19(1):4–17, February 2014.

[SNB$^+$] Nat Sakimura, NRI, John Bradley, Ping Identity, , Michael B. Jones, Microsoft, Breno de Medeiros, Google, Chuck Mortimore, and Salesforce. OpenID Connect Core 1.0. Available at http://openid.net/specs/openid-connect-core-1_0.html. [Online; accessed 01-April-2014].

[SP12] M. Sujithra and G. Padmavathi. Next generation biometric security system: An approach for mobile device security. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, CCSEIT '12, pages 377–381, New York, NY, USA, 2012. ACM.

[SS12] Roland Schlöglhofer and Johannes Sametinger. Secure and usable authentication on mobile devices. In *Proceedings of the 10th International Conference on Advances in Mobile Computing &#38; Multimedia*, MoMM '12, pages 257–262, New York, NY, USA, 2012. ACM.

[Win05] Phillip J. Windley. *Digital identity.* O'Reilly, 2005.

[WJ12] Qiang Wei and Zhi Jin. Service discovery for internet of things: A context-awareness perspective. In *Proceedings of the Fourth Asia-Pacific Symposium on Internetware*, Internetware '12, pages 25:1–25:6, New York, NY, USA, 2012. ACM.

[WWR$^+$12] D. Walczak, M. Wrzos, A. Radziuk, B. Lewandowski, and C. Mazurek. Machine-to-machine communication and data processing approach in future internet applications. In *Communication Systems, Networks Digital Signal Processing (CSNDSP), 2012 8th International Symposium on*, pages 1–5, July 2012.