USER-CONTROLLED IDENTITY MANAGEMENT SYSTEMS USING MOBILE DEVICES

MD. SADEK FERDOUS

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
Doctor of Philosophy

SCHOOL OF COMPUTING SCIENCE

COLLEGE OF SCIENCE AND ENGINEERING
UNIVERSITY OF GLASGOW

JULY, 2015

Abstract

Thousands of websites providing an array of diversified online services have been the crucial factor for popularising the Internet around the world during last 15 years. The current model of accessing the majority of those services requires users to register with a Service Provider - an administrative body that offers and provides online services. The registration procedure involves users providing a number of pieces of data about themselves which are then stored at the provider. This data provides a digital image of the user and is commonly known as the *Identity* of the user in that provider. To access different online services, users register at different providers and ultimately end up with a number of scattered identities which become increasingly difficult to manage. It is one of the major problems of the current setting of online services. What is even worse is that users have less control over the data stored in these providers and have no knowledge how their data is treated by providers. The concept of *Identity Management* has been introduced to help users facilitate the management of their identities in a user-friendly, secure and privacy-friendly way and thus, to tackle the stated problems. There exists a number of Identity Management models and systems, unfortunately, none of them has played a pivotal role in tackling the problems effectively and comprehensively.

Simultaneously, we have experienced another trend expanding at a remarkable rate: the consumption and the usage of smart mobile devices. These mobile devices are not only growing in numbers but also in capability and capacity in terms of processing power and memory. Most are equipped with powerful hardware and highly-dynamic mobile operating systems offering touch-sensitive intuitive user-interfaces. In many ways, these mobile devices have become an integrated part of our day-to-day life and accompany us everywhere we go. The capability, portability and ubiquitous presence of such mobile devices lead to the core objective of this research: the investigation of how such mobile devices can be used to overcome the limitations of the current Identity Management Systems as well as to provide innovative online services.

In short, this research investigates the need for a novel Identity Management System and the role the current generation of smart mobile devices can play in realising such a system.

In this research it has been found that there exist different inconsistent notions of many central topics in Identity Management which are mostly defined in textual forms. To tackle this problem, a comprehensive mathematical model of Identity and Identity Management has been developed. The model has been used to analyse several phenomenons of Identity Management and to characterise different Identity Management models.

Next, three popular Identity Management Systems have been compared using a taxonomy of requirements to identify the strength and weakness of each system. One of the major findings is that how different privacy requirements are satisfied in these systems is not standardised and depends on a specific implementation. Many systems even do not satisfy many of those requirements which can drastically affect the privacy of a user.

To tackle the identified problems, the concept of a novel Identity Management System, called *User-controlled Identity Management System*, has been proposed. This system offers better privacy and allows users to exert more control over their data from a central location using a novel type of provider, called *Portable Personal Identity Provider*, hosted inside a smart mobile device of the user. It has been analysed how the proposed system can tackle the stated problems effectively and how it opens up new doors of opportunities for online services.

In addition, it has been investigated how contextual information such as a location can be utilised to provide online services using the proposed provider. One problem in the existing Identity Management Systems is that providers cannot provide any contextual information such as the location of a user. Hosting a provider in a mobile device allows it to access different sensors of the device, retrieve contextual information from them and then to provide such information. A framework has been proposed to harness this capability in order to offer innovative services.

Another major issue of the current Identity Management Systems is the lack of an effective mechanism to combine attributes from multiple providers. To overcome this problem, an architecture has been proposed and it has been discussed how this architecture can be utilised to offer innovative services. Furthermore, it has been analysed how the privacy of a user can be improved using the proposed provider while accessing such services.

Realising these proposals require that several technical barriers are overcome. For each proposal, these barriers have been identified and addressed appropriately along with the respective proof of concept prototype implementation. These prototypes have been utilised to illustrate the applicability of the proposals using different use-cases. Furthermore, different functional, security and privacy requirements suitable for each proposal have been formulated and it has been analysed how the design choices and implementations have satisfied these requirements. Also, no discussion in Identity Management can be complete without analysing the underlying trust assumptions. Therefore, different trust issues have been explored in greater details throughout the thesis.

Acknowledgements

I would like to thank my supervisor Ron Poet for his continuous guidance and support throughout this research. He has given me the freedom and the flexibility to explore different avenues and has always encouraged me in a wide variety of research endeavours.

I would also like to thank my second supervisor Gethin Norman for all the interesting and challenging discussions and for brining a different point of view that helped immensely. Without his help, it would not be possible to shape this research in its current form.

My heartfelt thanks go to everyone in the School of Computing Science for making it a delightful place to work and for all the support I received during my PhD. Special thanks to my colleagues and friends Niaz Morshed Chowdhury and Soumyadeb Chowdhury for many stimulating discussions and for all the crazy ideas we thought we could explore. I am also thankful to Dr. Naima Reza, for proofreading this thesis in a very short period of time.

I would like to offer my sincere gratitude to all my family members, especially my mother Sultana Ferdous, for all the inspiration since my childhood and for helping me throughout my PhD with lots of encouragement and support. Last but not least, I am eternally indebted to my wife Farida Chowdhury, who herself is a PhD student, for enduring my miserable self throughout my PhD and for all the love and care during this long journey. It would certainly not be possible without her.

This research was funded by the Scottish Informatics and Computer Science Alliance (SICSA) for which I am extremely grateful.

To my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction and Motivation

Currently there are millions of websites around the world providing a plethora of different online services (also known as web-based services or simply web services). These services are accessed via the Internet using different protocols. Originally, the protocols for digital communication were designed to exchange information efficiently and reliably. At that budding stage, the identities of communicating parties could be assumed, and there was no need to verify these formally. This led to the omission of an Identity Layer in these protocols for the formal verification of the identity of a user [1]. As the web and web-based services evolved, the verification of a user's identity became crucial, particularly for a Service Provider (SP; the administrative body that offers and provides services) which needs to identify users in order to provide the relevant services only to the authorised users.

To overcome this limitation, the process of authentication was subsequently added which became an integrated part of web services. The authentication process is preceded by the process of Registration which involves users providing data about themselves which is then stored at the respective SP. The data regarding a user portrays the digital image of the user in a provider and is commonly known as the *Identity* of the user in that SP. With the rapid expansion of the Internet from the 1990s, the number of web services, as well as the user-base, was expanding rapidly with more and more identities being created, and as a result their management became challenging, both for SPs and for users. Identity Management (IdM, in short) was introduced by the industry to facilitate online management of user identities. Different groups worked on their own projects resulting in various incompatible IdM solutions. Additionally, several research initiatives in academia were undertaken leading to different prototypes of Identity Management Systems. Many of these systems not only differ in their architectures but also use different protocols and serve different purposes. They are extensively used by organisations to provide online services. Among them, the popular systems are mature and stable in their functionalities and provide services in a secure manner; unfortunately, they fail to comply with a number of different privacy requirements. Moreover, when a novel SP emerges in the landscape of online services, a new identity is issued to the user. The after effect is that the user ends up with a number of identities scattered across multiple SPs. The management of these identities becomes increasingly difficult for any single user as the number of SPs keeps increasing. Furthermore, users have less control over their data which are stored at different providers and lack knowledge regarding how their data is being treated.

Simultaneously, we have experienced another trend expanding at a remarkable rate the consumption and the usage of mobile devices including smartphones and tablets. In 2014, there were nearly 7 billion mobile subscribers around the world which is 95.5% of the world population [2]. In 2013, nearly 1 billion smartphones were sold and for the first time smartphones outsold feature phones (non smartphones) [2]. Also, the sale of other smart mobile devices such as tablets are rising steadily with almost 200 million tablets sold in 2013 [2]. Most of these mobile devices are being used to access the Internet which has contributed to the increase in global mobile data traffic by 69% in 2014 and the forecast was that the global mobile data traffic would increase 10-fold between 2014 to 2019 [3]. Another compelling fact is that mobile devices are not only growing in numbers but also in capability and capacity in terms of processing power and memory. Most are equipped with powerful hardware and highly-dynamic mobile operating systems offering touch-sensitive user-interfaces. In many ways, these mobile devices have become an integrated part of our day-to-day life and accompany us everywhere. It is fascinating to note the evolution of the mobile device during its short span of life, compared to the Personal Computer (PC). From a mere telecommunication device to an all-round multimedia gadget, mobile devices are rapidly reaching a point where the services of telecommunications, media and information technology are converging seamlessly. The portability, power, intuitive software and ubiquitous presence of mobile devices lead to the core objective of this research: the investigation of how such mobile devices can be used to overcome the limitations of current Identity Management Systems as well as to provide and consume innovative online services.

This thesis investigates the need for a novel Identity Management System that offers better privacy and allows users to have more control over their data from a central location. Additionally, the thesis investigates the role that the current generation of smart mobile devices can play in realising such a system. This thesis argues and shows evidence that the existing Identity Management Systems have limitations regarding different aspects of privacy and do not allow users to have complete control over their own data. Furthermore, the management of different scattered identities will become more and more difficult for any user as she accesses more novel online services. There exists a notion of *User-centric Identity Management* in the literature that has been proposed to tackle these problems [4]. In reality, findings in this thesis suggest that the existing systems based on the concept of User-centric Identity Management do little to improve the situation and the privacy problems are still prevalent. Hence, in this thesis, a novel Identity Management System, called the *User-controlled Identity Management System*, is proposed. The proposed system allows users to have complete control over their own data and to improve the privacy of the user while accessing online services.

To realise such a system, a proposal for a novel Identity Provider (IdP in short and IdPs in plural) which is hosted inside a user's smart mobile device is put forward. This thesis explores the ways the power and novel capabilities of these devices can be harnessed to implement the proposal of a novel Identity Provider and the User-controlled Identity Management System. In addition, the thesis also outlines the additional benefits such a system offers and how such benefits can be utilised to offer innovative online services. No discussion in Identity Management can be complete without analysing the underlying trust assumptions. Therefore, different trust issues have been explored throughout the thesis.

1.2 Thesis Statement

The statement of this thesis is that smart mobile devices can be used to realise the novel User-controlled Identity Management System which will:

1. tackle several outstanding shortcomings, especially regarding privacy, in existing Identity Management Systems,
2. offer users complete control over their data from a central location, and
3. provide additional functionalities that can be utilised to offer innovative online services.

Since each Identity Management System is bound by a set of trust assumptions, realising the User-controlled Identity Management System using mobile devices will also require the formulation of a set of trust assumptions that must be satisfied by the system.

1.3 Research Objectives

The core research goals of this thesis are to identify the shortcomings of existing Identity Management Systems, to rectify these shortcomings by proposing, designing and developing a novel Identity Management System and investigating how such a system can be used to provide innovative online services. To achieve these goals, the following research objectives have been formulated.

Research Objective 1 (RO1): Building a consistent vocabulary for Identity Management. The first objective is to review the existing work and literature on Identity Management in order to compile a consistent vocabulary. Different projects have assumed different interpretations of many similar topics related to Identity Management which introduce inconsistencies. To ensure consistency throughout the thesis, it is essential to build up a vocabulary related to Identity and Identity Management.

Research Objective 2 (RO2): Mathematical Modelling of Identity Management. The second objective is to develop a mathematical model of Identity and Identity Management. As mentioned earlier, there remains an inconsistency in different central topics of Identity Management. The main reason behind this inconsistency is the lack of a mathematical model on the core issues of Identity and Identity Management. To tackle this, a formal mathematical model of Identity and Identity Management has been developed.

Research Objective 3 (RO3): Comparative Analysis of existing Identity Management Systems. The third objective is to compile a set of requirements of an ideal Identity Management System. This set of requirements will be used to provide a comparative analysis of a few popular existing Identity Management Systems and to identify the strength and weakness of each system. Based on the result of the analysis, a system will be chosen for the subsequent research.

Research Objective 4 (RO4): User-controlled Identity Management System. The fourth objective is to propose a User-controlled Identity Management System that will be used to rectify different problems in existing systems.

Research Objective 5 (RO5): Framework for Context-aware Federated Services. The fifth objective is to propose a framework that will combine two popular trends of online services, context-awareness and federated services, using the proposed User-controlled Identity Management System.

Research Objective 6 (RO6): Attribute Aggregation in Federated Services. The sixth objective is to propose an improved architecture for attribute aggregation in federated services and to investigate how it can handle different privacy issues using the proposed Identity Management System.

Among these objectives, the first three (RO1 - RO3) are for developing terminologies related to IdM based on mathematical principles and for identifying shortcomings in the existing systems. The fourth objective (RO4) is to propose a system to tackle the identified shortcomings. The final two objectives (RO5 and RO6) are to investigate how the proposed system can be used to offer innovative online services.

1.4 Contributions

The key contributions of the thesis are the following.

Mathematical Model of Identity and Identity Management. A novel mathematical model of different core topics covering a wide range of topics related to Identity Management has been developed. This is the first model of this kind where, at first, each atomic related issue is treated separately to highlight the inter-relationship between each issue and then all these issues are combined to build up the whole model of Identity. The proposed mathematical model has been used to tackle the issue of inconsistency in different central topics of Identity Management. In addition, the applicability of the model has been illustrated by utilising it to characterise the behaviour of an Identity Management System and to characterise three popular IdM models mathematically. Furthermore, the model has been used to analyse three issues related to Identity Management: the effect of multiple identities, data ownership and Identity Theft. The model has been useful to analyse crucial problems in existing online services.

Dynamic Identity Federations. A mechanism for managing a dynamic federation in a fully automatic fashion using SAML (Security Assertion Markup Language) has been proposed. This approach can be easily adopted by any SAML implementation and requires no modification of SAML. A proof of concept has been implemented to show the applicability of the mechanism through use-cases.

User-controlled Identity Management Systems. The need for a more privacy-friendly Identity Management System is advocated. Then, a proposal of a User-controlled Identity Management System is put forward to allow users to have better control over their data. To realise such a system, a novel Identity Provider called Portable Personal Identity Provider (PPIdP) has been developed and a proof of concept using the mechanism of dynamic federations has been implemented. It has been analysed how the PPIdP can be used to rectify several problems of existing Identity Management Systems.

Context-aware Federated Services. To demonstrate how a PPIdP can be used to provide context-aware federated services, a framework for accessing such services has been designed and developed. Both context-aware and federated services have been popular trends in online services in recent years. The proposed framework is the first attempt to combine these two trends that shows how the additional capabilities of a PPIdP can be leveraged to provide innovative context-aware federated services. The implemented proof of concept shows the applicability of this approach through different use-cases.

A Hybrid Architecture for Attribute Aggregation. Finally, a hybrid architecture for aggregating attributes from heterogeneous identity providers is presented. The existing mechanisms of attribute aggregation are complex and require preliminary steps which are not intuitive. The proposed approach improves upon the existing work; in particular, it does not require any preliminary steps or complex user interactions. The implemented proof of concept shows the applicability of this approach through different use-cases. Also, it has been examined how the privacy of a user during attribute aggregation can be improved by integrating the PPIdP with the proposed architecture.

1.5 Publications

The research in this thesis has led to the following publications:

1. Md. Sadek Ferdous, Gethin Norma, Audun Jøsang and Ron Poet. "Mathematical Modelling of Trust Issues in Federated Identity Management". In the proceedings of the 9th IFIP WG 11.11 International Conference on Trust Management (IFIPTM-15), Hamburg, Germany, 26-29 May, 2015.

2. Md. Sadek Ferdous and Ron Poet. "Managing Dynamic Identity Federations using Security Assertion Markup Language (SAML)". Journal of Theoretical and Applied Electronic Commerce Research (JTAER). 10(2):53Ű76. 2015.

3. Md. Sadek Ferdous and Ron Poet. "CAFS: A Framework for Context-Aware Federated Services". In the proceedings of the 13th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-14), Beijing, China, 24-26 September, 2014.

4. Md. Sadek Ferdous, Gethin Norman and Ron Poet. "Mathematical Modelling of Identity, Identity Management and Other Related Topics". In the proceedings of the 7th International Conference on Security of Information and Networks (SIN-14), Glasgow, UK, 9-11 September, 2014.

5. Md. Sadek Ferdous and Ron Poet. "A Hybrid Model of Attribute Aggregation using Security Assertion Markup Language (SAML)". Submitted to the Journal of Computer Security (JCS), 2014.

6. Md Sadek Ferdous and Ron Poet. "Analysing Attribute Aggregation Models in Federated Identity Management". In the proceedings of the 6th International Conference on Security of Information and Networks (SIN-13). Aksaray, Turkey, 26-28 November, 2013.

7. Md Sadek Ferdous and Ron Poet. "Portable Personal Identity Provider in Mobile Phones". In the proceedings of the 12th IEEE International Conference on Trust,

Security and Privacy in Computing and Communications (IEEE TrustCom-13). Melbourne, Australia, 16-18 July, 2013.

8. Md Sadek Ferdous and Ron Poet. "Dynamic Federation using Security Assertion Markup Language (SAML)". In the proceedings of the 3rd IFIP WG 11.6 Working Conference on Policies & Research in Identity Management (IFIP IDMAN 2013). London, UK, 8-9 April, 2013.

9. Md Sadek Ferdous and Ron Poet. "A comparative analysis of Identity Management Systems". In the Security Track of the IEEE International Conference on High Performance Computing and Simulation (HPCS). Madrid, Spain, 2-6 July, 2012.

1.6 Thesis Outline

The rest of this thesis is structured as follows:

Chapter 2 provides a short introduction to three popular Identity Management Systems and reviews related work on different topics of Identity Management that have been chosen for further exploration in this research.

Chapter 3 builds up the consistent set of terminologies related to Identity Management and presents the mathematical model of Identity Management. The model is then used to characterise several aspects of Identity Management Systems as well as to characterise different Identity Management models. In addition, the model is used to analyse four application scenarios showing how it can be applied to examine a few problems in existing online services. This chapter addresses RO1 and RO2.

Chapter 4 presents a comparative analysis of three popular Identity Management Systems where each system is compared side-by-side using a set of requirements. For this, a taxonomy of requirements for an ideal Identity Management System has been formulated. This chapter addresses RO3.

Chapter 5 presents the proposal for creating federations in a dynamic fashion. The existing mechanism for creating a federation makes it difficult to manage and is not scalable. This problem has been tackled with the given proposal. In addition, the implementation based on the proposal has been extensively used in the subsequent chapters to realise different proof of concepts.

Chapter 6 introduces the core proposal of a User-controlled Identity Management System using the Portable Personal Identity Provider and explains how this can be used to tackle some of the stated problems in existing systems. In addition, this chapter explores different use-cases using the PPIdP, analyses different security, privacy and trust issues involved, the advantages the developed system offers and the current limitations it has. In doing so, this chapter addresses RO4.

Chapter 7 discusses the framework for context-aware federated services. This chapter illustrates how the PPIdP can be leveraged to build up the framework that combines these two trends, Context-aware and federated online services, for offering novel use-cases of online service access. Thus, this chapter addresses RO5.

Chapter 8 presents the hybrid architecture for aggregating attributes from multiple heterogeneous identity providers, examines different issues and requirements while designing and developing such an architecture and discusses the implemented proof of concept using several use-cases. Hence, this chapter addresses RO6.

Chapter 9 concludes the thesis by summarising the conducted research. It revisits the novel contributions of the research and investigates how the research objectives have been achieved. The chapter also discusses the limitations of the work and proposes a number of directions for future research.

Chapter 2

Background

2.1 Introduction

The need for managing user identities was first realised by service providers as the number of their user-bases increased. That need motivated the industry to design and develop different systems to manage the identities of users. Such a system is commonly known as an Identity Management System (IMS). The majority of research on Identity Management is heavily reliant on these systems. Hence, this chapter starts with a short introduction to the three most popular IMSs.

Next, the existing work on different aspects of Identity Management that have been selected for this research are reviewed. The motivations behind each of these aspects have been briefly discussed in Chapter 1. However, the motivation for choosing these aspects will become clearer in subsequent chapters.

2.2 Popular Identity Management Systems

Different Identity Management Systems are based on different IdM models and utilise different protocols. There are three classes of entities involved in each Identity Management System:

1. *Identity Providers (IdPs)* - entities that are responsible for managing digital identities of users and providing identity related services to Service Providers,

2. *Service Providers (SPs)* - entities that are responsible for providing online services to the users, and

3. *Users (Clients)* - entities that receive services from SPs.

Entities from these classes interact with each other using a respective protocol so that online services can be offered and ultimately provisioned. Currently, several IMSs exist. In this section, a short introduction to the three most popular Identity Management Systems is presented: SAML-based Systems [5], OpenID [6] and OAuth [7].

2.2.1 Security Assertion Markup Language (SAML)-based IMS

SAML-based IMSs are based on the Federated User Identity model [4] and utilise the Security Assertion Markup Language (SAML) [5]. In this system, involved parties need to create a virtual association, known as a federation, before they can interact with each other. SAML is an XML (EXtensible Markup Language)-based standard for exchanging authentication and authorisation information between different autonomous application domains [5]. It is based on the request/response protocol in which one party (an SP) requests particular identity information about a user and the other party (an IdP) responds with the information. The whole language comprises of four key concepts: assertions, bindings, profiles and protocols. A SAML assertion is the declaration about a user by an IdP for an SP. In its most general form, an assertion states the following [5, 8]: "*The assertion A issued at time T by issuer (IdP) I about the user U holds as long as conditions C are valid.*"

An assertion can contain three different types of statements:

1. Authentication statements - which are used to state that the user has been authenticated by the asserted IdP at the specified time,
2. Attribute statements - which are used to specify the attributes of the user and
3. Authorisation statements - which are used to assert that the user is permitted a certain action on specified resources under certain conditions.

The SAML protocol is used to define the rules on how to pack the above SAML statements inside the request/response packet and to process them on receipt. The SAML binding is used to map the SAML protocol into a communication protocol such as HTTP (Hypertext Transfer Protocol) or SOAP (Simple Object Access Protocol) by which the SAML elements are transported. Simply, bindings define the mechanisms by which SAML elements are placed inside any HTTP or SOAP packet. A SAML profile combines all of the above mentioned SAML elements and describes how they can be used to implement a use-case. The most important use-case is the Web Browser SSO (Single Sign On) profile which will be discussed later. The SSO facility allows a user to authenticate once and then to access multiple services without further authentications.

A SAML protocol flow based on the Web Browser SSO Profile where both the IdP and the SP are part of a federation and use the HTTP Post binding is collected from [8], illustrated in Figure 2.1 and discussed below:

1. A user wants to access a service provided by an SP and hence submits a service access request to the SP.

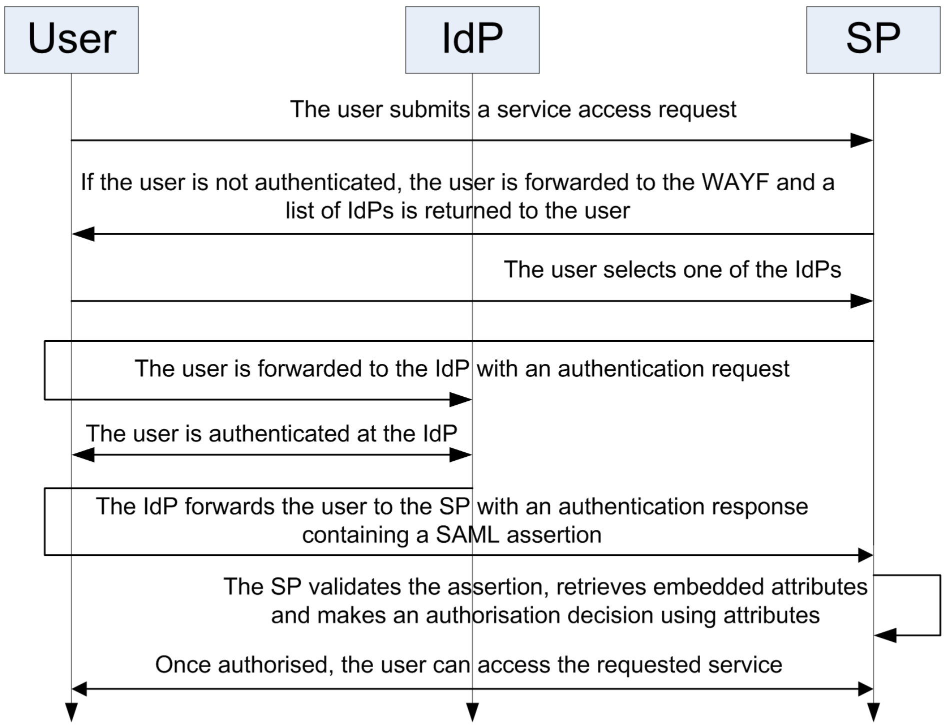

User
IdP
SP
The user submits a service access request
If the user is not authenticated, the user is forwarded to the WAYF and a list of IdPs is returned to the user
The user selects one of the IdPs
The user is forwarded to the IdP with an authentication request
The user is authenticated at the IdP
The IdP forwards the user to the SP with an authentication response containing a SAML assertion
The SP validates the assertion, retrieves embedded attributes and makes an authorisation decision using attributes
Once authorised, the user can access the requested service

Figure 2.1: Protocol flow in SAML.

2. The SAML interface in the SP captures the request and checks if the user is already authenticated (or authorised) at the SP. If yes, the user is allowed to access the service. If not, the following steps take place.

3. The user is redirected to the Discovery Service, also known as the Where Are You From (WAYF) Service, where a pre-configured list of trusted IdPs is shown to the user. The user selects one of the IdPs and an authentication request is sent to the IdP.

4. The IdP captures the request and checks if the user is already authenticated. If so, step 5 is ignored.

5. The authentication of the user takes place. There are many authentication mechanisms (such as username/password, Kerberos, X.509 certificates, etc.) supported by the SAML which depends on the respective IdP.

6. Once authenticated, a response containing a SAML assertion with user attributes is sent back to the SP.

7. Upon receiving the response, the assertion is extracted and validated. If valid, the user is considered authenticated. Next, user attributes are extracted from the assertion and the user is re-directed to the requested service.

8. The SP checks if the user is authorised to access the service using the extracted attributes. If so, the user is allowed to access it. If not, the request of the user is denied and an error message is displayed to the user.

To enable the protocol flow discussed above, a notion of trust needs to be established between an IdP and an SP inside a federation. A SAML SP needs to trust that the IdP will authenticate a user using appropriate security mechanisms and release attributes to the SP as per the contractual agreement [9]. Similarly, the IdP has to trust that the SP will not abuse the released attributes and use them only for the stated purpose as per the agreement. This notion of trust between the IdP and SP in SAML is established by exchanging the respective metadata of the IdP and SP and then storing such metadata at the appropriate repositories. This enables each party to build up the so-called Trust Anchor List (TAL). This exchange takes place in an offline fashion after a technical contract between the IdP and SP is signed. Moreover, this has to be done before any interaction takes place between the said IdP and SP. Once the exchange of metadata is complete, the IdP and SP are said to be part of the same federation. Metadata is an XML file in a specified format that contains:

- an entity descriptor (an identifier for each party),
- service endpoints (the locations of the appropriate endpoints for an IdP and an SP where the request will be sent to and where the response will be consumed respectively),
- a certificate,
- an expiration time, and
- contact information.

Metadata serves three purposes as explained below.

- Firstly, it allows each entity to discover the required service endpoint of another entity for sending SAML requests/responses.
- Secondly, the embedded certificate can be used by an SP to verify the authenticity of any SAML Assertion.

- Thirdly, metadata behaves like an anchor of trust for each party. During the discovery service at an SP, the list of IdPs returned to the user only contains those IdPs whose metadata can be found in its repositories (in other words in the TAL) and only those IdPs are considered trusted. An SP will allow a user to select only those IdPs which are trusted. Similarly, an IdP will respond only to those requests that are initiated from an SP whose metadata can be found in its TAL.

The life-cycle of a SAML-based federation consists of four major steps that define how a federation is created, maintained and updated. The steps are: federation, federation provisioning, maintenance and revocation. These steps are dependent on how the TAL is built and maintained. For example, when the metadata of two entities are inserted into their respective TAL, it indicates that both entities are now federated with each other, thus symbolising the federation step. Identifying and discovering the IdP from the TAL of the SP exemplifies the federation provisioning stage whereas the maintenance stage is required when the metadata of an entity stored in the TAL of another entity needs to be modified (e.g. due to changes in the location of service endpoints, etc.). Lastly, removing the metadata of two entities from their respective TAL exemplifies the revocation. More about the life-cycle of a federation will be discussed in Chapter 3.

There are a number of different libraries for SAML for building up a SAML-based IMS with Shibboleth [10] and SimpleSAMLphp [11] being the two most widely used libraries developed using Java and PHP respectively.

2.2.2 OpenID

OpenID is a decentralised Identity Management System which provides SSO solutions for web services over the Internet [12]. It aims to assist users by alleviating the need for maintaining and using different identities to access different web services as well as to aid an SP by removing the need to manage its user-bases. It is based on the Open Unfederated Identity Model (discussed in Chapter 3). With a user-base of more than 1 billion users, it is one of the most widely used IMSs and is used by many web service providers including AOL, BBC, Google, IBM, MySpace, Orange, PayPal, Verisign, LiveJournal and Yahoo [12, 13].

Like SAML, OpenID protocol has three different classes of actors (entities): Users, OpenID Providers (also simply known as Providers) and Service Providers, known as Relying Parties (RPs) in OpenID terminology. A user is an entity that wants to access a protected service provided by an RP. An OpenID Provider allows a user to create an identity. The provider is also responsible for authenticating a user who wants to access the protected service from an RP and provides an assertion regarding the user to an RP. An RP is essentially a service provider providing services to a user and consumes an assertion given by a provider to decide

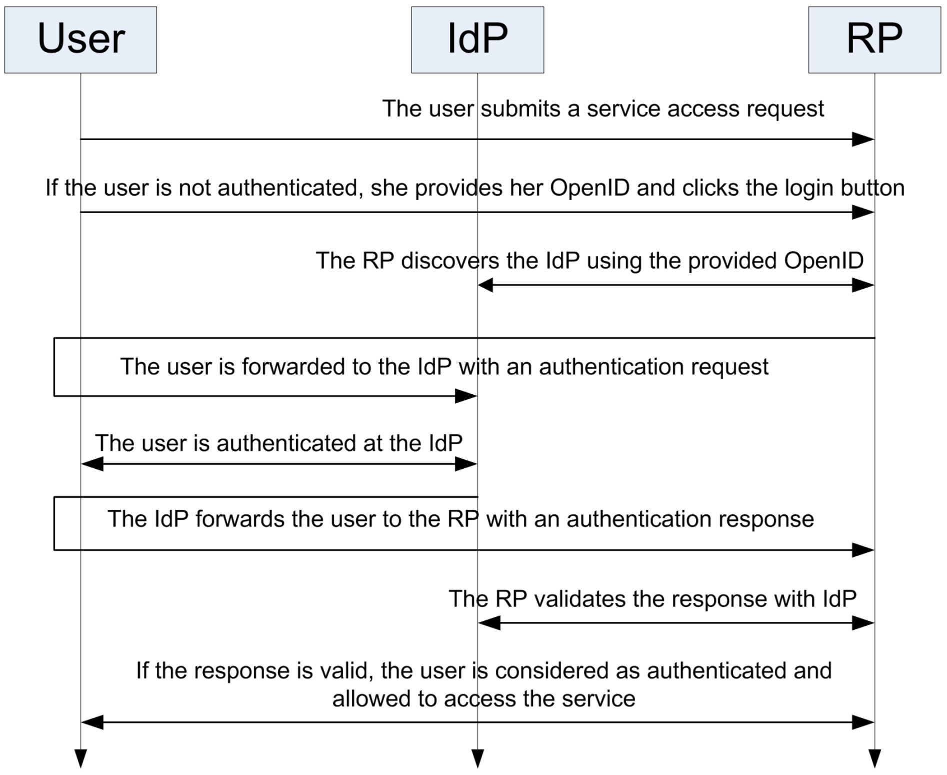

User
IdP
RP
The user submits a service access request
If the user is not authenticated, she provides her OpenID and clicks the login button
The RP discovers the IdP using the provided OpenID
The user is forwarded to the IdP with an authentication request
The user is authenticated at the IdP
The IdP forwards the user to the RP with an authentication response
The RP validates the response with IdP
If the response is valid, the user is considered as authenticated and allowed to access the service

Figure 2.2: Protocol flow in OpenID without exchanging keys.

if a user can access the service. There are a number of libraries for all major web technologies and programming languages such as Apache, C#, Coldfusion, Haskell, Java, JavaScript, Perl, PHP and Python [14].

To understand the protocol flow of the OpenID, it is considered that a user wants to access a service from an OpenID RP. Before engaging into the protocol flow, the first step for the user is to create an OpenID identifier (also known as just OpenID) at an OpenID Provider. An OpenID is essentially either a URL (Uniform Resource Locator) or an XRI (Extensive Resource Identifier) and analogous to an email address in the sense that an OpenID is universally unique. Many large popular web service providers such as Google, Yahoo, LiveJournal, Blogger, flickr, AOL and WordPress can be used as OpenID Providers since they allow their users to use their respective identifiers (usernames) to be used as OpenIDs. There are also many dedicated OpenID providers such as claimID, myOpenID, myid.net and VeriSign. The main point is that any provider can act as an OpenID provider by deploying the required libraries and users can choose any OpenID Provider for accessing a service. The protocol flow in OpenID is illustrated in Figure 2.2 and is discussed below:

1. A user visits an RP to access one of its services.
2. The RP needs to authenticate the user. As it supports OpenID, it has a special button tagged as *Sign in with OpenID*. Once clicked, this will allow the user to enter her OpenID.

3. The user enters her OpenID and clicks the *Login* button.
4. The RP uses that OpenID to discover the OpenID provider.
5. Once the provider is discovered, the RP may initiate an optional Key Exchange step in which the RP and the provider exchange a shared secret key to be used to verify the signature of the RP in a response message (see below).
6. Once the provider is found, the user is redirected to that respective provider using an OpenID authentication request message via HTTP redirect.
7. When the provider receives that authentication request, it needs to authenticate the user if the user is not already authenticated. For authenticating the user, the provider can choose any suitable mechanism.
8. Once the user is successfully authenticated, the provider sends an authentication response to the RP.
9. The RP now needs to validate the response by verifying the signature embedded inside the response. It can do it in two ways. If both parties have agreed to use a pre-arranged key in Step 5, the RP can use that key for verification. Alternatively, the RP needs to send back this response to the provider where the signature will be verified.
10. Then the provider will send back the verification response to the RP.
11. If the verification response is positive, the user will be authenticated at the RP and will be allowed to access the service.
12. As the user is now authenticated at the provider as well, she can use the corresponding OpenID for accessing any OpenID enabled services provided by different RPs without any authentication at the provider. She will just need to provide her OpenID at an RP, the RP will redirect her to the provider, the provider will find that she is already authenticated at its end and so it will just send the authentication response to the RP and the usual flow will take place. In this way, the SSO functionality is achieved in OpenID. However, it is important to remember that if the user chooses to use a different OpenID, she will need to authenticate herself at the respective provider.

2.2.3 OAuth

OAuth is one of the fastest growing community-based specifications that has been designed to provide mechanisms to circumvent some of the limitations related to the access delegation in the traditional model of accessing services [15]. To exemplify these limitations, it is assumed that a user wants to delegate her authorisation right to other users so that they can access some of the first user's resources. For this, the user has to share her identifier (username) and the secret credential (password) with all other users. If any of these users acts maliciously, the identity of the user is fully compromised. Again, sharing such information with another user allows the second user to gain the ultimate authorisation right (such as read, write, update and delete). A malicious user can abuse that right. Finally, the revocation of this right in the traditional model is rather troublesome. Once the user detects a malicious user and wants to revoke the authorisation right of that entity, the user has to change the secret credential which means that the user needs to re-distribute the secret credential again to other honest users to delegate her access rights.

OAuth provides a flexible solution to this problem that allows any user to delegate her access right in a more user-friendly and secure way. The original specification, known as OAuth 1.0, was finalised in April 2010 and is specified in RFC 5849 [16, 17]. However, it subsequently went through a complete modification and evolved to a new version called OAuth 2.0 [7], which is a completely new protocol.

OAuth protocol comprises of four different classes of entities [7].

Resource Owners. Resource owners (or owners in short) own and control the protected resources and are capable of granting (delegating) limited access rights to third parties for accessing protected resources under tight control. They take the role of Users in the traditional model.

Clients. Clients are third party applications that can make requests to access protected resources on a user's behalf. To make such a request, they have to receive an authorisation clearance (known as the authorisation grant in OAuth 2.0) from the corresponding resource owner.

Authorisation Servers. Authorisation servers are responsible for granting access tokens to clients after receiving valid authorisation grants.

Resource Servers. Resource servers host protected resources and are capable of accepting and responding to requests for accessing resources using access tokens. In many settings, resource and authorisation servers may be the same entity.

The number of legs is used to describe the number of entities involved in an OAuth interaction [18]. For example, if all four classes of entities are involved in an interaction, the flow is described as 4-legged. In case, if the authorisation and resource server is one combined entity, the flow can be described as 3-legged. A flow that is more than 4-legged is also possible if there is more than one client and one of the clients has the capability to delegate the authorisation right to other clients.

An abstract protocol flow is illustrated in Figure 2.3 and is discussed below:

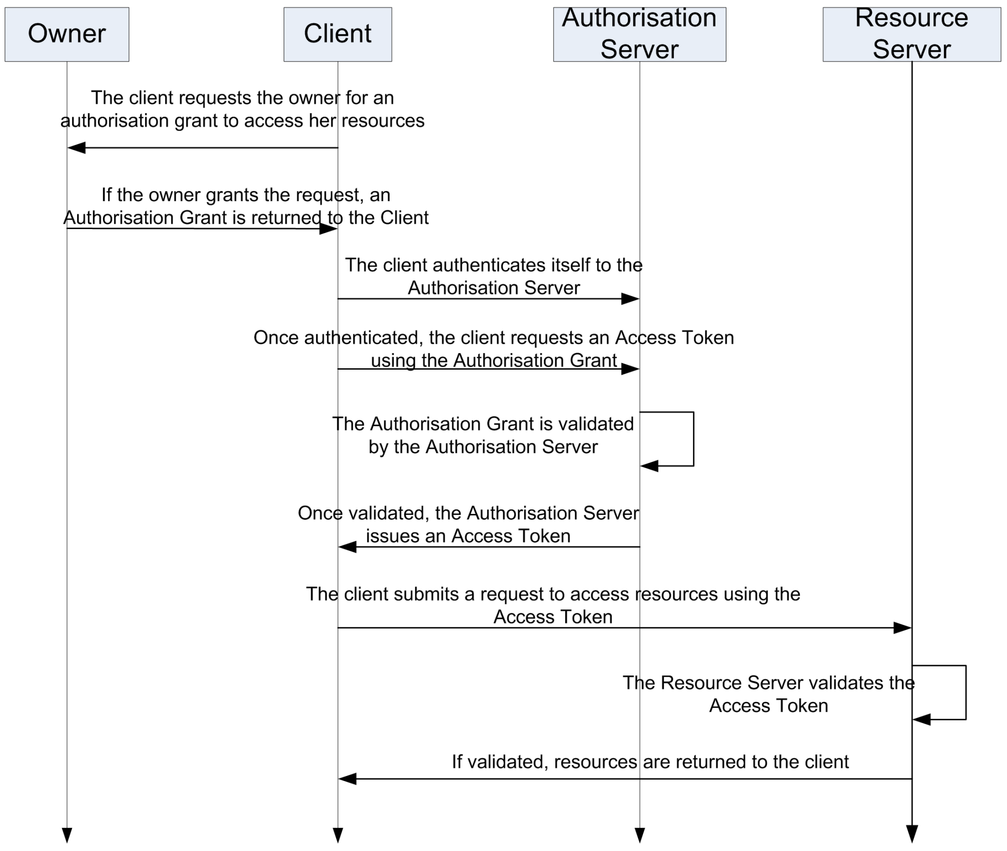

Owner
Client
Authorisation Server
Resource Server
The client requests the owner for an authorisation grant to access her resources
If the owner grants the request, an Authorisation Grant is returned to the Client
The client authenticates itself to the Authorisation Server
Once authenticated, the client requests an Access Token using the Authorisation Grant
The Authorisation Grant is validated by the Authorisation Server
Once validated, the Authorisation Server issues an Access Token
The client submits a request to access resources using the Access Token
The Resource Server validates the Access Token
If validated, resources are returned to the client

Figure 2.3: Protocol flow in OAuth.

1. A resource owner wants to delegate access rights to a client so that the client can access some protected resources of the owner, hosted at the resource server.

2. The flow starts with the client asking for authorisation of the owner to access those resources. The client can make this request directly to the owner or via the authorisation server using the HTTP re-direct method. In this abstract flow, it is assumed that this request is direct.

3. If the owner grants this authorisation request, a credential known as the authorisation grant is returned to the client. Depending on the type of method the client is using to request authorisation and on the methods the authorisation server can support, the grant will be one of the four available grants types.

4. The client then authenticates itself to an authorisation server and requests an access token using the authorisation grant.

5. The authorisation server validates the grant and if valid, issues an access token.

6. The client requests an access to the protected resources by presenting the access token to the resource server.

7. The resource server validates the access token and if valid, the access request is granted and the requested resource is returned to the client.

2.3 Dynamic Federation Management

Despite several advantages that SAML offers, one major constrain is the method by which trust is established and maintained. To enable federations using SAML, trust among participating organisations has to be pre-configured by system administrators (as discussed previously). Pre-configuring trust before any interaction hinders the establishment of a federation between two prior unknown parties in a dynamic fashion. Moreover, a well-known problem of SAML is that it becomes increasingly difficult to manage a SAML federation once the number of entities starts to increase [19].

There have been several proposals and drafts to overcome the problem of scalability and dynamism of SAML. The most influential work [20], called Distributed Dynamic SAML, prescribes that to trust any dynamically exchanged metadata, the metadata must be signed and the certificate that can be used to verify the signature must be included within the metadata. Assuming a trusted Certificate Authority (CA) issues the embedded certificate, each participating organisation will hold the root CA Certificates which can be used to validate the certificate chain in its TAL. Then, the trust on the metadata can be derived by just verifying the signature in the metadata using the embedded certificate with the traditional Public Key Infrastructure (PKI). The result of this proposal is the formulation of a working draft of the *SAML Metadata Interoperability Profile* which is a profile for a distributed SAML metadata management system [21, 22]. Based on the proposal and the working draft, an implementation of dynamic SAML has been developed by UNINETT [23] in their SimpleSAMLphp project [24, 11]. The SimpleSAMLphp is a native php-based implementation of the SAML protocol stack that can be used to quickly deploy a SAML IdP or SP. The proposal and the working draft and the SimpleSAMLphp implementation will allow the establishment of federations more quickly than previously possible. However, several crucial questions regarding trust assumptions at different parties have not been explored thoroughly. For example, each party validates the certificate of other parties using PKI to establish trust. However, is the established trust sufficient for any IdP to release sensitive user attributes to an SP which has been added dynamically, since there may not be any legal contract between them? And also, since there may be no legal binding, can an IdP trust that an SP would not abuse the released attributes in any way? Similarly, an SP will need to consider if it can trust any attributes that have been provided by a dynamically added IdP even though the SAML assertion containing those attributes are properly verified.

Furthermore, the SimpleSAMLphp implementation requires that the metadata of an IdP is already present at an SP so that the WAYF (Where Are You From) Service can display a list of IdPs to the user. Once the user selects an IdP, a SAML authentication request will be sent to the IdP. If the IdP has the capability to add an SP dynamically (e.g. an IdP deployed with SimpleSAMLphp), it can retrieve the metadata of the SP dynamically and store it temporarily in case the entity ID of that SP is not found in its TAL. To make this possible, the SimpleSAMLphp implementation requires that the entity ID of the SP has to be a URL from where the metadata can be fetched. In summary, SimpleSAMLphp only allows an IdP to retrieve any SP metadata, not the other way around. It can be regarded as a *semi-automatic* federation where an IdP has to be pre-configured at an SP and thus does not fully address the problem of dynamic federations.

There are some existing works that also provide proposals for dynamic federations. In [25], the authors propose a SAML extension to accommodate reputation data in XML format. According to their proposal, trust has to be negotiated based on that reputation data before any entity can join the federation. Each entity will maintain a dynamic list called the Dynamic Trust List (DTL), instead of the static TAL, which will contain the list of joined entities in the same federation with their reputation data and will be updated dynamically as the federation evolves. To realise their proposal, a novel exchange protocol was developed to request and respond to reputation data. Even though the authors have used the term *dynamic federation*, their proposal does not consider how two entities can be federated automatically. Instead it deals with the issue of enabling trust before two entities are federated and how this trust can evolve over time as they interact with each other. Hence, their proposal differs considerably from the true meaning of a dynamic federation.

The authors in [26] proposed a dynamic federation architecture, called *DFed*, based on SAML and Automatic Trust Negotiation (ATN) to establish trust between participating parties at run time. The DFed framework consists of different Dynamic Federation Service Providers (DFSPs). Each DFSO can act as an IdP and SP and consists of different components such as Gate Keeper (GK), Directory Services, Trusted DFSP Repository, SAML Agent and ATN Agent. A GK is responsible for the SSO Protocol, a Directory Service is responsible for storing attributes and policies, a DFSP repository stores the information of others federated SPs, a SAML Agent is responsible for carrying out the SAML Protocol and an ATN agent is responsible for ATN protocol and trust negotiation. When a user tries to access services from one DFSP (DFSP1) using the DFed protocol, she is shown the list of other federated DFSPs. If she chooses any of these DFSPs, she is forwarded to the selected DFSP (DFSP2). Upon completing the authentication at DFSP2, she will be returned back to DFSP1. At this point, the DFSP1 can request additional attributes of the user to take any authorisation decision. If the user wants to use another unfederated DFSP (DFSP3), she needs to provide its ID which is then used to start a trust negotiation protocol between DFSP1 and DFSP3. Upon completing the trust negotiation process, two DFSPs are federated automatically and the user can use DFSP3 to access services from DFSP1. An implementation based on SAML was presented to show the applicability of their proposal. The main limitation of the proposal is that it is required to alter SAML extensively in order to accommodate the interaction between different DFSPs. Furthermore, the discussion regarding the trust negotiation procedure is vague since it is not clear what type of information is needed during the trust negotiation procedure and how the negotiation procedure establishes different level of trust.

There is another solution proposed in [27] where trust values are calculated based on the modified Dijkstra algorithm and to calculate a distributed reputation based on the PageRank algorithm from Google and use the trust and reputation value to create dynamic federations. Like in [25], their proposal deals with the issue of dynamic trust establishment between different entities and does not consider how they can be federated dynamically. Furthermore, this also requires a major change of the SAML Protocol to integrate and utilise such trust values during protocol flow.

2.4 User-controlled Identity Management Systems

In this researche, the concept of a *User-controlled Identity Management System* is proposed for tackling several problems of existing Identity Management Systems and for allowing users to have more control over their own attributes. To enable this, a user-controlled Portable Personal IdP, which is hosted in a mobile device of the user, is introduced. Being a novel topic, there in no published literature on the User-controlled Identity Management System. The closest to the theme of the User-controlled Identity Management System is the concept of the Portable Identity Provider which is a special type of IdP. The initial concept of such an IdP can be found in [28] where the authors proposed their idea of *Identity-aware Devices*. An identity-aware device contains a local IdP hosted inside a Trusted Processing Module (TPM) in the device. The local IdP is coupled with a telecom provider which acts as the principal IdP. The user needs to create an account with the telecom provider and link the local IdP with the principal IdP. The linking procedure allows the local IdP to fetch some crucial user attributes from the principal IdP and store them safely in the local storage. Using the local IdP, the user can release those attributes to access services from an SP which is a part of the same federation as the principal IdP. Being part of the same federation as the principal IdP, the SP can validate the released attributes using existing technologies such as SAML. The idea of the user-controlled IdP is similar to this approach with one major difference: this approach requires a user to heavily rely on the principal IdP, i.e. the telecom operator whereas the proposed concept advocates the need to decouple the reliance on any other external IdP.

Another work related to this theme can be found in [29] in which a portable Liberty Alliance[1]-enabled IdP installed in a mobile phone is presented. The IdP is installed on the phone and is accompanied by a Relay Server which is maintained by a Mobile Operator. The relay server is responsible for providing the URL of the IdP. During authentication, the user provides the URL to the SP which uses it to send authentication request from the relay server which in turn forwards the request to the IdP. The IdP then authenticates a user using the PIN (Personal Identification Number) of the SIM (Subscriber Identity Module). The advantage of such a portable IdP is that the authentication information is not transmitted over the network. However, there is no discussion on how the federation is established and, with the absence of a federation, how an assertion can be validated. Moreover, the mobile operator has to establish a relay server meaning the operator has to offer a new service for it. Without any business prospect on their part, many operators may not be interested in bearing the associated cost. Moreover, apart from authenticating a user, this approach does not state how user attributes will be transmitted to an SP.

In [31], the author proposes a form of device-centric identity which allows a user to authorise a device to use a cryptographic key to identify the user and allow access to online services. The user can add new devices or remove old ones from the list of authorised devices. In essence, the authorised device can act as an IdP. The proposal is only an outline and how such a proposal can be envisaged is not explained.

Current popular smartphone Operating Systems such as Android [32], iOS [33] and Windows [34] are associated with their respective identity providers. For Android, the IdP is Google Account [35], AppleID [36] for iOS and Windows Live ID [37] for Windows. Among them, Apple ID and Windows Live ID are mostly used to access services associated with Apple and Windows respectively. Only Google has a Federated Login service based on the OpenID protocol to allow users to use their Google accounts to access services offered by other service providers [38]. Users carrying a mobile phone having a Google account can access supported online services while on the move. Even though it adds an element of portability, the Google IdP is not installed in a mobile phone and acts just like any third party IdP holding user attributes. Hence, it cannot be tagged as a user-controlled IdP.

Also, it is worth noting that every Identity Management System is bound by a set of functional, security, privacy and trust requirements. Unfortunately, none of the works mentioned in this section provides a thorough analysis of these requirements. In this thesis, these requirements will be introduced and analysed at appropriate chapters.

[1]More precisely the “Identity Federation Framework” (ID-FF) formulated by the Liberty Alliance (LA, in short) consortium [30]

2.5 Context-aware Identity Management

A large number of works can be found in the literature focusing on how contexts can be integrated either into an authentication framework or into an authorisation framework. There are only a handful of works that deal both issues of authentication and authorisation together and the number of works that analyse the effect of context awareness in all aspects of identity management is almost negligible. Each of these aspects is discussed in the following subsections.

2.5.1 Context-aware Authentication

A context-aware authentication mechanism uses the context of a user for authenticating the user. There are numerous such mechanisms. Here, a few of the most influential works are reviewed.

In [39], the authors have presented a secure and usable context-aware user authentication mechanism to be used in a hospital environment. The mechanism consists of several components such as clients, context monitors and context servers. A client is essentially a service provider equipped with a smartcard reader. Each user is provided with a smartcard with the capability to execute an applet. The applet is initialised with an RSA key pair generated using the identifier and password of the user. The identifier, the password and the private key are stored in the smartcard whereas the respective public key is stored in the context server along with the user identifier. A context monitor consists of different external sensors that determine the location of a user. Each context monitor is connected to the context server which is aware of the locations of all context monitors. To determine the location of a user, the context server uses either an RFID tag belonging to a user or the WLAN signal emitted by devices belonging to a user. Using these components, the authentication protocol is as follows. A user comes nearby to a context monitor and the monitor detects the presence of the RFID tag. The identifier of the RFID tag is transmitted to the context server which determines the identifier of the user using the identifier of the RFID tag. The server also determines the location of the user using the identifier of the user and the location of the monitor. Next, the user inserts her smartcard into the reader. The reader reads the encrypted identifier from the card and transmits it to the server via the client. The server decrypts the user identifier and then uses it to determine the user's location. The location is transferred back to the client and then the client determines if the user is allowed to access a service. The proposed mechanism is assessed to be secure against different attacks. The proposal has only considered the identifier to determine the identity of a user ignoring other attributes which may constitute the identity of the user. In addition, the proposal did not consider how it could be used for identity management.

A framework for authenticating a user based on the proximity to external sensors is presented in [40]. The authors argued that detecting a user inside a building could be quite tricky and suggested on using proximity sensors. The paper illustrates two situations discussed below.

The first scenario is known as the Personal Beacon Authentication in which a user can be authenticated using her mobile device and an external sensor known as the Personal Beacon. The beacon and the mobile device can communicate with each other using Bluetooth or near-field magnetic communication. The beacon uses a PKI for authenticating a user where the administrator of the beacon generates the pair of RSA keys and obtains a certificate containing the public key signed by a CA. The root certificate of the CA is installed in the mobile device. When the mobile device comes near to the beacon, the authentication mechanism is activated. If the device and the beacon communicate for the first time, an initial phase, known as the *enrolment phase*, takes place where the mobile device obtains the certificate containing the public key of the beacon. The mobile device validates the certificate by using the root certificate. Once validated, the enrolment phase concludes. This allows the mobile device to participate in a challenge-response protocol for authentication with the beacon every time they are nearby. For this, the device generates a random value (A) and passes to the beacon for signing. Upon receiving the random value, the beacon generates another random value (B) and concatenates two random values ($A||B$). Then, the concatenated random values are signed by the beacon and returned to the device. The device validates the signature of the concatenated random values and retrieves B which is then returned to the beacon. Once the beacon receives B, the user of the device is considered authenticated. After authentication, how the user accesses services is not explored in the paper.

The second scenario is known as the Organisation Authentication Beacon in which a device does not have any location information. Instead, the location information is retrieved from the installed proximity sensor known as the organisational beacon. Like before, a mobile device and an organisational beacon communicate using Bluetooth. The administrator of the beacon sets up the X.509 certificate whereas the root certificate is installed in the device. During communication, a Transport Layer Security (TLS) connection is established using the X.509 certificate. The location information is securely transmitted using this TLS connection from the beacon to the device. Upon receiving the location information, the device switches itself to the authenticated mode. What is achieved in this mode is not further explored in the paper.

The proposal and its implementations did not consider mutual authentication since the identities of the beacons are verified using certificates whereas the identity of the device remains unverified. The lack of identity verification of a user will allow every user nearby to the beacon to access same services without any personalisation. Furthermore, an SP will not be able to provide restricted services to a user without knowing her identity. Also, the paper did not consider how this setting could be explored for identity management.

Another paper that describes how contextual data can be used for mutual authentication of users can be found in [41]. The authors have proposed a mechanism consisting of two entities: users and context providers. The context of a user consists of several information such as: time, location and activity. A context provider is responsible for providing secure and verifiable contexts of a user retrieved using different external sensors. Two users authenticate each other by retrieving their respective contextual data from the respective context provider and exchanging them with each other. The proposed mechanism utilises Identity Based Encryption (IBE) for secure exchange of contextual information and a combination of timed fuzzy logic and private set intersection for privacy-preserving calculation of confidence that determines if a user can be considered authenticated to another user. Since the proposal deals with mutual authentication of two users, it is impractical to be integrated with online services. Furthermore, the authors did not consider the suitability of their proposal for identity management.

In [42] Hsieh et al. have presented an One Time Password (OTP) authentication scheme that utilises contextual information such as time and location of a user device. A service provider, regarded as the *Application Server* in the paper, utilises such contextual information to provide services to a user. The user is identified using the International Mobile Subscriber Identity (IMSI), a unique identifier for the mobile device of the user whereas an OTP is generated by concatenating the location of the mobile device retrieved using the GPS sensor of the device and the time of service request. According to the proposal, this information is transferred in encrypted format to the service provider where the provider decrypts and retrieves such information using PKI. Unfortunately, the proposal is quite sketchy in the sense that the authors did not elaborate the mechanism by which PKI could be established for their proposal and the location could be extracted from the concatenated OTP. In addition, the absence of any security analysis undermines the credibility of the proposal. Furthermore, it is not clear if this approach could be used for identity management.

In order to strike a balance between security and usability, the authors in [43] present a probabilistic model for authenticating a user (generally the owner) in a mobile device. The model utilises two factors: active authentication factor and passive factor. An active authentication factor symbolises the existing mechanisms for authenticating a user in the current generation of smart mobile devices, namely, PIN, Password and No-PIN, whereas, a passive factor represents the context of a user, namely, in the form of locations and the time of a day. The model automatically chooses an active authentication factor based on the combination of a specific location or time. For example, the model relaxes the restriction of using a PIN or password when a user is at her home or at her workplace, however, imposes the restriction of a PIN or password for other places. Another example is that a PIN can be used at a user's home or workplace and a password can be used for other places. A user can set her preferred mode of active authentication factor for different contexts. The proposed model has been implemented in a smartphone app which has been used for an extensive user study. The result of the study suggests an enhancement in usability without compromising security. The focus of this work is to authenticate a user on a mobile phone and hence, it does not consider the issue of identity management using contexts. Moreover, since the authentication is conducted on a mobile device, it is impractical for integrating into online services.

2.5.2 Context-aware Authorisation

A context-aware authorisation mechanism uses the context of users to authorise (in other words, to control accesses) them for accessing a service. Among many such mechanisms, a few of the most influential ones are presented.

One of the earliest works to propose how contextual information can be incorporated for access control scenarios is Generalized Role Based Access Control (or GRBAC) [44] which is an extension of the popular Role Based Access Control (RBAC) model [45]. In this approach, three different types of roles are defined: Subject Roles (denoted as *SRole*), Object Roles (denoted as *ORole*) and Environment Roles (denoted as *ERole*). Subject roles, similar to the concept of roles in RBAC, represent the role of an individual in a system. Object roles represent different security related properties of an object in a system. Finally, environment roles represent contextual information that expresses the time of day, weather conditions and other spatial and temporal values. Similar to RBAC, GRBAC defines policies that determine which operations (denoted as *op*) are permitted in a transaction. A transaction is defined as a tuple of <*SRole, ORole, ERole, op*> which expresses that a subject having *SRole* is permitted to perform operation *op* on an object with *ORole* under the conditions of *ERole* contextual information. Compared to the commonly used RBAC model, GRBAC is more expressive which makes it suitable for context-aware authorisation schemes. Based on the concept of GRBAC and environment roles, an architecture has been presented in [46] which illustrates the applicability of leveraging contextual information in the form of environment roles to control accesses in a smart home environment. The authors represent environment roles in different forms. For example, the roles of users in the smart homes are represented using a hierarchy of family such as father, mother and siblings. Temporal roles are represented using a hierarchy of day and time whereas spatial roles are represented using internal locations within the smart home such as the first floor, second floor, bedroom, kitchen and so on. Based on these environment roles, the authors have implemented a system that utilises the transaction tuple of GRBAC for controlling accesses to different resources inside the smart home. The smart home has been equipped with sensors to capture the environment roles and the user has the ability to specify policies that dictates which resources are accessible by which roles under what contexts. Even though both these works are influential in nature in the domain of context-aware authorisation, one major problem is that environment roles introduce complexities in administering policies which in turn makes it difficult to ensure that the policies are conflict free. In addition, a large number of environment roles makes the system very difficult to manage manually.

To tackle this problem, Zhang and Parashar have proposed a Dynamic RBAC (DRBAC) model [47] by extending the traditional RBAC model. The model adjusts static roles and permission assignments dynamically using contextual information. The DRBAC consists of several components:

- *USERS* representing the set of entities who try to access services from a service provider.
- *ROLES* representing the set of roles that users can assume in a service provider.
- *PERMS* representing the set of permissions for accessing one or more resources in the provider.
- *ENVS* representing the contextual information of a user in the system.

DRBAC defines different rules to assign roles (from the *ROLES* set) to a user (from the *USERS* set) and permissions (from the *PERMS* set) to resources and uses policies to determine which user having what roles can access which resource(s). One of the striking features of DRBAC is its capability to capture the changes of the environment in the system and of the user and to enforce permissions accordingly. For example, a user having roles r_1 and r_2 can assume role r_1 only if she is at location l_1 and can assume role r_2 only if she is at location l_2. Depending on the roles at a particular context, the user is allowed to access particular resources. One limitation of the proposal is its lack of any explanation on how the location of a user would be detected by the provider. In addition, there is no discussion if the proposal can be used for identity management.

None of these works consider how to verify the authenticity and validity of contextual information. To overcome this limitation, a context sensitive access control architecture was defined in [48]. The architecture introduces several components including Context Owner (CO), Context Provider (CP), Context Broker (CB), and Context-aware Service Provider (CASP). A CO is the owner of the contextual information that is collected from different sources. It is solely responsible for deciding when, why, how and where contextual information is released, and who can store such information. A CP is responsible for ensuring different contextual information is released as per the requirements of the CO of the respective information. A CASP is responsible for providing services to a user, based on her context. The authors provide a detailed protocol flow using their architecture and explain the mechanism for verifying the validity and authenticity of contextual information. The major problem of the architecture is the reliance on several external parties which may be difficult to implement for federated services considering different security domains and trust issues. Also, it is unclear how their proposal can be used for identity management.

The authors in [49] have proposed a model describing how contextual information can be included during an authorisation process in highly dynamic environments. Contextual information is provided by different devices which are regarded as observers. The validity of the contextual information is verified by a trusted third party. The defined model consists of contextual conditions which are tuples of the form (<*context, operation, object*>) and dictates which entity with which context can perform which action on which object. A user is authorised to perform the specified operation on the respective object only if the context is valid. The authorisation decision also includes several conditions that are combined using logical rules. Their model of tuple is similar to the eXtensible Access Control Markup Language (XACML) model [50] and does not provide any additional advantages over the XACML model.

2.5.3 Context-aware Authentication and Authorisation

Frameworks presented in this section utilise the context of a user for authentication as well as for authorisation while accessing a service. Like before, among many existing works, a few of the most influential ones are discussed below.

Based on the concept of GRBAC in [44] and the architecture presented in [46], a framework called Context-Aware Security Architecture (CASA) and its proof-of-concept implementation are presented in [51] which demonstrates how contextual information of a user can be utilised for authentication and authorisation of the user in a smart home environment. The framework can be used to develop secure context-aware services and consists of different components such as Policy Editor, Security Management Service (SMS), Authorisation Service, Authentication Service, Environment Role Activation Service (ERAS) and Context Management Service (CMS). The policy editor allows any administrator to assign roles to users and write policies using the Generalized Policy Definition Language (GPDL) [44] for controlling accesses to different resources. The SMS is responsible for storing and managing roles and policies and providing them to other components whenever required. The authorisation service is responsible for enforcing authorisation decisions during service access requests. For this, the policies are retrieved from the SMS and it is determined if the user having a particular role can access the requested service. The authentication service is responsible for authenticating a user using either a push or pull method. In the push method, a user submits her credential during each service access request whereas in the pull method a user is authenticated when a particular service needs authentication. The ERAS is responsible for activating different environment roles depending on different states of the environment which are collected from the CMS. Finally, the CMS is responsible for collecting and disseminating states of the environment using different sensors. The authors have presented a detailed discussion regarding the implementation of the CASA and how it can be utilised to develop other context-aware applications. Even though this is one of the earliest works that showcased the potential for secure context-aware services, one main drawback is its limited focus within the domain of a smart home. This makes it difficult to deploy it in other scenarios such as online services.

The authors in [52] argue that the existing RBAC model fails to meet the need for a flexible, on-demand and dynamic authentication and authorisation solution for the healthcare domain. To mitigate this problem, the authors have presented a dynamic and context-aware infrastructure which can fulfil different security and privacy requirements. Their proposal recognises five different types of contexts: time of an access request, location in the form of IP address, physical or semantic location (e.g. inside hospital), user ID used to identify an entity, object type representing the type of resource or service requested and object ID used to identify the requested resource or service. The infrastructure has three main components: an authentication engine, an authorisation engine and a context service. The authentication engine is responsible for authenticating a user and supports different modes such as username/password and biometric (e.g. fingerprint, iris, signature and voice). The authorisation engine is responsible for authorising a user and enforcing policies while accessing services. It allows administrators to setup policies using WS-Policy [53] which is an XML-based standard to specify access control rules for different resources. Users are categorised in different roles and policies define which role having which permission can access which resources. The context service component is responsible for evaluating a context type and is invoked by the authorisation engine. The implementation details are quite sketchy and it is unclear how different contexts are collected and disseminated. Also, the proposal does not discuss how the authenticity of contexts can be validated. In addition to this, it is unclear if their proposal can be adopted for online services

An agent-based framework for authentication and access control in context-aware services is presented in [54]. The framework uses an additional entity called an authentication and access control agent which is actually a trusted third party. The context is represented using different information: the location of a user and services, environmental information (such as temperature, brightness and loudness), different roles of a user, different network types (such as office LAN, public LAN and home LAN), etc. The agent is responsible for collecting contextual data from different servers such as a location management server, presence management server and user profile management server, for obtaining the user's identity from an IdP, authenticating a user or the user's terminal and obtaining a user's attributes. The administrators can define authorisation policies. Based on the identified contexts, the attributes of the user and the conditions defined in the policy, the user is granted or rejected to access a service. However, there is no discussion how services can be deployed. The interactions between the authentication and access control agent and the service providers are not very clear. Also, the architecture relies heavily on external servers which, in one hand, will be difficult to manage to provide federated services and, on the other hand, will be difficult to manage the privacy of the user across many external servers.

An attribute-based framework for authentication and authorisation in mobile environments has been presented in [55]. Here, attributes are represented as the context of a user which can be utilised to authenticate as well as to authorise the user during a service access request. Examples of contextual attributes are different properties - such as time, temperature and location - of the environment from which a user makes a request to access a service, the nature (e.g. premium or basic) of the service requested and the proof of a specific transaction in the form of an e-receipt and e-token. The framework consists of several components: Client, Attribute Provider (AP), Service Provider (SP), Attribute Verifier (AV) and Certificate Authority (CA). A client represents a user and her compatible mobile device. The user can use the mobile device to access services. The mobile device is responsible for gathering contextual attributes of the environment and storing the attributes securely in a Trusted Processing Module. This allows the device to release the stored attributes to an SP in such a manner that the SP can verify the authenticity of the gathered attributes. An AP is responsible for disseminating contextual attributes of an environment using different sensors. An SP is responsible for providing services to a user. An AV is responsible for verifying the authenticity of attributes. For secure interactions between all entities, a PKI using a CA is deployed. The protocol flow assumed by this framework is as follows. A cafe has deployed this framework to allow its buyers to avail premium services from an SP. Let as assume that a user visits the cafe and buys a coffee from the cafe. As a proof of her purchase, an e-receipt is stored in the TPM of the mobile device. While enjoying the coffee, the user decides to access the said services from the SP. She uses her device to fetch contextual attributes of the cafe from the AP which are then stored in the TPM of the device. The e-receipt and the contextual attributes are then released to the SP. The SP validates the e-receipt which proves that the user is a client of the cafe. Then, the attributes are forwarded to the AV for validation. A successful validation proves that the user is currently at the cafe. The result of the validation is sent back to the SP. At this point, the SP allows the user to access the requested services. The main limitation of the proposal is its lack of any discussion regarding implementation. It is not clear how the AP will be installed and maintained in the environment. Since installing and maintaining APs in different environments will be expensive, it is unclear which entity (the cafe or the SP) will bear the associated cost.

An interesting work where contextual information such as roles and locations are used for authenticating and authorising a user has been presented in [56]. The approach uses passive and active authentication cues for authenticating a user. The active authentication cues are usernames/passwords, while the passive authentication cues are contextual information, namely roles and the location of the user. An access control mechanism is proposed in which the role and location information are combined in creating policies to determine what resources can be accessed by which users. The access control system and the policies are based on the XACML model. The role of the user is retrieved from the system using the user-supplied username once the authentication is completed. The location of the user is determined by allowing the user to forward the location information which is retrieved from a QR Code (Quick Response Code) using a mobile phone. The proposed architecture consists of several components such as Core Access Management Module(CAMM), User Database and Policy Store, Client Mobile Device, Authentication Site and Additional Context Cues (ACC). The CAMM is responsible for generating and managing QR codes in different indoor locations and for gathering usage analytics such as the popular positions among users for authenticating users via QR codes as well as all authentication attempts via different QR codes. The database and policy store is responsible for storing attributes of users and policies that define access control rules. Each user needs to utilise a mobile device equipped with the Wi-Fi capability to make authentication requests, a camera to scan QR codes and corresponding software to decode any scanned QR code. An authentication site is the physical place where QR codes are displayed and users scan the codes to make authentication requests. The ACC is responsible for gathering additional contextual information, for example, the calendar information of a user. A typical use-case using this architecture is as follows. A QR code is displayed at an authentication site. A user uses her mobile device to scan the QR code. After decoding the code, an authentication request is sent automatically to the server. At this point, the user needs to submit her username and password for authentication. Once authenticated, attributes of the user (e.g. roles and other information) and associated policies for the authentication site are retrieved from the store. The server evaluates the policies to determine if the user can access a resource in that authentication site. This is another work in which the focus is limited in controlling accesses in a physical location. This makes it difficult to deploy for online services.

2.5.4 Context-aware Identity Management

The only work to discuss the issue of context-awareness in Identity Management can be found in [57] where the authors have proposed and implemented a framework for selecting appropriate partial identities of a user depending on her context. To design the framework, the authors raised a few crucial questions closely related to the issues of context-awareness of identity management:

- What data can sufficiently constitute a partial identity of a user including her contextual information?

- What will be the source of this contextual information and how such data can be collected in a standardised way?
- Which partial identity should be used in a particular situation?
- How the information regarding the selected partial identity can be propagated in a privacy-friendly manner?

These questions have been utilised for formulating requirements and based on the requirements a novel framework has been proposed. The framework consists of three major components: i) Contextual Information, ii) Personal Identity Manager and iii) Privacy Manager. The contextual information component consists of three sub-components: Context Provider, Context Server and Context Request. A context provider resides in a mobile device of a user and is responsible for retrieving contextual information, such as times and outdoor locations via GPS and indoor locations via Ultra-wide Band (UWB) signal, from different sensors and providing them to other components. In addition, the context provider is equipped with an interface to allow a user to set up meaningful semantic contexts such as *at work, at home, etc.*. The context server is responsible for storing contexts of different users. For this, it queries for contexts from the context provider of each user and stores them at its end. The context requester also resides in a mobile device of a user and is responsible for retrieving contexts of a user from the context provider or contexts of other users from the context server.

The personal identity manager consists of three sub-components: personal information module, contextual information module and policies and rules module. The personal information module is responsible for storing information regarding personal identities in an XML file. A user's identity has been categorised in three different types: personal identity (PID), corporate identity (CID) and social identity (SID). Essentially, each such identity represents a partial identity of a user and a user can add any information and define it to be a part of a certain partial identity. The contextual information module is responsible for retrieving the user's contextual information from the provider and presenting such information in a graphical interface. The policies and rules module provides an interface to the user for setting up rules and policies that dictate which conditions must be satisfied to request and access a certain partial identity.

The privacy manager module consists of three sub-components: the profile information module, privacy manager decision module and profile zoning module. The profile information module allows a user to select her attributes to define different profiles. For example, a user may select her official email address, office phone number and schedule calendar to be included in the office profile whereas her personal email address and mobile phone number for the personal profile. Finally, the privacy manager decision and profile zoning modules consult with different modules to determine which profile can be released to which entity.

Even though the proposal is promising, it lacks substantially in several aspects. Firstly, the relevance between a partial identity and a profile has not been clarified. It is unclear if a profile is a sub-category of a partial identity. In addition, what information constitutes a partial identity is never thoroughly analysed and hence, the first question remains unanswered. Secondly, the focus of the proposal is somewhat narrow as its applicability has been investigated in MANET (Mobile Ad-hoc Network) where different users can share their profiles using contextual information. It has not been investigated how this proposal can be utilised for accessing online services. Thirdly, the implementation is vague in the sense that only the interfaces for different components have been presented and discussed, ignoring the protocol that has been used during user interactions. The lack of discussion regarding practical use-cases and protocol flow makes it a bit difficult to understand the usefulness of the proposal. Finally, the inclusion of external components makes this framework impractical to be used with the existing architecture of federated services without analysing the underlying trust requirements between themselves.

2.6 Attribute Aggregation

Attribute Aggregation is the mechanism to combine attributes from multiple identity providers while accessing a single online service. There are different models, either in the form of proposals or implementations, currently available. A detailed discussion regarding existing attribute aggregation models can be found in [58, 59]. In addition to discussion of different models, the authors in [59] have provided an analysis on several challenges of attribute aggregation and identity management. One of the limitations of both works is the lack of any comparative analysis among the available attribute aggregation models which makes it difficult to understand how each model performs against others in different aspects and to identify the strengths and weaknesses of each model. Moreover, none of them has considered the need for an analysis of different (functional, security, privacy and trust) requirements.

To fill in this gap, the authors in [60] have presented a survey of requirements required for attribute aggregation from multiple sources. The requirements have been formulated by distributing questionnaires in different international communities of security professionals. The questionnaire consists of several questions in different aspects such as general, privacy, control, protocol and trust. Based on the responses from security professionals, a list of requirements is formulated. Then, the list of requirements has been used to compare four existing aggregation models and to identify their strengths and weaknesses. However, the trust, functional, security and privacy requirements have not been properly categorised and many trust, functional, security and privacy requirements have not been even considered. Based on the requirements identified in [60], the authors in [61] have presented a proposal and implementation of a novel Attribute Aggregation model, called *Linking Service*, for Federated Identity Management with the aim to counteract the weaknesses in the existing models.

Even so, it has been found in this research that all existing proposals have some limitations. Many of these models are theoretical in nature and have no implementation. The models that have been implemented either require complex user interactions or are based on unrealistic assumptions. This is one of the key reasons why the process of attribute aggregation has not yet been popular in online services. The findings will be presented in Chapter 8 where the issue of attribute aggregation will be discussed thoroughly.

2.7 Conclusion

This chapter provides a short introduction of three popular Identity Management Systems. These three Identity Management Systems have been used in Chapter 4 for a comparative analysis. Based on the comparative analysis, a system has been chosen for this research.

Additionally, this chapter also reviews the existing works on selective aspects of Identity Management which have been chosen for this research. From the review presented in previous sections, the following conclusions can be drawn:

- A SAML-based federation suffers from a serious limitation: a federation cannot be created in a dynamic fashion. This makes it hard to scale and maintain as the number of parties starts to increase in the federation. The existing work allows a federation to be created dynamically in a semi-automatic fashion, requires a major change of SAML or only discusses how a federation can be created, ignoring other aspects of maintaining a federation.

- The concept of User-controlled Identity Management System is new and hence there exists no reference of any existing work on this concept. The closest, in relation to this concept, that can be found is the concept of the Portable Identity Provider. However, such an IdP is still reliant on the external IdP and thus it will be not under the full control of the user.

- There exist a substantial amount of works that investigate how contextual information can be integrated with the authentication and authorisation process. However, there is only one work that discusses the issue of context-awareness with respect to all aspects of Identity Management. Unfortunately, the proposed approach has several limitations and will be hard to deploy in existing architecture of federated services.

- Attribute Aggregation being a novel topic, also has a very few existing works. Most of these works are theoretical in nature meaning they have not been implemented. The aggregation models that have been implemented have serious limitations: they require complex user interactions or are based on unrealistic assumptions.

Chapter 3

Identity Management: Basic Terminologies and Models

3.1 Introduction

A number of groups have worked in isolated projects resulting in various incompatible Identity Management Systems. Not only these systems are inconsistent with each other, but different groups came up with different notions and semantics of many central topics related to Identity Management. For example, some fundamental issues of Identity Management such as Identifiers, (Digital) Identity and Partial Identity are treated in different ways in [62, 63, 64, 65, 66, 67]. The term *Identity* is defined as "the equivalent to the physical presence of that entity" in [62], whereas [64] defines Identity as "the set of permanent or long-lived temporal attributes associated with an entity". The same concept of Identity is defined by "the dynamic collection of all of the entity's attributes" in [63]. The first definition is a mere abstraction, whereas the second and third have significantly different meanings. Such differences on core topics introduce inconsistencies and such inconsistencies in turn introduce confusion, making it difficult to gain an understanding of Identity Management.

To rectify the situation, an effort has been undertaken to develop a consistent vocabulary related to Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity and Identity Management. The goal is to build a common language which can be used as a point of reference in any future research project. The result of this work is an evolving document [67]. Though appreciable, their focus is somewhat narrow as it only deals with those definitions that are related to privacy aspects of Identity Management. The domain of Identity Management is much broader, and thus the document in [67] is not comprehensive in nature. A more general effort with the similar goal can be found in [63] that provides definitions on different issues related to Identity Management omitting the relation between each definition, and thus making it difficult to apprehend the whole concept. In addition, definitions in [63] are given in textual forms which can lead to both misinterpretation and misunderstanding.

A mathematical model formally defining the different issues of Identity Management would extend and enhance the textual definitions by removing discrepancies, reducing confusion and allowing a solid understanding, based on mathematical foundations. A paper toward this direction can be found in [68]. Unfortunately, similarly to [67], this work focuses mainly on the privacy aspects of Identity Management. Therefore, as noted in [69], a mathematical modelling of the general aspects of Identity Management is still missing. To fill in this gap, a comprehensive mathematical model of Identity and Identity Management has been built using a compiled set of textual terminologies as the basis.

This chapter presents the compiled set of textual terminologies which are accompanied by their respective mathematical notations. In this manner, the mathematical model of different core topics related to Identity Management is gradually built up by mathematically defining atomic issues of *Entity*, *Domain*, *Attribute*, *Identifier*, *Partial Identifier*, *Partial Identity* and *Data*. Then, these concepts are combined to build up a mathematical model to define *Digital Identity*. Next, the devised model of Digital Identity is used to define a basic model of Identity Management and show how the digital identity model can be used to characterise the behaviour of an Identity Management System. And finally, three applications are discussed to illustrate how the model can be used to analyse different scenarios related to Identity Management and to characterise three popular IdM models mathematically.

3.2 Entity, Identity, Identity Management and Other Related Terminologies

3.2.1 Entity

An entity is a physical or logical object which has a separate distinctive existence either in a physical or a logical sense [70]. Both animate and inanimate objects can be considered as entities. Different disciplines interpret the term *Entity* in different ways and with different meaning. In the scope of this thesis, a person, an organisation or a machine (computer) operated by any person or organisation, a network, a network element, a software program, a hardware and a web service will all be denoted as entities. An entity can be classified in different ways. The following definitions classify different entities and have been collected from [71].

Physical Entity. A physical entity is an entity for which some sort of physical constituent is compulsory. That is, an entity which exists physically with a separate distinctive existence in the physical world can be denoted as a Physical Entity. Every living and non-living object that surround us are physical entities. However, in the scope of this thesis, the focus will be restricted to people and hardware as physical entities.

Virtual Entity. A virtual entity is an entity which is or has been the product of the mind or imagination. Thus, an entity which has no physical existence and exists inside someone's mind or imagination with solely logical meaning can be denoted as a Virtual (or Logical) Entity. Therefore, any kind of thought, concept, perception, category or abstraction can be thought of as a virtual entity. Examples are specific roles (such as the CEO of an organisation or the President of a country), different groups (British citizens), avatars in online games such as Second Life, Computer programs, etc. [71]. In the scope of this thesis, only organisations and computer programs will be considered as virtual entities.

Following the imposed restrictions, an entity can be physical or virtual but not both; meaning the sets for physical and virtual entities are disjoint.

Digital Entity. A digital entity is any entity which primarily exists in some digital domains (see below), e.g., as digitally encoded information or as a running computer program. In other words, the digital encoding or representation of an entity (Physical or Virtual) so that it can have its unique existence in the digital world can be denoted as a Digital Entity.

The following sets of entities are now formally introduced.

Definition 1 *Let E denote the set of all entities. The set of people (or persons) is given by P, while the sets of organisations, hardware and software are given by O, H and S respectively. Furthermore, PE is the set of Physical Entities and VE the set of Virtual Entities.*

According to the above definitions, persons, organisations, hardware, and software are all entities of different types and are disjoint from each other, and their union constitute the whole set of entities. Therefore the following equations hold:

- $E = P \cup O \cup H \cup S$;
- $P \cap (O \cup S \cup H) = \emptyset$;
- $O \cap (S \cup H) = \emptyset$;
- $S \cap H = \emptyset$.

Since it is assumed that only person(s) and hardware constitute a Physical Entity, therefore $PE = P \cup H$. Similarly, as it is assumed that a virtual entity is either an organisation or software, it follows that $VE = O \cup S$. From the rules above, the sets PE and VE are disjoint, i.e. $PE \cap VE = \emptyset$, and the sets of PE and VE are both subsets of E.

In this thesis, the focus is on digital identities (DE). Therefore, all entities are considered as digital entities and from this point on, when entities are mentioned, it will imply digital entities. In this sense, the sets of entities and digital entities are identical in this thesis. For brevity, the symbol E will be used to denote the set of digital entities.

3.2.2 Domain, Application Domain or Identity Domain

A domain is the encompassing environment under which an entity exists and operates. In its most general form, a domain is everything that surrounds us, physically and virtually. However, we human beings tend to divide the whole domain into different smaller domains using social terms and conditions to define them separately. We then present different images of ourselves using roles and relationships in different domains. Examples of domains include Personal, Family, Friends, Work and Country. Such different representations of the same person are, in fact, common traits of the modern society and allow us to establish, maintain and nurture different relationships simultaneously [72].

A digital entity allows a person to reflect her digital image into the online world. The online world can also be divided into several domains, much like the physical world. However, unlike the physical world, social roles and relationships are not the predominant factors that divide the domain; it is mainly achieved through the concept of an application domain or namespace in which each (digital) entity is represented and identified uniquely. An Application Domain, also known as the Identity Domain, is essentially the virtual boundary, namespace or environment in which an entity can be uniquely identified. The scope of an application domain determines its validity. There are different scopes of an application domain: scopes can be local (the application domain is valid inside a department), enterprise-wide (the application domain is valid inside the whole organisation), national (the application domain is valid inside a country) or global (the application domain is valid all over the Internet). The focus in this thesis is on such an application domain. When a domain is mentioned, it will actually imply an application (or identity) domain and hence the terms will be used interchangeably throughout the thesis. Note that the term *Context* is used in many Identity Management literatures to describe the same meaning as the application domain. Interestingly, the term *Context* has a specific meaning in the literature of Context-aware Services which will be explored later in the thesis. To avoid any ambiguity over the term of *Context* between these two disciplines, the term *Context* will be avoided while talking about application domains.

Whenever a digital entity is mentioned, it is important to elaborate the domain it operates in. A subscript in each notation will denote the domain it operates in. So, E_d denotes the set of defined entities in domain d. If the domain is clear from the surrounding discussion, the subscript will be omitted. The notation $DOMAIN$ will be used to denote the set of domains.

A real world analogy would be helpful to understand the modelling. Two systems are considered: the first is for blogging and the second is for emailing. These two systems, denoted as BLOG and the EMAIL respectively, represent two different application domains. It is assumed that each system has two users, i.e. two defined entities: $JOHN$ and $RAHIM$ in the BLOG, and $RAHIM$ and $ALICE$ in the EMAIL. Therefore:

$$E_{\mathrm{BLOG}} = \{JOHN, RAHIM\} \text{ and } E_{\mathrm{EMAIL}} = \{ALICE, RAHIM\}$$

In the following sections, different elements of the model will be explained based on this example.

3.2.3 Attributes

An attribute is a distinct, measurable, physical or abstract *named* property belonging to an entity in a given application domain and whose *value* can be used to identify that entity (not necessarily uniquely) within that application domain [73, 63]. Here, the term “identify” is used with its literal meaning. This term will be defined more precisely in later sections.

According to this definition, each attribute of an entity has a name and a value. The name of the attribute (or simply the attribute) alone cannot identify an entity, the attribute together with its value need to be used to identify an entity (not necessarily uniquely). In any application domain, the value of an attribute is provided either by each entity or is provided by a third party on the entity’s behalf. It is assumed that *Age* is an attribute in an application domain and its value, say for example, 30, can be used to identify a set of entities using the query: “Identify those in the application domain whose Age is 30”. This will return a set of entities which may be empty if there is no entity whose age is 30 or may contain just one element if there is only one entity with age 30 or more than one element in case there are more than one entities in the application domain whose age are 30. Similarly, a postcode is an attribute whose value can be used to identify a set of entities who live within the same postcode area.

Another important aspect of an attribute in an application domain is its data type which defines the values the attribute can take. For example, in some application domains, the data type of the attribute *Age* could be String, whereas in other application domains the date type could be Integer. The data type of an attribute, hence, depends on the specific application domain. For brevity, data types of attributes will not be considered in this thesis. Instead, the focus will be on the attribute (in other words, the attribute name) and its value.

The value of an attribute may or may not identify an entity uniquely. In common terminologies, such values are also known as *Data* regarding an entity and *Personal Data* when the involved entity is a Person. In the mathematical model derived in [68], attribute values are regarded as Data as well. However, since each datum can be associated with at least with one entity and can be used to identify entities (not necessarily uniquely), it is semantically more meaningful to consider data as attributes. For example, reputation data on eBay or Amazon, a blog entry and comments on a blogging site, photos and statuses on social networking sites can be used to identify at least one entity. The query "who has positive feedback of at least 95% on ebay" would identify a group of entities with positive feedback of 95% or more and is much like the query "who lives at the postcode of 'G3 7AH' in the UK" identifying the group of people living in that particular postcode area.

The notation A_d will be used to denote the set of attributes in domain d. Similarly, the notation AV_d will denote the set of attribute values for attributes defined in A_d and provided by the entities (or by third parties on behalf of the entities) in domain d.

Returning to the example, it is assumed that the domain $BLOG$ has three different attributes: username, age of the respective entity and the postcode of residence. For the entity $JOHN$, the values of these attributes are $john$, 32 and $G3$ respectively, while for $RAHIM$, the values are $rahim$, 21 and $G3$ respectively. Therefore:

$$\begin{aligned} A_{BLOG} &= \{username, age, postcode\} \\ AV_{BLOG} &= \{john, rahim, 21, 32, G3\} \end{aligned}$$

Note that the set of attributes in different domains may overlap since the same attributes may exist in different domains. Therefore, for two different domains d_1 and d_2, there is no requirement that $A_{d_1} \cap A_{d_2} = \emptyset$.

Now, different functions will be introduced to relate attributes and entities to their values in an application domain.

Definition 2 *The partial function $atEntityToValue_d : A_d \times E_d \rightarrow AV_d$ returns the corresponding value of an attribute for a specific entity in domain d.*

The reason $atEntityToValue_d$ is defined as the partial function is that not all entities might have a value for a respective attribute. Therefore, the function is only defined for the entities that have a value for those attributes.

This function can be used to define another function $atToValueSet_d : A_d \rightarrow \mathcal{P}(AV_d)$ that, for a specific attribute, derives the set of values associated with some entities. Formally, for any attribute $a \in A_d$:

$$atToValueSet_d(a) \overset{\text{def}}{=} \{atEntityToValue_d(a, e) \mid e \in E_d \ \& \ atEntityToValue_d(a, e) \text{ is defined }\} \tag{3.1}$$

Equation 3.1 in turn can be extended to define a function $atSetToValSet_d : \mathcal{P}(A_d) \to \mathcal{P}(AV_d)$ that retrieves the set of values for a set of attributes in domain d in the following way, for any set of attributes $B_d \in \mathcal{P}(A_d)$:

$$atSetToValSet_d(B_d) \overset{\text{def}}{=} \bigcup_{b \in B_d} atToValueSet_d(b) \tag{3.2}$$

According to the example:

$$\begin{aligned}
atEntityToValue_{BLOG}((username, RAHIM)) &= rahim \\
atEntityToValue_{BLOG}((age, RAHIM)) &= 32 \\
atEntityToValue_{BLOG}((postcode, RAHIM)) &= G3 \\
atEntityToValue_{BLOG}((username, JOHN)) &= john \\
atEntityToValue_{BLOG}((age, JOHN)) &= 21 \\
atEntityToValue_{BLOG}((postcode, JOHN)) &= G3
\end{aligned}$$

$$\begin{aligned}
&atToValueSet_{BLOG}(username) \\
&= atEntitToValue_{BLOG}(username, JOHN) \cup atEntityToValue_{BLOG}(username, RAHIM) \\
&= \{john\} \cup \{rahim\} \\
&= \{john, rahim\}
\end{aligned}$$

$$\begin{aligned}
&atToValueSet_{BLOG}(age) \\
&= atEntityToValue_{BLOG}(age, JOHN) \cup atEntityToValue_{BLOG}(age, RAHIM) \\
&= \{21, 32\}
\end{aligned}$$

$$\begin{aligned}
&atToValueSet_{BLOG}(postcode) \\
&= atEntityToValue_{BLOG}(postcode, JOHN) \cup atEntityToValue_{BLOG}(postcode, RAHIM) \\
&= \{G3\}
\end{aligned}$$

$$\begin{aligned}
& \mathit{atSetToValSet}_{BLOG}(\{\mathit{username}, \mathit{postcode}\}) \\
&= \mathit{atToValueSet}_{BLOG}(\mathit{username}) \cup \mathit{atToValueSet}_{BLOG}(\mathit{postcode}) \\
&= \mathit{atEntityToValue}_{BLOG}(\mathit{username}, \mathit{JOHN}) \cup \\
&\quad \mathit{atEntityToValue}_{BLOG}(\mathit{username}, \mathit{RAHIM}) \cup \\
&\quad \mathit{atEntityToValue}_{BLOG}(\mathit{postcode}, \mathit{JOHN}) \cup \\
&\quad \mathit{atEntityToValue}_{BLOG}(\mathit{postcode}, \mathit{RAHIM}) \\
&= \{\mathit{john}, \mathit{rahim}, \mathit{G3}\}
\end{aligned}$$

Next, functions that map attributes and values to the corresponding entities are defined.

Definition 3 *The function $\mathit{atValueToEntitySet}_d : A_d \times AV_d \to \mathcal{P}(E_d)$ maps each attribute and attribute value pair to the set of Entities which have that respective value for the attribute in domain d. Formally, for any $a \in A_d$ and $v \in AV_d$:*

$$\mathit{atValueToEntitySet}_d(a, v) = \{e \in E_d \mid \mathit{atEntityToValue}_d(a, e) = v\}$$

This function can be used to define another function $\mathit{atValueSetToEntitySet}_d : \mathcal{P}(A_d \times AV_d) \to \mathcal{P}(E_d)$ that retrieves the common set of entities for a set of attributes and values in domain d. That is, for any $B_d \in \mathcal{P}(A_d \times AV_d)$:

$$\mathit{atValueSetToEntitySet}_d(B_d) \overset{\text{def}}{=} \bigcap_{b \in B_d} \mathit{atValueToEntitySet}_d(b) \tag{3.3}$$

Returning to the example:

$$\begin{aligned}
\mathit{atValueToEntitySet}_{BLOG}((\mathit{username}, \mathit{rahim})) &= \{\mathit{RAHIM}\} \\
\mathit{atValueToEntitySet}_{BLOG}((\mathit{age}, 32)) &= \{\mathit{JOHN}\} \\
\mathit{atValueToEntitySet}_{BLOG}((\mathit{postcode}, G3)) &= \{\mathit{JOHN}, \mathit{RAHIM}\}
\end{aligned}$$

Now, using (3.3):

$$\begin{aligned}
& \mathit{atValueSetToEntitySet}_{BLOG}(\{(\mathit{username}, \mathit{rahim}), (\mathit{postcode}, G3)\}) \\
&= \mathit{atValueToEntitySet}_{BLOG}((\mathit{username}, \mathit{rahim})) \cap \\
&\quad \mathit{atValueToEntitySet}_{BLOG}((\mathit{postcode}, G3)) \\
&= \{\mathit{RAHIM}\} \cap \{\mathit{JOHN}, \mathit{RAHIM}\} \\
&= \{\mathit{RAHIM}\}
\end{aligned}$$

3.2.4 Identifiers, Partial Identifiers, Null Identifiers and Credentials

There are three different classes of attributes. One class can uniquely identify an entity, the second class can identify at least one entity and the third can identify no entities. These types are defined as the *Identifier*, *Partial Identifier* and *Null Identifier* respectively. Each type is explained below. Then, a special attribute called *Credential* is introduced that accompanies an identifier value during the identification and authentication process.

Identifier. An Identifier is an attribute whose value can be used to uniquely identify an entity within an application domain [71, 74]. An entity may have many attributes associated with itself, however, the most representative attribute whose value can be used to uniquely identify an entity can be defined as the Identifier of that entity. According to this definition, an Identifier is similar to the functionalities of a primary key in a relational database table.

There are many attributes in an application domain that may be used to uniquely identify an entity at a certain point in time. However, when more entities are added into the application domain, it may happen that one of the new entities has the same value for that attribute as an existing entity, thus violating the condition that its value can be used to uniquely identify an entity. To avoid unnecessary complications, each application domain considers one attribute as the identifier and ensures that its value can be used to uniquely identify an entity as long as the application domain exists. To determine the chosen identifier for a particular domain, the following function is defined.

Definition 4 *The function* $identifier : DOMAIN \rightarrow A_d$ *returns the chosen* identifier *such that for any domain* d*, if* $i = identifier(d)$*, then the following conditions are satisfied:*

- $atEntityToValue_d(i, e)$ *is defined for all* $e \in E_d$*;*
- $atEntityToValue_d(i, e_1) \neq atEntityToValue_d(i, e_2)$ *for all distinct* $e_1, e_2 \in E_d$.

The above conditions correspond to the following (required) properties when an attribute has been selected as the Identifier in a domain.

- Each entity of an application domain must have a value for the identifier;
- The value for the identifier can be used to uniquely identify any entity at any point in time as long as the application domain exists. More formally, if $i = identifier(d)$ and $v \in atToValueSet_d(i)$, then $|atValueToEntitySet_d(i, v)| = 1$.

These conditions can be enforced during the registration process (an essential step of Identity Management which will be explored later). More precisely, when a new entity is registered (added) in an application domain, it has to be ensured that the value of the identifier for that entity is not null and also unique with respect to the existing values of the identifier for other existing entities.

It is assumed that the *username* is the attribute that the exemplary system $BLOG$ has selected as the Identifier. Therefore, $identifier(BLOG) = username$.

Partial Identifier. When the value of an attribute identifies at least one entity within a specific application domain, the attribute is defined as a Partial Identifier [71]. An example of a partial identifier is *Surname*. Many people may share the same surname and therefore it is likely that the surname can identify more than one person. Formally, the set of partial identifiers can be defined in the following way:

Definition 5 *The set of* partial identifiers $PI_d \subseteq A_d \backslash \{identifier(d)\}$ *in domain* d *is such that* $pi \in PI_d$ *if and only if* $atEntityToValue(pi, e)$ *is defined for some* $e \in E_d$.

This means, for any $pi \in PI_d$ and $v \in atToValueSet_d(pi)$:

$$|atValueToEntitySet_d(pi, v)| \geq 1$$

Null Identifier. Null Identifiers are those attributes which do not have any value for all entities. The formal definition is given below:

Definition 6 *The set of* null identifiers $NI_d \subseteq A_d \backslash \{identifier(d)\}$ *in domain* d *is such that* $ni \in PI_d$ *if and only if* $atEntityToValue_d(ni, e)$ *is undefined for all* $e \in E_d$.

From the above definitions, it follows that the sets of Partial and Null Identifiers are disjoint, i.e. $PI_d \cap NI_d = \emptyset$.

Returning to the example, all other attributes apart from username in $BLOG$ will be considered as partial identifiers, that is, $PI_{BLOG} = \{age, postcode\}$. There is no null identifier in the example.

Next, functions to relate identifier, partial identifiers and entities are defined.

Definition 7 *The (injective) function* $idValueToEntity_d : AV_d \rightarrow E_d$ *maps a value of the identifier to the respective (unique) entity in domain* d. *That is, for any* $v \in atToValueSet_d(identifier(d))$:

$$idValueToEntity_d(v) = e$$

where $e \in E_d$ *is the unique entity for which* $atEntitToValue_d(identifier(d), e) = v$.

The $\mathit{idValueToEntity}$ function can be used to define a function, $\mathit{idValueToEntitySet}_d : \mathcal{P}(AV_d) \rightarrow \mathcal{P}(E_d)$ that retrieves the set of entities for a set of values of the identifier. More precisely, for $B_d \subseteq \mathit{atToValueSet}_d(\mathit{identifier}(d))$,

$$\mathit{idValueToEntitySet}_d(B_d) = \bigcup_{b \in B_d} \mathit{idValueToEntity}_d(b) \tag{3.4}$$

Furthermore, the following function is defined to find how each partial identifier can be mapped to a set of entities.

Definition 8 *Let $\mathit{piToEntitySet}_d : PI_d \times AV_d \rightarrow \mathcal{P}(E_d)$ be the function that maps each partial identifier and value to the corresponding entities. Formally, for any $pi \in PI_d$ and $v \in \mathit{atToValueSet}_d(pi)$:*

$$\mathit{piToEntitySet}_d(pi, v) = \{e \in E_d \mid \mathit{atEntityToValue}_d(pi, e) = v\}$$

Again returning to the example:

$$\begin{array}{rcl} \mathit{idValueToEntity}_{BLOG}(\mathit{john}) & = & \mathit{JOHN} \\ \mathit{idValueToEntity}_{BLOG}(\mathit{rahim}) & = & \mathit{RAHIM} \\ \mathit{piToEntitySet}_{BLOG}((\mathit{postcode}, \mathit{G3})) & = & \{JOHN, RAHIM\} \end{array}$$

Next, a function to relate the identifier to partial identifiers is introduced.

Definition 9 *Let $\mathit{idValueToPiValueSet}_d : AV_d \rightarrow \mathcal{P}(PI_d \times AV_d)$ be the function that maps a value of the identifier to the corresponding set of partial identifiers and their values in domain d. That is, for any $v \in \mathit{atToValueSet}_d(\mathit{identifier}(d))$, if $\mathit{idValueToEntity}_d(v) = e$, then $\mathit{idValueToPiValueSet}_d(v)$ is given by the set:*

$$\{(pi, v) \mid pi \in PI_d,\ \mathit{atEntityToValue}_d(pi, e) \textit{ is defined and equals } v\}$$

Following the example, $\mathit{idValueToPiValueSet}_{BLOG}(\mathit{john}) = \{(\mathit{age}, \mathit{31}), (\mathit{postcode}, \mathit{G3})\}$.

Figure 3.1 provides a graphical analogy of the relation between the user (entity), the identifier and the partial identifiers.

Credentials. In the real world, different physical attributes of a person are the attestation of who she is. Biometrics in the form of photograph, signature, or PIN (Personal Identification Number) are widely used credentials to be used with an ID or bank card. However, the situation is somewhat different in the online world as the physical face-to-face recognition is largely absent. To testify the identity of an entity online, another attribute called a *Credential*

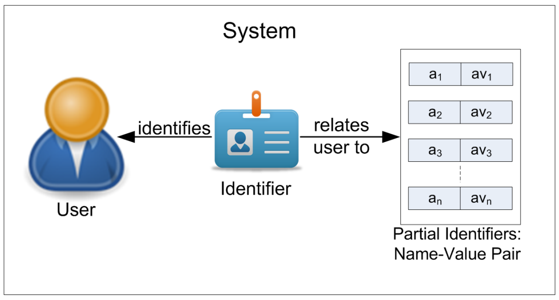


Figure 3.1: The relation between the user, identifier and the partial identifiers.

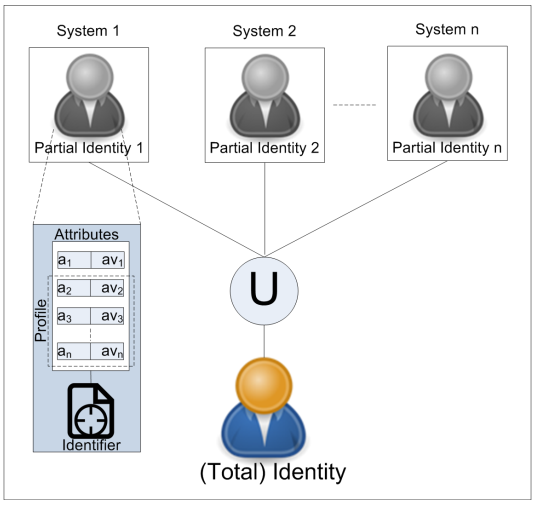


Figure 3.2: The (total) identity of an entity.

is needed. A generalised definition of Credential can be quoted from the Glossary of the Modinis Project: “A credential is a piece of information attesting to the integrity of certain stated facts” [63]. In this thesis, a credential will be considered as an attribute that accompanies an identifier and can be used to attest the authority of an entity over the supplied identifier (known as the authentication process, see Section 3.3.1). Such a credential is assumed to be secret to everyone except to the valid user. The simplest and also the weakest form of such credentials is a password. Attribute certificates are a common type of credential in the PKI (Public Key Infrastructure) System. For a more secure authentication process, biometrics such as fingerprints, face recognition, voice recognition, retina scan or secure hardware tokens such as the OTP (One Time Password) generator could be used.

Let, $cred_d$ be the attribute that holds the credential for each entity in domain d.

Definition 10 *Let* $checkCredential_d : (identifier(d) \times AV_d) \times (cred_d \times AV_d) \rightarrow \{\mathtt{true}, \mathtt{false}\}$ *be the function which given the input of an identifier and credential with their corresponding values returns* `true` *if the supplied identifier value equals the original identifier value and the credential value equals the original credential value or returns* `false` *otherwise.*

3.2.5 Partial Identity and Identity

The Partial Identity of an entity in an application domain can be defined as the sets of identifier and partial identifiers and their values within that domain [67]. The formal definition is as follows.

Definition 11 *For a domain d, the* partial identity *of an entity $e \in E_d$ within domain d,* *denoted* $partialIdentity_d^e$*, is given by:*

$$partialIdentity_d^e = \{(a, v) \mid a \in A_d,\ atEntityToValue_d(a, e) \textit{ is defined and equals } v\}$$

Informally, the $partialIdentity$ of an entity e is the set all attributes, including the identifier and partial identifiers, and their values for which $atEntityToValue$ is defined.

For the running example,

$$partialIdentity_{BLOG}^{JOHN} = \{(username, john), (age, 31), (postcode, G3)\}$$

Now, the identity of an entity can be defined as the union of all partial identities of that entity in all domains (Figure 3.2) [67]. Since the focus is on digital entities, the defined identity of an entity should be treated as its digital identity as well.

Definition 12 *For any entity* $e \in E$*, the* identity *of* e*, denoted* $identity^e$ *is given by:*

$$identity^e = \cup\{(d, partialIdentity_d^e) \mid d \in DOMAIN \textit{ and } e \in E_d\}\,.$$

Note that such a combined view of full identity of an entity only makes sense from the first person perspective of an entity (a user). From an organisational point of view, such a combined view would allow any organisation to gain unlimited control over the user data and hence is very lucrative, but potentially privacy-invasive as far as the user is concerned. A more privacy-friendly approach is to share a limited view of user data across organisational boundaries whenever needed. Such a limited view will be defined below (see the definition of Profile introduced in the next section). Sometimes, it may happen that two different application domains are combined (maybe due to merging of organisations) into a single domain, it may seem that two entities may end up with the same set of attributes and therefore the same partial identity. However, it is important to remember that, by definition, the identifier in an application domain cannot have duplicate values, which means two entities cannot have the same identifier value. It ultimately ensures that the partial identity sets of two entities in a single application domain are different since the partial identity sets will have at least one different attribute value. Therefore, during such merging of application domains, the system may need to update its sets of identifiers to ensure this consistency.

3.2.6 Profile

In a real world application setting, a user provides the identifier chosen for that particular application domain along with its value for the user during authentication process (described later) which can be mapped to a user using the $idValueToEntity$ function to see if the user exists into that particular application. Furthermore, the value of the identifier can be used to retrieve other attributes (partial identity) of the user using the $idValueToPiValueSet$ function.

The identifier and a subset of partial identifiers along with their values can be combined to define a profile of an entity for a particular application domain. Therefore, the formal definition is:

Definition 13 *For a domain d, the profile of an entity $e \in E_d$ is defined as follows:*

$$PROFILE_d^e = \{(identifier(d), v)\} \cup \mathcal{A}$$

where $v = atEntityToValue_d(i, e)$ and $\mathcal{A} \subseteq idValueToPiValueSet_d(v)$.

That is, the profile of an entity in an application domain is the combined set of the identifier and its respective value for the entity and a subset of related partial identifiers and their values. The profile of an entity e in domain d is the combined set of the identifier i and its respective value for the entity and a subset of partial identifiers and their values retrieved by the $idValueToPiValueSet$ functions. Only subsets of partial identifiers are included, instead of the complete set, to illustrate situations when only a few partial identifiers are sufficient for a service access scenario. More formally, $PROFILE_d^e \subseteq partialIdentity_d^e$.

The inclusion of the identifier in the profile deserves further attention. In case the $PROFILE$ is passed between different domains (which is often the case in traditional Identity Management Systems) and the identifier is permanent in nature, meaning it remains the same irrespective of the service scenario, it raises the question of privacy invasion since the user can be tracked or profiled in another domain with this permanent identifier. In such cases, an Anonym or a Pseudonym (see below) can be used instead of a permanent identifier. Here in $PROFILE$, the Identifier is treated in an abstract level to allow accommodating any type of Identifiers.

3.2.7 Anonymity, Anonym and Pseudonym

Anonymity will allow a user to access services anonymously. For that, the user needs to utilise an anonym or a pseudonym. An anonym is a non-identifiable attribute that cannot be associated with an entity and is used by an entity to access a service without revealing her identity. It can be used only once at an SP and using it more than once turns it into a pseudonym (see below) [75]. It is mainly used to preserve the privacy of the entity and thus provides anonymity on behalf of the entity.

On the other hand, a pseudonym is an arbitrary one-time identifier of an entity by which a certain action or a series of action can be linked to this specific entity without revealing the true identity of an entity (a user) [63] at an SP. In this sense, it allows a user to access a service anonymously and thus preserves privacy of the entity, like an anonym. However, unlike an anonym, it is not used for one-off scenarios. A pseudonym can be used as many times as the system allows, thereby allowing linkability and profiling in a specific domain and undermining linkability and profiling across multiple domains.

3.2.8 Digital Identity Model

Having formalised the required sets and functions to represent a digital identity, they can be combined into a unified model for digital identity as follows.

Definition 14 *The basic components of a* Digital Identity Model (DIM) *is the tuple:*

$$(E, DOMAIN, A, AV, identifier, \langle E_d, A_d, AV_d, cred_d, atEntityToValue_d, checkCredential_d\rangle)$$

where

- *E is the set of entities;*
- *$DOMAIN$ is the set of domains;*
- *A is the set of attributes;*
- *AV is the set of attribute values;*
- *$identifier : DOMAIN \to A$ which returns the chosen identifier for each domain;*
- *$E_d \subseteq E$, $A_d \subseteq A$ and $AV_d \subseteq AV$ are the sets of entities, attributes and attribute values in domain $d \in DOMAIN$;*
- *$cred_d \in A_d$ is a special attribute that holds the credential for each entity in the domain $d \in DOMAIN$;*
- *$atEntityToValue_d : A_d \times E_d \to AV_d$ which returns the value of an attribute for a specific entity in the domain $d \in DOMAIN$;*
- *$checkCredential_d : (identifier(d) \times AV_d) \times (cred_d \times AV_d) \to \{\mathtt{true}, \mathtt{false}\}$ which given the input of an identifier and credential with their corresponding values returns true if the identifier and credential match and false otherwise for each domain $c \in DOMAIN$.*

Using these components and the definitions, the following additional components of a DIM model can be defined:

- *$PI_d, NI_d \subseteq A_d$ are the sets of partial and null identifiers respectively in domain $d \in DOMAIN$;*
- *$partialIdentity^e_d \subseteq A_d \times AV_d$ is the partial identity of the entity $e \in E$ in domain $d \in DOMAIN$;*
- *$PROFILE^e_d \subseteq A_d \times AV_d$ is the profile of the entity $e \in E$ in domain $d \in DOMAIN$;*
- *$identity^e \subseteq \{(d, \mathcal{P}(A_d \times AV_d)) \mid d \in DOMAIN\}$ is the total identity of the entity $e \in E$ in all domains;*
- *$atValueToEntitySet_d : A_d \times AV_d \rightarrow \mathcal{P}(E_d)$ maps each attribute and attribute value pair to the set of Entities which have that respective value for the attribute in the domain $d \in DOMAIN$;*
- *$idValueToEntity_d : AV_d \rightarrow E_d$ maps a value of the identifier to the respective (unique) entity in the domain $d \in DOMAIN$;*
- *$piToEntitySet : PI_d \times AV_d \rightarrow \mathcal{P}(E_d)$ maps each partial identifier and value to the corresponding entities in the domain $d \in DOMAIN$;*
- *$idValueToPiValueSet_d : AV_d \rightarrow \mathcal{P}(PI_d \times AV_d)$ maps a value of an identifier to the corresponding set of partial identifiers and values in the domain $c \in DOMAIN$.*

Other functions that have been discussed can be induced or generated from the basic functions mentioned in the definition.

3.3 Identity Management

The ever increasing number of both online services and users that access each service leads to an even faster increase in partial identities. Therefore, the issue of managing these partial identities with identifiers and corresponding credentials is becoming a difficult task. Identity Management was proposed to facilitate the management of online identities [70]. Before different aspects of Identity Management are discussed, a definition of the term "Identity Management" would be useful. Surprisingly, a large variation of definitions of Identity Management exist, for example in [76, 63, 67, 77]. In [76], Identity Management is defined as "technologies and policies for representing and recognising entities with their digital identities" while [63] defines Identity Management as: "the managing of partial identities of entities, i.e., definition, designation and administration of identity attributes as well as choice of the partial identity to be (re-) used in a specific application domain". In [67], it is defined as: "Identity management means managing various partial identities (usually denoted by pseudonyms) of an individual person, i.e., administration of identity attributes including the development and choice of the partial identity and pseudonym to be (re-)used in a specific application domain or role".

These definitions are simple to understand; unfortunately they only focus on the identification, representation and management of identities and do not cover all the aspects (which will be described shortly) of Identity Management. The definition provided in the ITU-T X.1250 Recommendation [78] is the closest in spirit to the definition that I believe is appropriate. Unfortunately, it also fails to capture some other aspects of Identity Management that have been captured in previous definitions. It seems that none of these definitions captures the full notions of Identity Management, however, they can be combined to create a comprehensive definition that satisfies all aspects of Identity Management which is given below.

Definition 15 *Identity Management is a set of functions and capabilities (e.g., administration, management and maintenance, discovery, communication exchanges, correlation and binding, policy enforcement, authentication and assertions) used for:*

- *creation and management of identity information (e.g., identifiers, partial identifiers, credentials);*
- *assurance of identity information;*
- *assurance of the identity of an entity (e.g., users/subscribers, groups, user devices, organizations, network and service providers, network elements and objects, and virtual objects);*
- *selection of identity information to be used for authorisation and for service provisioning; and*
- *supporting business and security applications.*

A system that is used for managing the identity of a user is called an Identity Management System (IMS). Each IMS involves following classes of entities:

- **Clients/Users.** Clients/users receive services from service providers (see below). To receive a service, a user usually needs her partial identity with the identifier and the related credential from the corresponding identity provider (see below). Any entity can be a client, however, it is assumed that each client is a person and will be denoted by the set U.
- **Service Providers.** Service Providers (SPs) are organisations that provide services to users or to other service providers. Examples include mobile phone operators, web

service providers, etc. [79]. In its simplest form, an SP can be combined with an identity provider (see below) to act as a single entity. It is also known as the Relying Party. The notation SP will denote the set of service providers. It is easy to see that $SP \subseteq O$, where O represents the set of organisations defined in Section 3.2.1.

- **Identity Providers.** Identity providers (IdPs) are organisations that store and provide partial identities to allow users to receive services from an SP. The notation IDP will denote the set of identity providers. Similarly, $IDP \subseteq O$. Since, an SP may or may not be combined with an IdP, there is no requirement that $SP \cap IDP = \emptyset$.

3.3.1 Steps in Identity Management Systems

With the above mentioned parties, each IMS, generally, utilises a set of steps to allow any user to manage her identity in that specific application domain and to access services from an SP. The steps are *Registration*, *Identification and Authentication*, *Authorisation*, *Service Provisioning* and *De-registration*. This set of steps is also known as the Life-cycle of an Identity Management System. The domains in Identity Management consist of the set of identity and service providers, i.e. $DOMAIN = IDP \cup SP$. The subscript in the following operations specifies the domain (IdP or SP) in which each operation is valid. The steps of each IMS are discussed and modelled below.

- **Registration.** Registration is the initial step in which a user, who wants to access services provided by an SP, registers herself at the respective IdP by providing personal information. At this step, the user either chooses a unique value for the identifier of the service and a value for the related credential or the respective values of the identifier and the credential are generated automatically on her behalf by the IdP. The user may also provide values (data) for partial identifiers. At the end of this process, the set of entities, attribute values are updated in the IdP with the newly created/inserted data. The process can be modelled in the following way. For an $idp \in IDP$:

$$\begin{aligned} E'_{idp} &= \{e'\} \cup E_{idp} && \text{(add entity (user))} \\ AV'_{idp} &= \mathcal{AV}' \cup AV_{idp} && \text{(add identifier and other attribute values)} \end{aligned}$$

  Here, e' represents the newly created/joined entity and $\mathcal{AV}'$ represents the newly created/inserted data for that entity including the compulsory unique identifier value and optional values for other attributes. E'_{idp} and AV'_{idp} represent the updated sets of entities and attribute values respectively.

  The newly added information does not have any domain attached to it as it does not belong to any domain related to IdM before the user is registered. Once the user is registered, she then becomes a part of the registered domain.

After this registration process, the domain and range of several functions (whose domain and/or range contain E_{idp} or AV_{idp}) defined in Definition 14 will be updated to replace the sets E_{idp} and AV_{idp} with E'_{idp} and AV'_{idp} respectively to ensure that the newly inserted entity and data are considered while using those functions. The updated basic functions are as follows:

– $\mathit{atEntityToValue}_{idp} : A_{idp} \times E'_{idp} \to AV'_{idp}$. Assuming that the entity e' has added the attribute value $av' \in \mathcal{AV}'$ for the attribute $a \in A_{idp}$, $\mathit{atEntityToValue}_{idp}$ will return:

$$\mathit{atEntityToValue}_{idp}(a, e') = av';$$

– $\mathit{checkCredential}_{idp} : (i \times AV'_{idp}) \times (\mathit{cred}_{idp} \times AV'_{idp}) \to \{\texttt{true}, \texttt{false}\}$. Here, $i = \mathit{identifier}(idp)$. Assuming that the entity e' has added the attribute values $av'_1, av'_2 \in \mathcal{AV}'$ for $\mathit{identifier}_{idp}$ and cred_{idp} respectively and $i = \mathit{identifier}(idp)$, the function $\mathit{checkCredential}_{idp}$ will return:

$$\mathit{checkCredential}_{idp}((i, av_1), (\mathit{cred}_{idp}, av_2)) = \begin{cases} \texttt{true} & \text{if } av_1 = av'_1 \text{ and } av_2 = av'_2 \\ \texttt{false} & \text{otherwise.} \end{cases}$$

Following the previous examples, it is assumed that a new user named FARIDA has registered herself at the *Blog* System with attribute values of *farida, 25, G12* for attributes *username, age, postcode* respectively. Therefore, the updated sets of entities and attribute values are:

$$E'_{\text{BLOG}} = \{\mathit{JOHN}, \mathit{RAHIM}, \mathit{FARIDA}\} \text{ and}$$
$$AV'_{\text{BLOG}} = \{\mathit{john}, \mathit{rahim}, \mathit{farida}, \mathit{21}, \mathit{32}, \mathit{25}, \mathit{G3}, \mathit{G12}\}$$

Examples of a few functions using these updated sets are:

$$\begin{aligned} \mathit{atEntityToValue}_{BLOG}((\mathit{username}, \mathit{FARIDA})) &= \mathit{farida} \\ \mathit{atEntityToValue}_{BLOG}((\mathit{age}, \mathit{FARIDA})) &= \mathit{25} \\ \mathit{atEntityToValue}_{BLOG}((\mathit{postcode}, \mathit{FARIDA})) &= \mathit{G12} \\ \mathit{atToValueSet}_{BLOG}(\mathit{username}) &= \{\mathit{john}, \mathit{rahim}, \mathit{farida}\} \\ \mathit{atSetToValSet}_{BLOG}(\{\mathit{username}, \mathit{postcode}\}) &= \{\mathit{john}, \mathit{rahim}, \mathit{farida}, \mathit{G3}, \mathit{G12}\} \end{aligned}$$

Examples of other functions are similar and hence are not shown.

Many existing systems allow any user to create more than one account in that respective system using the same procedures described above. The user must choose a unique value for the identifier and can add one or more attribute values as described before during the registration process. It is important to understand that the system will treat each registered account as a separate account belonging to a separate user even though there might be multiple accounts by a single user. This particular property essentially makes it easier to capture this scenario using the developed model. Following the previous example, it is assumed that the user *(*JOHN) wants to register himself at the *Blog* System as a new user with attribute values of *jack, 26, G11* for attributes *username, age, postcode* respectively. This entity is denoted as $JOHN'$. Even though the system treats $JOHN$ and $JOHN'$ as separate entities, in reality they are the same entities. As these two accounts belong to two separate entities in the system, all previously discussed functions are applicable to them as well.

- **Identification and Authentication.** This step is required when, upon completing the registration step, a user wants to access the services offered by an SP. Before any service can be accessed, the user needs to be identified and authenticated. To be identified, an entity has to present the identifier. Quoting from the Modinis Document, "Identification is the process of using claimed or observed attributes of an entity to deduce who the entity is" [63]. That is, Identification is the process of identifying an entity by determining an association between the supplied identifier value and the entity. An example of an identification process in the real world is to find an association between a person and her claimed name. It is important to understand that with the absence of an identifier in the real world which can uniquely identify an entity in a global scale, the identification using names (or other identifiers) may result in multiple entities being identified. For example, when a name is used as an identifier, it may identify more than one entities in a specific domain (at offices or public places, in a city, etc.). On the other hand, the identification process associates only one entity with an identifier in an application domain (cf. Section 3.2.4). Authentication is the process of proving an association between an identifier (or an attribute) and an entity supplying the identifier value. To prove the association, the entity usually has to supply a credential value that accompanies the identifier value. Again from the Modinis document, "Authentication is the corroboration of a claimed set of attributes or facts with a specified, or understood, level of confidence" [63]. The process of identification is usually followed by the authentication process. In short, identification is the process of finding an association between an identifier value and the entity and authentication is the process of proving that the entity has the right to present that identifier value. In many systems, the process of identification and authentication are combined together into a single step.

  To model this step, a very simple algorithmic procedure, called *IdAuthenticaion*see Algorithm 1, can be defined that takes the input of an entity, identifier and credential for the IdP and the user supplied identifier and attribute values and either returns a successful result (indicated by the `true` value) if the entity (the user) can be identified with the

$idValueToEntity$ function and $checkCredential$ returns `true` for the supplied identifier and credential values or returns an unsuccessful result (indicated by the `false` value).

Algorithm 1 *IdAuthentication*$(e, i, v_1, v_2, cred_{idp})$. A simple algorithm to check if an entity can be identified and ultimately authenticated using supplied identifier and attribute values.

```
Input: $e \in E_{idp}, i = identifier(idp), v_1 \in atToValueSet_{idp}(i), v_2 \in atToValueSet_{idp}(cred_{idp})$
Output: $y \in \{\texttt{true}, \texttt{false}\}$
  if ($e = idValueToEntity_{idp}(v_1)$ and $checkCredential((i, v_1), (cred_{idp}, v_2)) = \texttt{true}$) then
    return true
  else
    return false
  end if
```

Once the user is authenticated, the corresponding entity is regarded as an authenticated entity. Formally, a set of authenticated entities is defined as follows:

$$AUTHN_{idp} = \{e \in E_{idp} \mid IdAuthentiation(e, i, v_1, v_2, cred_{idp}) = \texttt{true}\}$$

where,

$$\begin{aligned} i &= identifier(idp) \\ v_1 &= \text{value of the identifier } i \text{ of } e \\ v_2 &= \text{value of the credential of } e \end{aligned}$$

- **Authorisation.** Authorisation is the process to decide if a certain action on a specific resource is allowed by an entity in a specific domain based on an identifier value or partial identifier values [63]. The authorisation usually takes place in an SP. Two other sets are needed: $ACTION_{sp}$ and $RESOURCE_{sp}$ which signify the sets of actions and resources respectively in an SP.

  The authorisation process takes place only if an entity is already authenticated in an IdP. Therefore, an entity from the $AUTHN_{idp}$ set is needed. The SP needs to know the identifier and/or partial identifiers along with their values for that entity as well as the particular action the entity wants to perform on a specific resource. A user profile containing attributes and their values for that entity as well as the user requested action and resource in question are transferred to the SP using an Identity Protocol (see below). Upon receiving these values, the SP utilises an access control mechanism to authorise an entity. The mechanism may include an Access Control List (ACL) [80] or the Role Based Access Control (RBAC) model [45]. However, the generic mechanism is to have a list that determines which entity having which particular identifier or partial identifiers (or roles in RBAC) can perform which actions on what resources.

Let ACL be the set of tuples at an SP that defines which entity with which profiles can access what action on which resource.

$$ACL_{sp} \subseteq \bigcup_{e \in AUTHN_{idp}} \mathcal{P}(PROFILE^{e}_{idp}) \times ACTION_{sp} \times RESOURCE_{sp}\,.$$

Now, another very simple algorithmic procedure, called *CheckAuthorisation* (see Algorithm 2), can be defined that takes these inputs and either returns a successful result (indicated by the `true` value) if the tuple consisting of all inputs can be found in the ACL indicating that the entity with this profile is allowed to perform that action on that resource or returns an unsuccessful result (indicated by the `false` value) if the entity is not allowed to perform that action on that resource.

Algorithm 2 $CheckAuthorisation(ACL_{sp}, e, prof, act, res)$. A simple algorithm to check if an authenticated entity with its profile is authorised to perform a certain action on a specific resource.

```
Input: $ACL_{sp}, e \in AUTHN_{idp}, prof \subseteq \mathcal{P}(PROFILE^{e}_{idp}), act \in ACTION_{sp}, res \in RESOURCE_{sp}$
Output: $y \in \{\texttt{true}, \texttt{false}\}$
  if $((prof, act, res) \in ACL)$ then
    return true
  else
    return false
  end if
```

- **Service Provisioning.** Once the user is identified, authenticated and authorised to access a particular service, she can access the service. In terms of Identity Management, accessing a service is known as Service Provisioning. While a user accesses a service provided by an SP, the SP may opt to record a log trace that records the type of service access at a particular time on a particular date by an authenticated and authorised entity. In such log traces, time and date are important parts to ensure accountability and can be used to find out the history of a service accessed by an authenticated and authorised entity on a specific date and time.

  Among different services, one exemplary service provided by an IdP deserves further attention which allows an entity e to retrieve and update attribute values. Formally, when authorised to access this service in domain d, an entity e will have partial control over the basic DIM component represented by the $atEntityToValue_d$ (see Definition 2) function which will allow her to view and update entries (using the $atEntityToValue_d(a, e)$ function for all $a \in AV_c$). On the other hand, if the entity is not authenticated it will not be able to either view or update its attribute values, meaning formally the entity has neither access nor control over the component $atEntityToValue_d$.

- **De-registration.** The final step in the Identity Management life-cycle is the De-registration process which allows any user to de-register from an IdP that the user does not wish to access any more. The de-registration process usually removes the association between an entity and identifier and also removes the respective entity by deleting the partial identity from the system. In such, it is the reverse process of registration.

  The De-registration process is modelled in the following way. For any, $idp \in IDP$:

  $$\begin{aligned} AV'_{idp} &= AV_{idp} \setminus \mathcal{AV}' && \text{(remove data including identifier value)} \\ E'_{idp} &= E_{idp} \setminus \{e'\} && \text{(remove entity)} \end{aligned}$$

  Here, E'_{idp} and AV'_{idp} represent the updated set of entities and attribute values respectively in idp.

  As with the registration process, de-registration also requires the functions (whose domain and/or range contain E_{idp} or AV_{idp}) in Definition 14 are updated to ensure that the deleted entity and corresponding data are not considered.

  Following the previous examples, it is assumed that *JOHN* has de-registered himself from the *Blog* System. Therefore, the updated sets of entities and attribute values are:

  $$E'_{\text{BLOG}} = \{RAHIM, FARIDA\} \text{ and } AV'_{\text{BLOG}} = \{rahim, farida, 21, 25, G3, G12\}$$

  Examples of a few functions using these updated sets are:

  $$\begin{aligned} atToValueSet_c(username) &= \{rahim, farida\} \\ atSetToValSet_c(\{username, postcode\}) &= \{rahim, farida, G3, G12\} \end{aligned}$$

  Examples of other functions will be similar and hence are not shown.

3.3.2 Identity Protocol

From the discussion above, some of the above operations (Registration, Identification, Authentication and De-registration) take place at an IdP and the rest (Authorisation and Service Provisioning) take place at an SP. Depending on the domain, each of these operations can be divided into two categories: the operations that take place in an IdP and will be denoted by IDP_OP and the operations that take place in an SP and will be denoted by SP_OP. The notations REG, $IDFN$, $AUTHN$, $AUTHRSN$, SRV and $DEREG$ will denote Registration, Identification, Authentication, Authorisation, Service Provisioning and De-registration respectively. So formally,

$$IDP\_OP = \{REG, IDFN, AUTHN, DEREG\}$$

$$SP\_OP = \{SRV, AUTHRSN\}$$

The notation $IDMOP$ (short for, Identity Management Operations) will denote the combined set of these five steps. So,

$$IDM\_OP = IDP\_OP \cup SP\_OP$$

IDP_OP_d, SP_OP_d and IDM_OP_d will be used to denote the respective operations in domain d.

Readers may notice a slight discrepancy in the Authorisation step where sets from different domains are put together in Algorithm 2. For example, the subset of the profile of authenticated entities ($prof \in \mathcal{P}(PROFILE^{e}_{idp})$, where $e \in AUTHN_{idp}$) from the domain of an IdP are combined together with sets of actions ($ACTION_{sp}$), resources ($RESOURCE_{sp}$) and ACL (ACL_{sp}) from the domain of an SP to define the algorithm 2. This is to signify the realistic scenario in which the SP needs to utilise some information (authenticated entity and her identifier, partial identifiers and possibly data item at the IdP) received from the IdP during the authorisation phase. In reality, a mechanism called the *Identity Protocol* is defined to enable this transfer and is part of every Identity Management System. Each Protocol consists of two sub-protocols: Request Protocol and Response Protocol which are explained below.

Request Protocol. It flows from an SP to an IdP via a user. The IdP is usually selected by the user, however, sometimes, the SP can select the IdP on behalf of the user. In a request protocol, the SP asks the IdP to authenticate the user and requests the authentication information (the profile of a user).

Response Protocol. This is initiated in response to the Request Protocol and flows from an IdP to an SP. It consists of the profile (the $PROFILE_{idp}$ set) from the IdP comprising a user's identifier (usually an anonym or pseudonym), partial identifiers and optionally other data. The profile can be automatically selected by the IdP based on a policy defined by the user or the user can explicitly select the attributes and data for the profile. Upon receiving the response, the SP extracts the required information from the response and initiates the authorisation process.

Different Identity Management Systems have different Identity Protocols. Examples of widely used identity protocols include SAML [5], OAuth [7], OpenID [6] and so on. It is impractical to try modelling an identity protocol since each protocol uses completely different mechanisms. Hence, the notation $PROTOCOL$ will denote the underlying protocol in an abstract notion to establish the fact that the system model has an unspecified mechanism to pass on identity information whenever required.

3.3.3 The Basic Identity Management Model

The Basic Identity Management (BIM) Model is based on the DIM and consists of SP, IDP, the five steps and the Protocol along with sets, functions and relations defined in them that utilise the DIM.

3.3.4 Trust Issues in Identity Management

The issue of trust is one of the central issues in Identity Management as the parties involved need to trust each other to some certain extent. An IdP and an SP need to trust each other regarding the released attributes. On the other hand, a user needs to trust both an IdP and an SP regarding different scopes. Such trust issues will be explored in greater details in Chapters 6, 7 and 8.

3.4 Applications

In this section, four possible application scenarios will be explored using the developed model. In the first scenario, the effect of multiple partial identities of a user will be analysed. In the second scenario, the issue of data ownership and controllability will be explored. In the third scenario, the cause of identity theft will be discussed. In the last scenario, how the model can be used to characterise the behaviour of three popular identity management models will be shown.

3.4.1 Effect of Multiple Partial Identities

The typical use of online services requires any user to register in multiple identity (or service) providers by providing different attribute values during the registration process as mentioned in previous sections. It means that the user ends up with multiple partial identities scattered across multiple providers. If $n_{e,d}$ denotes the number of attribute values provided by the entity (i.e. user) e in a domain (i.e. provider) d, then:

$$n_{e,d} = |partialIdentity^e_d| \,.$$

Supposing $d_1, \ldots, d_m$ are all domains (providers) in which e has partial identities, the total number of attributes (denoted by N_e) can be derived for the user e as:

$$N_e = \sum_{i=1}^{m} n_{e,d_i} \, .$$

The value N_e signifies the number of attributes that need to be managed by the user e. Ideally, N_e should have a small value which will allow users to manage their attributes in a convenient way. Unfortunately, with the proliferation of novel online services requiring users to register to access those services, the value of N_e keeps increasing. One of the central focuses in Identity Management research is to reduce N_e. There are two ways N_e can be reduced: i) by lowering $n_{e,d}$ in each domain d, i.e., only storing smaller number of attributes at each provider and ii) by lowering m (the number of providers) so that attributes are stored at a small number of providers. The second option is more suitable for two reasons.

- Firstly, it may not be always possible to know beforehand which attributes might be required/requested by an SP later on; hence an IdP might prefer to store as many attributes as possible.
- Secondly, when attributes are scattered across many IdPs, it becomes increasingly difficult for a user to effectively manage attributes stored in those IdPs.

Also, the same attributes may be stored in multiple places resulting in unnecessary redundancy. Minimising m would enable a user to manage attributes efficiently and hence is the focus of all existing IMSs. In addition, in many IMSs, a legal contract between an IdP and an SP dictates the handling of user attributes between themselves. However, there may not be any legal contract between the user and the IdP and therefore, the handling of attributes may be governed by the respective Terms and Conditions. The absence of any legal contract between a user and an IdP means that the handling of user attributes is only bound by a trust assumption where the user can only hope that the respective party will honour the imposed trust [9].

Ideally, m=1 would be the most suitable choice as far as a user is concerned. It means that there is only one IdP storing all attributes of the user and providing them to an SP. With this goal, Microsoft introduced the Passport System to become the IdP of the Internet [81]. However, the attempt failed and the reason behind this failure is that the inclusion of the Passport in each interaction between users and SPs were not properly justified and users were not very confident and comfortable about a third party holding all their attributes [1]. Since then, it has been predicted and envisioned that there will exist more than one IdP with their specific purposes. For example, a bank IdP can be used for financial activities, the Governmental IdP for accessing Governmental services, and other IdPs for other services.

All this means is that the value of m will always be more than 1. The optimal value of m that will enable users to manage their attributes in the most efficient manner is yet to be found and might vary from one person to another.

In order to reduce the value of m, it is important to analyse the issue of trustworthiness of an IdP. By trustworthiness of an IdP it is meant the level of trust another entity has over that IdP. Even though trust is a subjective opinion, one IdP may be considered as a highly trusted entity whereas another IdP may be considered as an untrusted (or a low trusted) entity. There are several factors that are used to determine the trustworthiness of an IdP. One of the most crucial factors is the registration procedure that any user needs to go through to register into an IdP. The highly trusted IdP, such as the IdP established by the Government, financial institutes (e.g. banks) or educational institutes such as universities, will go through a rigorous registration procedure before any user is registered. For example, one has to be a citizen/resident of a country to register for a Governmental IdP and one has to prove her identity by showing a passport or a driving licence to register for a bank IdP. On the other hand, any user can register herself easily online without proving her identity by simply filling in a web form for several larger social-network based IdPs such as Facebook, Google, Twitter, etc. Users do not need to prove their identities and can simply fill in the web form with superficial or even random meaningless values. Hence, such an IdP can be regarded as a low trusted or even an untrusted IdP. Since it is difficult to establish and maintain a highly trusted IdP, the number of highly trusted IdPs will be always low. Thus, the higher value of m is largely contributed by the large number of low trusted and untrusted online IdPs (Figure 3.3). This is also evident in the current setting of online services where a user needs to register for any novel online services which increases the space of such low trusted and untrusted IdPs. If a mechanism can be found to reduce the number of such IdPs, it can significantly reduce the value of m.

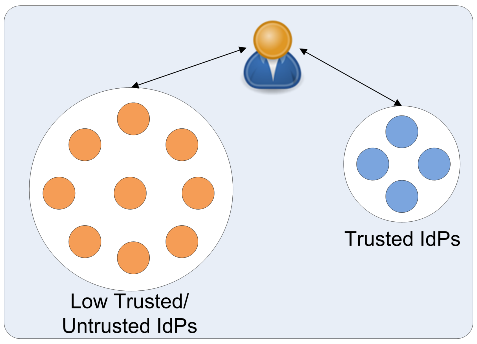


Figure 3.3: Current identity space.

3.4.2 Data Ownership and Controllability

The term *Data Ownership* is used to represent the situation of determining which entity actually has ownership over a set of data. With the ever-increasing popularity of online services and the increase of partial identities stored in those service providers, the issue of data ownership has gained much attention in recent years [82]. “Who owns our data?” - seems a simple question; the answer to it, unfortunately, is not straightforward and has been highly debated [82]. For example, the authors in [82] argue that even though a user is the owner of her personally identifiable information (i.e. partial identity), the ownership expands to other entities as the user registers into different providers and these entities (providers) become the owner of such information as well. However, I contradict this view and argue that a user is the ultimate owner of her partial identity, no matter where that partial identity is registered. Registering to a specific provider with her partial identity merely enables the user to access a particular set of services from that provider or using that provider. In such, the registration process enables a user to access such services from a provider, in no way it makes the provider the owner of the partial identity. This view is also resonant with what Facebook defines in their terms and conditions: “You own all of the content and information you post on Facebook...” [83]. Next, a function that returns the owner of a partial identity is defined.

Definition 16 *The function $ownerIs_d : partialIdentity_d \rightarrow \mathcal{P}(E)$ returns the set of entities containing the owners of the registered partial identity in the domain d.*

As per the view, $ownerIs_d(partialIdentity^e_d) = \{e\}$ and $|ownerIs_d(partialIdentity^e_d)| = 1$ for any entity $e \in E_d$ in domain d, that is, there is only one owner of a partial identity and the $ownerIs$ function returns a singleton set.

Next, the topic of *Data Controllability* is explored which is used to represent the situation of determining which entity has control over a set of data. By *control* it is meant that an entity can have read or write access on the partial identity of the same or another entity in a valid way and can utilise such access for her benefit in one way or another. When an entity is the owner of the partial identity, she should have full control (read and write access) over her partial identity. This will allow her to access a particular service from a provider using her partial identity as well as to update her partial identity when it is required. However, when an entity is not the owner of a partial identity, allowing the entity with the full control of that partial identity can have negative consequences. To illustrate this scenario, a function that returns the entity that controls the partial identity of an entity is defined.

Definition 17 *The function $hasFullControl_d : partialIdentity_d \rightarrow \mathcal{P}(E)$ returns the set of entities that have full control over the partial identity in the domain d.*

For the sake of the security and privacy of an entity, it is desirable that only the owner of a partial identity should have the full control over the respective partial identity. That is, $ownerIs_d(partialIdentity^e_d) = hasFullControl_d(partialIdentity^e_d)$, where there is only one owner of a partial identity, $|ownerIs_d(partialIdentity^e_d)| = 1$. Unfortunately, the current setting of online services allows a provider to have unprecedented control over the partial identity of a registered user. This allows the provider to mine the user data and build up a profile for targeted advertisements, which ultimately is the main source of their revenue [84]. This means that the function returns more than one entity for a particular partial identity: $hasFullControl_d(partialIdentity^e_d) = \{e, e'\}$ where e' signifies the provider. In such, $ownerIs_d \subseteq hasFullControl_d$ instead of the desired condition of $ownerIs_d = hasFullControl_d$. This has following serious negative consequences.

- Users have less control over their own data.
- Providers are able to access the user data and can share it with any third party without the permission of the user. The user has no way of knowing how their data are being abused.

The revelations of Edward Snowden, an American whistle-blower, revealed an example of such negative consequences [85]. According to him, National Security Agency (NSA) of the USA has been performing a massive-scale online surveillance over the user of several big online service providers such as Facebook, Google and Yahoo. The NSA has forced these providers to release their user data for the purpose of continuous surveillance. This has been possible because the provider has unlimited control over their user data. In this thesis, the mechanisms to counteract these issues will be investigated.

3.4.3 Analysing Identity Theft

Identity theft is one of the major online fraudulent activities and with the proliferation of online services it has been a source of huge financial losses in recent years. It has been reported by Javelin Strategy & Research that the financial loss due to the identity theft reached nearly \$21 billion only in the US in 2013 [86]. The prediction is that such attacks might grow in numbers in the upcoming years. The identity theft has been a difficult issue to tackle and many attempts have been proved fruitless so far. Before using the model to analyse the root cause of identity theft, a suitable textual definition of identity theft is explored.

The term *Identity Theft* has been defined in many ways in the literature [87, 88]. The definition provided by Koops et al. [89] is preferred: "*Identity 'theft' is fraud or another unlawful activity where the identity of an existing person is used as a target or principal tool without* *that person's consent*". The person whose identity is being abused is the *Victim* and the person who abuses the victim's identity is the *Attacker*. This definition is too literal, there is no mention if the term *identity* in the definition means the total identity or the partial identity in a particular domain.

For the time being, it is assumed that it refers to the partial identity of an entity e in a particular domain d. Now, to steal the entity's partial identity in a domain, an attacker needs to get hold of all the attribute values provided by the victim in the domain d which is represented by set $partialIdentity_d^e$. This set can be generated if one has access to the basic DIM component $atEntitToValue_d$ (see Definition 2). As discussed in the *Service Provisioning* step of Section 3.3.1, an attacker will have sufficient access to this component to obtain e's profile, if the attacker is identified, authenticated and authorised as e in domain d. Furthermore, as explained in the *Identification and Authentication* step of Section 3.3.1, achieving this is dependent on the attacker getting hold of the values of both the identifier (e.g. a username) and credential (e.g. a password) of e for domain d. There are numerous ways an attacker can get hold of these two pieces of information such as using simple or complex social engineering techniques, having access to a piece of paper containing such information in written format, deploying advanced phishing techniques, by guessing or brute forcing a password [4, 90].

There have been many attempts to address this problem, which mostly focus on hardening the credential (e.g. strong passwords which are difficult to guess or brute force, biometrics which are difficult to forge or hardware tokens to generate one time password), using secure hardware (e.g. smartcard) or using multi-channel authentication (e.g. using a fixed password at first and then provide a secondary password received in the mobile phone). Nonetheless, it does not matter how strong the credential is; if the attacker gets hold of it, there are always chances of identity theft. It seems that the root cause might be something else. To analyse that, the following function is introduced.

Definition 18 *The function* $suppliedBy_d : (ident(d) \times AV_d) \times (cred_d \times AV_d) \to E$ *returns the entity who has supplied the value of an identifier and credential pair in the domain* d.

Note that, $suppliedBy$ can return an entity who may not exist in that domain.

For any $v \in atToValueSet_d(ident(d))$ and $v' \in atToValueSet_d(cred_d)$, current online services assume that :

$$idValueToEntity_d(v) = suppliedBy_d((ident(d), v), (cred_d, v')) .$$

However, in case of an identity theft, there is:

$$idValueToEntity_d(v) \neq suppliedBy_d((ident(d), v), (cred_d, v')) .$$

It may therefore appear that to tackle the identity theft problem it is just needed to introduce a function having the capability of Definition 18. Unfortunately, developing and deploying such a function can be extremely difficult, if not impossible, with the current structure of the Internet where anybody can be "anything".

3.4.4 Modelling Popular Identity Management Models

In this section, it will be analysed how the Basic Identity Management Model can be extended to characterise the behaviour of three popular Identity Management models: Isolated User Identity (SILO) Model, Federated User Identity (FED) Model and Open Unfederated Identity (OUI) Model. Since an IMS can effectively consist of several identity domains (of IdPs and SPs), the notation $DOMAIN_{m_i}$ will denote a single identity domain i for an IdM Model m whereas the notation $DOMAIN_m$ will denote the combined identity domain consisting of several single identity domains i for an IdM Model m.

Additionally, the notation IDP_m and SP_m will be used to denote sets of IdPs and SPs respectively in a specific Identity Management Model m. Before starting, a function will be defined to determine the set of SPs shared by a single IdP in a specific IdM Model.

Definition 19 *The function $sharedBy_m : IDP_m \rightarrow \mathcal{P}(SP_m)$ maps an IdP to the set of SPs that utilise the service of that IdP for Identity Management operations in a specific Identity Management Model m.*

Isolated User Identity (SILO) Model

The Isolated User Identity model, also known as the SILO model, represents the most common and the simplest IdM model [4]. There are only two parties involved in the scenario: service providers and users. Each service provider has its own identity domain, i.e. has its own IdP and manages its namespace locally. Each SP provides the partial identity to its users who wish to receive its services. Even though this model represents the simplest form of IdM from the service provider's perspective, it does not allow interoperability, meaning, users from one service provider cannot access services from other service providers. This can create an immense burden for the users as they need to remember many identifiers (e.g. user-ids) and related credentials (e.g. passwords) and soon those become unmanageable when the number of such SPs starts to increase. Currently, all major and leading online service providers such as Amazon, EBay, Facebook and different banks follow this model. Figure 3.4 illustrates the SILO model in which there are three SPs.

In this model, for some $n \in \mathbb{N}$:

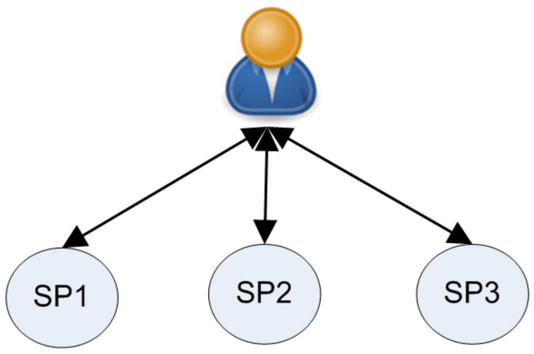


Figure 3.4: SILO Model.

- $\mathit{IDP}_{\mathit{SILO}_i} = \{\mathit{idp}_i\} = \{\mathit{sp}_i\} = \mathit{SP}_{\mathit{SILO}_i}$ for all $1 \leq i \leq n$;
- $\mathit{DOMAIN}_{\mathit{SILO}_i} = \mathit{IDP}_{\mathit{SILO}_i}$ for all $1 \leq i \leq n$;
- $\mathit{DOMAIN}_{\mathit{SILO}} = \bigcup_{i=1}^{n} \mathit{IDP}_{\mathit{SILO}_i}$.

The properties above signify that each single identity domain in the SILO Model, denoted SILO_i, consists of a singleton set of IdPs containing a single entity idp. The set of IdPs can also be regarded as the set of SPs since they are analogous to each other. Additionally, $\mathit{DOMAIN}_{\mathit{SILO}}$ represents the union of several single identity domains of the SILO Model. Figure 3.5 illustrates the idea of the identity domain in the SILO Model when $n = 4$. Each inner rounded rectangle represents a single SILO identity domain which consists of a singleton set of IdP ($\mathit{IDP}_{\mathit{SILO}_1}, \ldots, \mathit{IDP}_{\mathit{SILO}_4}$) containing only one IdP ($\mathit{IdP}_1, \ldots, \mathit{IdP}_4$). The vertical line in each inner rounded rectangle signifies the property that the singleton set of IdP can also be regarded as the singleton set of SPs ($\mathit{SP}_{\mathit{SILO}_1}, \ldots, \mathit{SP}_{\mathit{SILO}_4}$) each containing only one SP ($\mathit{SP}_1, \ldots, \mathit{SP}_4$). The outer rounded rectangle represents the union of several single identity domains. Moreover, for any $1 \leq i \neq j \leq n$:

- $|\mathit{sharedBy}_{\mathit{SILO}_i}(\mathit{idp}_i)| = 1$ for all $\mathit{idp}_i \in \mathit{IDP}_{\mathit{SILO}_i}$;
- $\mathit{sharedBy}_{\mathit{SILO}_i}(\mathit{idp}_i) \cap \mathit{sharedBy}_{\mathit{SILO}_j}(\mathit{idp}_j) = \emptyset$ for all $\mathit{idp}_i \in \mathit{IDP}_{\mathit{SILO}_i}$ and $\mathit{idp}_j \in \mathit{IDP}_{\mathit{SILO}_j}$.

That is, each IdP is shared by only one SP and two IdPs belonging to different single identity domains will not be shared by the same entity. Identity Management operations in one single domain (inside one IdP or SP) is completely separate from the operations in other domains. Lack of connection between any inner rounded rectangle signifies this fact and it can be modelled in the following way, for any distinct $\mathit{DOMAIN}_{\mathit{SILO}_i}, \mathit{DOMAIN}_{\mathit{SILO}_j} \in \mathit{DOMAIN}_{\mathit{SILO}}$:

$$\mathit{IDMOP}_{\mathit{DOMAIN}_{\mathit{SILO}_i}} \cap \mathit{IDMOP}_{\mathit{DOMAIN}_{\mathit{SILO}_j}} = \emptyset\,.$$

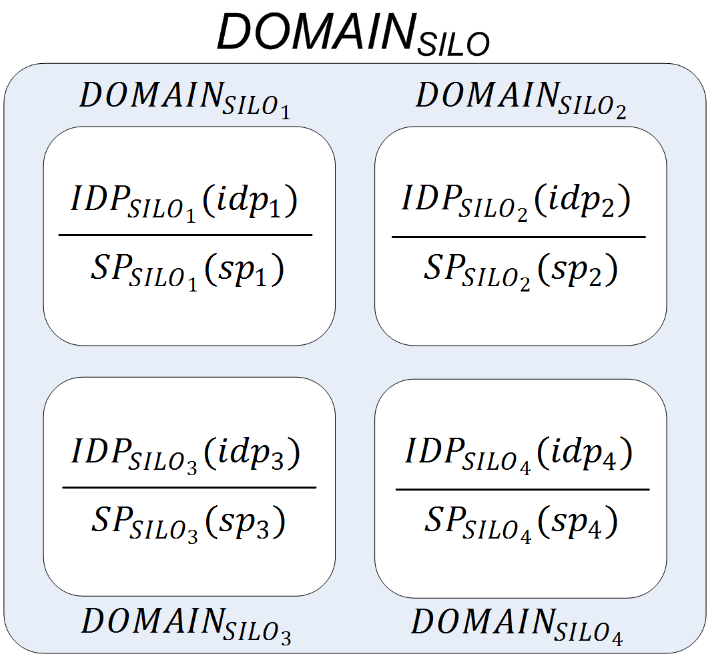


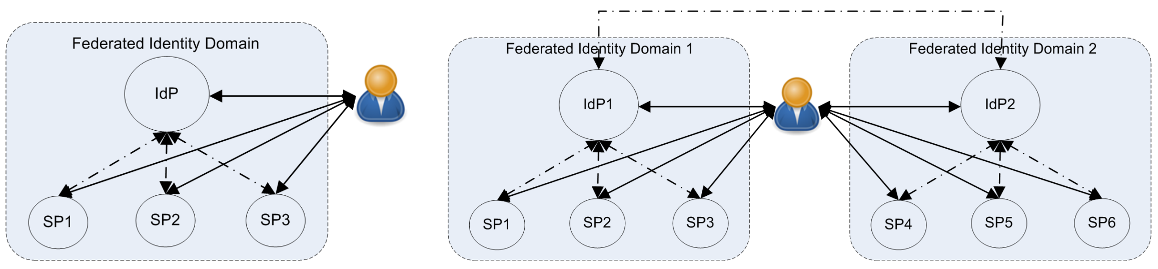
Figure 3.5: Representation of the SILO Model.

Federated User Identity (FED) Model

The Federated Identity Model may consist of one or more IdPs and one or more SPs [4]. Each SP may or may not have its own IdP while each IdP has their own identity domain. Thus, this model unifies different identity domains to create a unified larger identity domain. Such an identity domain is commonly known as the Federated Domain, Identity Federation or simply Federation and all entities (IdPs and SPs) are said to be a part of the so-called Circle of Trust (CoT). The IdPs provide the identifier and the related credential to the users. To access a service, the user authenticates herself to an IdP and once authenticated, she is redirected to a service provider's domain to access the service. Once a user is authenticated at the federated domain she can access the service of all SPs that are part of the same federated domain using a mechanism commonly known as Single Sign On (SSO). The Google Service is an example of this model consisting of one Google IdP and many Google SPs such as Gmail, Google Drive, Calender, Google Scholar, etc. Logging into the Google IdP allows a user to access all Google services without any further logins using the SSO mechanism. Another example of this model can be given in the setting of a university. A university may have one federated domain consisting of one central IdP and a number of SPs such as Library, Email, Printing Service, Result and Moodle. A user can access all these services by just one initial login using the SSO mechanism. This model has gained considerable attention in the last few years and is one of the most widely used models in the setting of the higher educational establishments and Governmental services. SAML [5] is one of the most matured and widely used protocols in such settings.

The Federated Identity Management (FIM) is based on the concept of this model [81]. It is also known as Federated Identities or Federation of Identities. In the ITU-T X.1250 recommendation, a federation is defined simply as "*An association of users, service providers and identity providers*" [78]. In other words, a federation with respect to the Identity Management is a business model in which a group of two or more trusted parties legally bind themselves with a business and technical contract [81, 91]. It allows a user to access restricted resources seamlessly and securely from other partners in different Identity Domains using the SSO mechanism.

A federated identity domain can be formed consisting of only one IdP in an identity domain and more than one SP with each SP residing in a separate identity domain (Type 1 in Figure 3.6). Several identity domains can be combined to form a larger identity domain where each smaller federated domain is of Type 1 (Type 2 in Figure 3.6). The dashed boundary in each figure signifies the federated identity domain, that is a single CoT and each solid circle represents a separate identity domain. That is, IdP, SP1, SP2 and SP3 in Figure 3.6(a) and IdP1, IdP2 and all SPs in Figure 3.6(b) are part of a single CoT respectively. Federated Identity Domain 1 and 2 in Figure 1(b) represent a combined Type 2 federated domain.


Federated Identity Domain
IdP
SP1
SP2
SP3
Federated Identity Domain 1
IdP1
SP1
SP2
SP3
Federated Identity Domain 2
IdP2
SP4
SP5
SP6

(a) Type 1. (b) Type 2.

Figure 3.6: Federated Identity Domain.

The FIM offers a number of advantages in different aspects to different stakeholders [81, 92]:

- The separation of duties among different organisations is the main advantage of FIM. An IdP only concentrates on managing partial identities and providing identity information to an SP while an SP concentrates only on providing services to authenticated users based on the identity information. Such separation allows the SP to offload the associated cost of managing and maintaining user identities to the trusted IdP.

- FIM provides scalability in the sense that it allows an SP to offer its services not only to its own user-bases, but also to users from other SPs and thus maximising the number of users using the same infrastructure.

- It is also attractive from an IdP's perspective as it allows the IdP to maintain an association with end users and enables them to access an array of diversified services with the same identity.

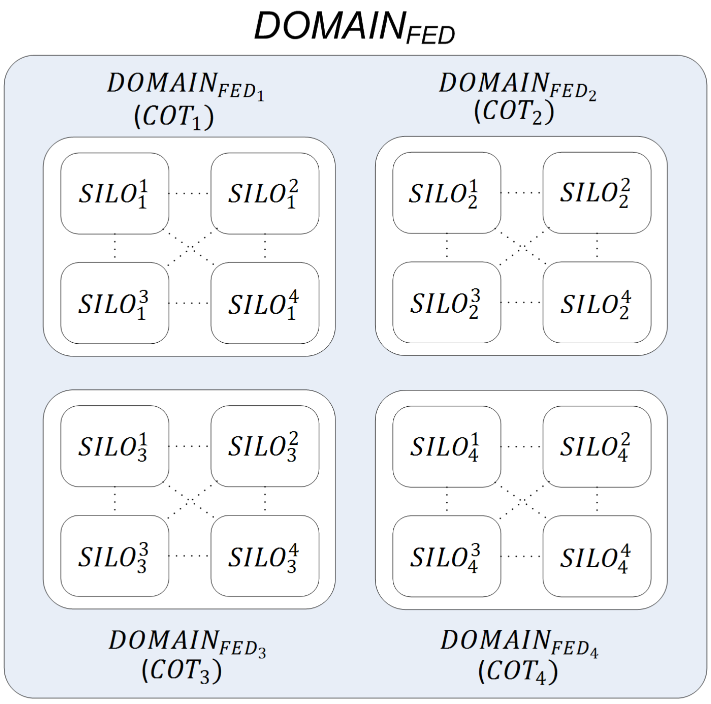
- FIM utilises standardised approach to ensure the improved security and privacy of users and make sure that the identity information is minimally propagated and thereby reduces the risk of identity theft.
- The CoT can easily be expanded by integrating new applications and service partners into the federation. It takes a minimal effort for any new partner to be a part of that circle as they need not to worry about managing the user-base.
- It allows users to leverage the SSO facility to avail services from different providers without any further logins. This also reduces the need to maintain partial identities in different domains. Thus, SSO improves efficiency and enhances user experience across different domains.
- FIM also alleviates users from remembering different identifiers and their related credentials as only one partial identity at an IdP is sufficient to access different services from all SPs in a federation. In this sense, it also increases security imperatively as only a very few accounts are managed by a user; a strong password can be chosen and easily remembered.

The issue of trust is a fundamental concept in FIM as different participating organisations need to trust each other inside the federation in a sufficient level to allow them to exchange user information and trust those information. The issue of trust with respect to FIM in different aspects will be explored in details in Chapters 6, 7 and 8.

Mathematically, this model is: $DOMAIN_{FED} = \{SILO_i^j \mid 1{\le}i{\le}n \text{ and } 1{\le}j{\le}n_i\}$, where $n \in \mathbb{N}$ and $n_i \in \mathbb{N}$ for all $1 \le i \le n$ such that:

- $DOMAIN_{FED_i} = COT_i = \bigcup_{j=1}^{n_i} DOMAIN_{SILO_i^j}$ for all $1 \le i \le n$;
- $|sharedBy_{FED_i}(idp)| = n_i$ for all $idp \in COT_i$ and $1 \le i \le n$;
- $DOMAIN_{FED} = \bigcup_{i=1}^{n} COT_i$.

The first property signifies that each single federated domain or each single CoT (COT_i), in essence, consists of several SILO identity domains (denoted as $SILO_i$). However, unlike the SILO model, this model offers interoperability among different SILO domains, meaning that identity management operations such as identification and authentication that take place inside an IdP are shared among several SPs inside the same CoT and is indicated by the second property above. To ensure interoperability, an agreement is signed among participating organisations to create a single CoT. The combined federated identity domain ($DOMAIN_{FED}$)



$DOMAIN_{FED}$
$DOMAIN_{FED_1}$ (COT_1)
$DOMAIN_{FED_2}$ (COT_2)
$DOMAIN_{FED_3}$ (COT_3)
$DOMAIN_{FED_4}$ (COT_4)
$SILO_1^1$ $SILO_1^2$ $SILO_1^3$ $SILO_1^4$
$SILO_2^1$ $SILO_2^2$ $SILO_2^3$ $SILO_2^4$
$SILO_3^1$ $SILO_3^2$ $SILO_3^3$ $SILO_3^4$
$SILO_4^1$ $SILO_4^2$ $SILO_4^3$ $SILO_4^4$

Figure 3.7: Representation of the Federated Model.

consists of several single CoTs (the third property). Figure 3.7 illustrates the Federated Identity Domain. Now, there are three layers of identity domains. In the innermost layer, there are several SILO identity domains represented by the innermost rounded rectangle. Each of these SILO domain has the similar functionality as defined in SILO Model with one exception that they have been federated using a contract to share the identity information of their users among each other. The dotted line between them signifies this fact. These federated SILO domains are combined to create the CoT represented by the middle rounded rectangles. The outer rounded rectangle represents the union of several CoTs.

In this model, Identity Management operations are totally different from one CoT to another CoT. That is, for any $1 \leq i \neq j \leq n$:

$$IDMOP_{COT_i} \cap IDMOP_{COT_j} = \emptyset$$

Lack of any connection between two COTs in the figure also signifies this fact. Inside each single CoT, Registration, Identification, Authentication and De-registration take place inside respective identity domain (IDP_{SILO_i}) and Authorisation and Service Provisioning take place at any other SP that is inside the same CoT. That is, once the user is authenticated at one identity domain in an IdP, she will be considered authenticated at all IdPs which belong to the same CoT. So, for any $1 \leq i \leq n$ and $idp_i, idp'_i \in COT_i$:

- $IDFN_{idp_i} = IDFN_{idp'_i}$;

- $AUTHN_{idp_i} = AUTHN_{idp'_i}$

Life-cycle of an Identity Federation. Once an entity (IdP or SP) joins in a federation, it will remain there as long as the contract allows. During this time it might be required to update corresponding information of that entity inside the federation. Finally, the entity must be removed from the federation once the contract ends or the entity wants to leave the federation. These tasks are carried out by the respective IT administrators at an IdP or an SP and can be combined to introduce the notion of the Life-cycle for an identity federation. The life-cycle of an identity federation implies the steps that are required to create, revoke, maintain and to utilise a federation. The steps are:

- **Association (Federation)**. The association is the process by which an IdP or SP is added to a new or existing federation.
- **Provisioning**. This is the intermediary step in which the entities (an IdP or SP) utilise the federation to offer services to the users.
- **Maintenance**. This intermediary step allows the administrator to update any information regarding the respective entities.
- **Revocation (Defederation)**. In this step, an entity (an IdP or SP) is removed from the federation.

These four steps essentially define the way a federation can be managed.

Open Unfederated Identity (OUI) Model

Even though the Federated model is one of the most mature models, it has its own drawbacks. The major drawback is the necessity to create federations beforehand and then to maintain it. A federation becomes hard to manage as the number of IdPs and SPs increases. To avoid the complication of creating and maintaining a federation, another model has gained much attention and popularity in recent years with the proliferation of social networks. This model consists of a number of IdPs and SPs, however, they do not need to be a part of a federation to interact with each other. All entities can interact in run-time using the respective identity protocol. Moreover, a user can use any IdP supporting the respective identity protocol to access services from an SP of this model and thus is not restricted to a particular IdP like the Federated Model. That is why I prefer to call it the *Open Unfederated Identity Model*. Many popular social networks such as Facebook, Twitter and Linkedin store a number of different user attributes and act as IdPs using this model to allow users to access services from different online services. OpenID [6] and OAuth [7] are the two most popular identity protocols for this model. Figure 3.8 illustrates the idea of the identity domain in the OUI Model. The absence of the dashed boundary in Figure 3.8 signifies the fact that this model does not have any federation domain.

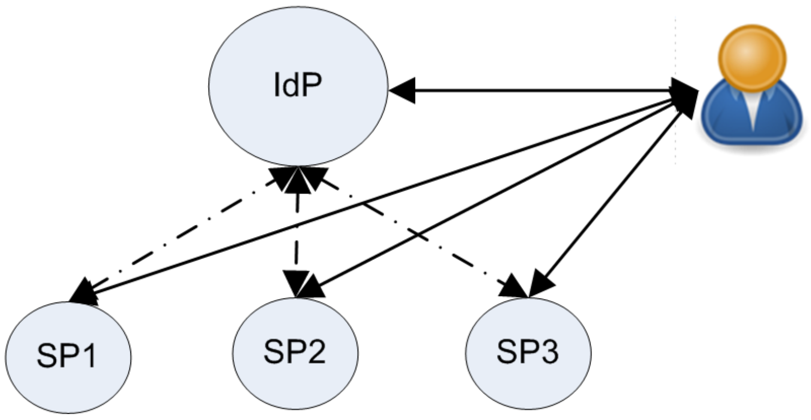


Figure 3.8: Open Unfederated Identity Model.

This model is based on the open trust paradigm in which every entity trusts another entity and thus the issues of trust are not reliant on any factor such as being a part of a CoT. Such characteristics of this model make it easier to deploy and maintain. However, this model may not be suitable for scenarios when a minimum level of trust between different entities is required.

In this model for some $n \in \mathbb{N}$:

- $IDP_{OUI} = \bigcup_{i=1}^{n} idp_i$;
- $SP_{OUI} = \bigcup_{i=1}^{n} \{sharedBy_{OUI}(idp_i)\}$ for all $idp_i \in IDP_{OUI}$;
- $DOMAIN_{OUI} = IDP_{OUI} \cup SP_{OUI}$ for all $1 \leq i \leq n$.

That is, the set of IdPs consists of IdPs supporting the protocol for this model. The set of SPs consists of all the SPs that are related with each single IdP using the $sharedBy$ function. Note that, one single IdP in this model can be used to access services from all SPs of this model or even more than one IdP can be used to access services from a single SP. Also, the identity domain ($DOMAIN_{OUI}$) consists of the set of IdPs (IDP_{OUI}) and the set of SPs (SP_{OUI}) and represents one single identity domain for this model. The representation of this model is illustrated in Figure 3.9. The dashed line in this figure represents that any IdP from the set of IdPs (IDP_{OUI}) can be shared by any service provider from the set of SPs (SP_{OUI}). The domain of this model is the union of the set of IdPs and the set of SPs.
Inside the identity domain, Registration, Identification, Authentication, De-registration take place at one of the IdPs and Authorisation and Service Provisioning take place at one of the SPs. However, once the user is authenticated at one of the IdPs, she will be considered authenticated at all SPs since one IdP can be used to access services from all SPs. This is

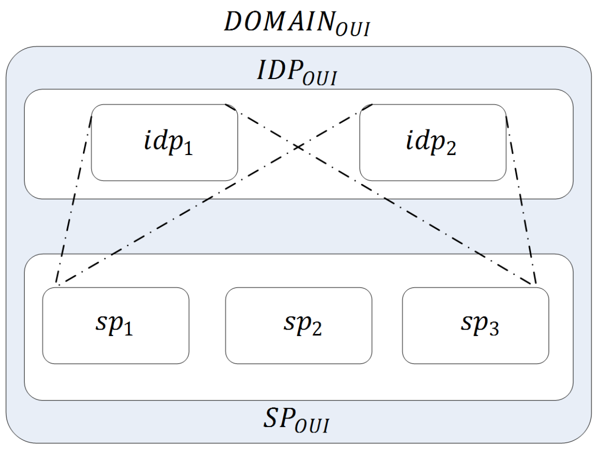


Figure 3.9: Representation of the OUI Model.

made possible by passing any authentication information between an IdP and an SP using an Identity Protocol. So, for any $idp \in IDP_{OUI}$ and $sp \in SP_{OUI}$:

- $IDFN_{idp} = IDFN_{sp}$;
- $AUTHN_{idp} = AUTHN_{sp}$.

This represents the fact that the identification and authentication operations are shared between the IdP and SP within the identity domain.

3.5 Conclusion

In this chapter, a few preliminary concepts on Identity and Identity Management have been provided. Different notions and semantics exist for the same topic related to Identity Management in different places. Therefore, it is required to build a common and consistent vocabulary on related topics of Identity and Identity Management. The concepts provided here will act as the consistent vocabulary and will be used as the point of reference throughout the thesis.

In addition, the developed mathematical framework to formally model the central aspects of IdM have been presented. Also, it has been explored how the framework can be used to model several aspects of an IMS. Being based on formal mathematical foundations, the framework can be used to build a deep understanding of the central issues of IdM. This is the first model of its kind with the potential to be used for rigorous reasoning and formal analysis of IdM. In addition, four application scenarios have been illustrated to show how the model can be applied to analyse a few problems of the current setting of online services and to mathematically model a few popular IdM Models. Among the analysed problems, the focus in this thesis is on two problems: i) the effect of multiple partial identities and ii) the issue of data ownership and controllability and hence in the subsequent chapters, it will be investigated how these two problems can be tackled.

Chapter 4

Comparative Analysis of Identity Management Systems

4.1 Introduction

In this chapter, a comparative analysis of three popular IMSs - SAML-based IMSs, OpenID and OAuth - is presented. Each of these IMSs has been briefly discussed in Chapter 2. The motivation for this comparative analysis is to identify the strength and weakness of each system and lay out the foundation for the subsequent research. With this goal in mind, a taxonomy of requirements for an ideal IMS has been compiled. Then, the taxonomy has been used to compare the chosen systems.

This chapter starts by discussing different requirements from the taxonomy and then provides a comparative analysis of the chosen systems using this taxonomy.

4.2 A Study of Requirements

Early IMSs were developed in isolated projects which had their own sets of requirements to fulfil. These requirements were disparate in nature and lacked consistency. Efforts to converge these requirements into a unified consistent set can be found in the literature [4, 93, 94, 95, 96, 97]. Based on the existing work, especially the requirements given in [97, 93], a list of core requirements has been compiled that any ideal Identity Management System should satisfy. The list of requirements is presented in the form of a taxonomy in Figure 4.1. The list is concise and more focused in the sense that it has excluded some optional requirements (such as Interoperability, Gateways, Law Enforcement/Liability and Affordability) given in [97, 93]. These optional requirements are not regarded as the core set of requirements since they do not influence, in any way, the performance and the security of

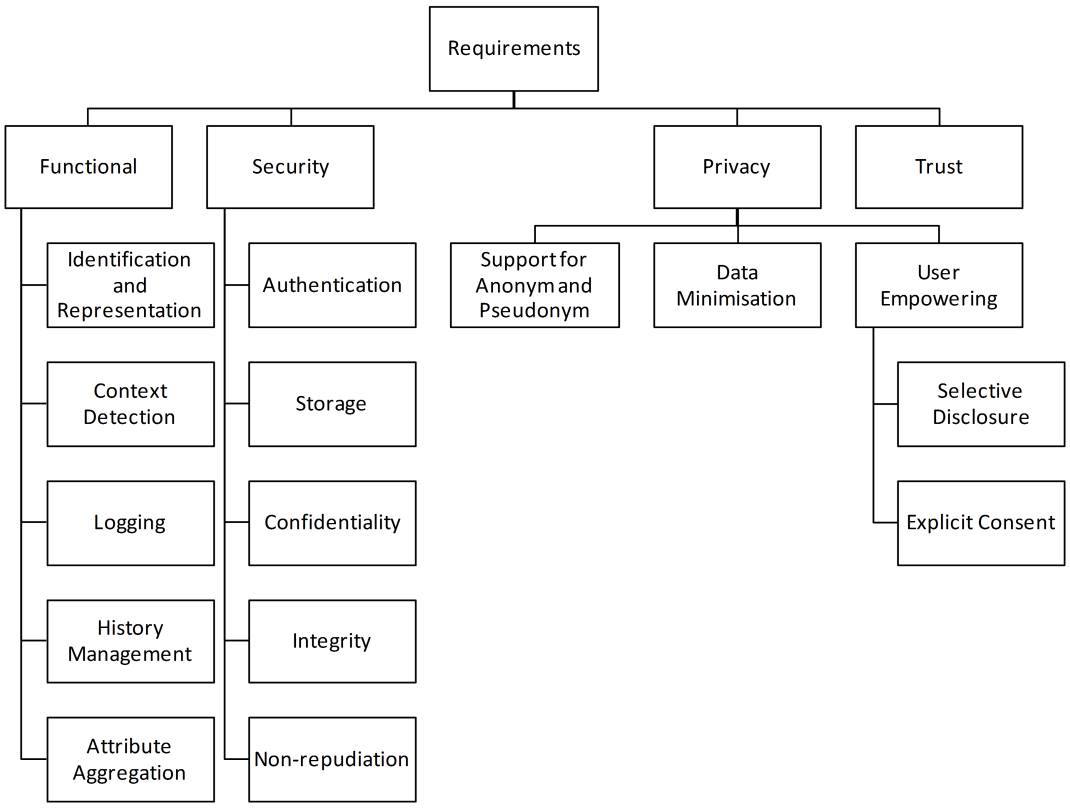


Figure 4.1: Taxonomy of core requirements for Identity Management Systems.

an IMS. Moreover, a few additional requirements (Storage, Context Detection and Attribute Aggregation) have been added which have not been considered in any previous work. Such requirements can play a crucial role for introducing innovative use-cases with the emergence of social networks and the popularities of mobile devices. Next, each of these requirements is explained.

4.2.1 Functional Requirements (FR)

The core set of requirements for a functional Identity Management System are as follows.

Identification and Representation (I-R). The main functionality of an IMS is to identify its users correctly, and hence, represents the core requirement of an IMS. As mentioned earlier, a user is identified with an identifier in a specific system. Different Identity Management Systems use different types of identifiers and represent them in different ways.

Context Detection (CD). In the literature of Context-aware Services, the term *Context* has different meanings. According to [98], contexts are described as locations, identities of nearby people, objects and changes to those objects. The importance of detecting contexts during the service provisioning phase in an IMS was initially ignored. This can result in the release of unnecessary yet sensitive attributes to different parties. To be effective, secure and privacy-friendly, the IMS should have the ability to detect the context and then inform the user about the context so that the user can choose the appropriate partial identity. This is still an on going research area.

Logging (LOG). Each transaction (the action of releasing attributes to an SP) using an IMS should be logged. The log then can be used to facilitate the History Management feature (see below).

History Management (HM). History Management should be an important part of any modern IMS. A good history management facility should display all the logged transactions in a user-friendly way. This will allow users to check their data trail, i.e. what attributes have been released to what entity, when they have been released, if there is any privacy policy attached to the released data, etc. In addition, users should be able to delete any existing data trail. Such detailed history management facility will allow users to build trust in an IMS, help them to fine-tune their privacy preferences and to make an informed choice for any future correspondence.

Attribute Aggregation (AA). An Attribute Aggregation mechanism would allow the user to aggregate attributes from multiple providers and pass them to an SP in a single service session. With the proliferation of online IdPs, it is not realistic to assume that all attributes of a single user will be stored in a single IdP. Essentially, the majority of users store a lot of different attributes across multiple IdPs. Allowing users to aggregate their attributes in a single session will enable novel service access scenarios. Therefore, it is timely to integrate this feature in an IMS.

4.2.2 Security Requirements (SR)

An IMS can be used for accessing different services ranging from participating in social network activities, blogging, emailing, accessing Government services, online banking and e-commerce activities. There are different levels of security for each of these services yet each service requires a minimum security guarantee to ensure that only an authenticated and authorised user can access the requested service securely. Any lapse of security could not only allow intruders to capture sensitive personal information, but also impersonate a valid user and vandalise her reputation with illicit activities. Unlike the real world, it is difficult for the impersonated person to prove her innocence once an attacker impersonates her. Therefore, a minimum level of security should be guaranteed in every Identity Management System to ensure that user attributes are stored in an IdP securely and are released to service providers in a secure manner. The following requirements are necessary and sufficient to ensure a minimum level of security for an IMS.

Authentication (AUTH). A core security requirement for an IMS is to authenticate a user in a secure manner. The identification process described previously identifies a user with the identifier and the authentication process verifies the association between the user and the supplied identifier. As mentioned earlier, the verification process depends on a credential. There are different levels of assurance or grades for satisfying the authenticity which generally depends on the combination of the identifier and credential. For simple web services, a simple identifier and an easy to use credential such as a user-id and password could be enough. For more secured services such as financial or Government services, biometrics, OTP (One Time Password), hardware tokens, etc. are more appropriate. Moreover, the authentication process must be carried out securely to ensure that a malicious user cannot misrepresent herself by authenticating as another user.

Storage (STO). Each attribute of a user must be stored securely in an IdP. This will ensure that user attributes will not be readily exposed to an attacker even if the IdP where the attributes are stored is compromised. With recent attacks on some major global companies such as Sony [99], Linkedin [100] and Apple [101] where user data were stolen from their databases, securely storing attributes is even more crucial. This requirement can be achieved by encrypting user attributes before storing them and safeguarding the respective database with different database security mechanisms.

Confidentiality (CON). Confidentiality is to ensure that the transmitted data between two parties is not disclosed to any unauthorised entity. It guards against any eavesdrop attack by which an attacker can retrieve data in transmission. Traditionally, confidentiality is satisfied using cryptographic mechanisms. An IMS has to make sure that it uses latest cryptographic mechanisms to ensure confidentiality.

Integrity (INT). Integrity is to ensure that the transmitted data is not altered during transmission. It can also be satisfied using cryptographic mechanisms.

Non-repudiation (NR). Non-repudiation ensures that a user, once committed for a transaction, cannot deny her commitment in the transaction. This is very important in e-commerce scenarios where monetary transactions are involved. Once again, cryptographic mechanisms can be used to ensure non-repudiation.

4.2.3 Privacy Requirements (PR)

The privacy of a user and the user's identity is a very important issue. Each single piece of datum, no matter how negligible it seems, needs to be privacy-protected since such data could be used to build a profile of the user without her knowledge. Privacy Enhancing Technologies (PETs, in short) are the mechanisms by which privacy can be guaranteed. Quoting the definition of PET from [102]: "A coherent system of ICT measures that protects privacy by eliminating or reducing personal data or by preventing unnecessary and/or undesired processing of personal data; all without losing the functionality of the data system". Integrating an IMS with PET should protect the privacy of the user. The requirements discussed below can be used to ensure the privacy of a user in the IdM setting.

Support for Anonym and Pseudonym (SAP) A Privacy-aware IdM System should have strong support for Anonymity (denoted as *ANON* in the following section) and should be used as the base rule for all underlying communication. Likewise, it should support the usage of Pseudonym (denoted as *PSD* in the following table) to ensure that users are unlinkable at different SPs as desired. Privacy protection techniques using various cryptographic methods can be used to satisfy this requirement.

Data Minimisation (DM). The data minimisation requirement will ensure that only the required data is transferred, stored and processed at an SP. This can improve the privacy of a user as it guards against the release of unnecessary yet sensitive personal data to any party which ultimately reduces the risk of privacy breach.

User Empowering (UE). All the above mentioned privacy requirements will be in vain if users are not in control of their data and have no idea what data is released to what entity. To enable such control, the user must be empowered. This can be achieved by satisfying two requirements below that will allow them to disclose their data selectively and to provide explicit consent before any data is being released to an SP.

- **Selective Disclosure (SD)**. This requirement will allow a user to select specific attributes before they are released to an SP. By this way, the user can choose specific attributes for a specific SP.
- **Explicit Consent (EC)**. This requirement will enable a user to provide explicit consent before any data is released to an SP.

4.2.4 Trust Requirements (TR)

Based on the chosen systems, two different types of trust can be identified. One type, denoted as *Federated Trust*, represents the trust requirements in the setting of the FIM Model. The other trust, denoted as *Open Trust*, represents the trust requirements in the setting of the Open Unfederated Identity Model. The *Open Trust* paradigm does not impose any specific trust requirement as all entities in this paradigm trust each other. However, the *Federated Trust* imposes several complex trust requirements. Such trust issues in the setting of FIM will be further explored in later chapters.

4.3 Comparative Analysis

Now, a comparative analysis of the three most popular Identity Management Systems - SAML-based IMSs, OpenID and OAuth - is presented. Identifying what requirements are satisfied by which systems and presenting them in a concise manner can be challenging since the requirements satisfied by different IMSs are written in their respective specifications and scattered among several documents. This section aims to aid in this regard by presenting the finding in a concise way so that any reader can instantly deduce which requirements are fulfilled by which systems. This will also help to identify the strength and weakness of each system as well as to pinpoint any gaps in them.

Analysing existing Identity Management Systems according to a certain set of criteria is not novel and two examples that this analysis has followed are [93, 94]. A few Identity Management Systems, such as Microsoft .NET Passport, Liberty Alliance Architecture and Novell DigitalMe, and applications, such as Mozilla 1.4 Navigator, Microsoft Outlook Express 6 SP1 and CookieCooker, have been analysed against a set of requirements in [93]. However, this work can be considered out-dated in the sense that none of those systems is still functional. A more recent attempt with the same objective can be found in [94] in which four Identity Management Systems (Liberty Alliance Architecture, Shibboleth, PRIME Architecture and Microsoft CardSpace) have been analysed against a set of requirements. However, all systems except Shibboleth are either obsolete (Liberty Alliance and the CardSpace) or have very limited acceptance (PRIME Architecture).

This analysis is similar to that of [94], however, it is more concise (327 pages in [93] and 76 pages in [94]), concentrates on two current systems (OpenID and OAuth) which have never been analysed in this manner. Also, the analysis is more focused since it concentrates only on the core set of requirements of an ideal IMS. Moreover, the findings in [94] were published in a descriptive way and readers would need to read through different sections to identify the missing features. The findings have been presented in tabular formats which is more illustrative and will allow any reader to instantly identify which requirements are met by which systems.

To compare the chosen systems against the set of requirements, it was needed to understand their architectures and to familiarise with their protocols. Then, it has been checked if each requirement is met by a system. For this, their protocol descriptions, specification documents and corresponding wiki-pages have been consulted and sometimes development forums have been visited. The findings of the analysis are presented in Table 4.1 and Table 4.2. The tick (✓) mark, sometimes accompanied with an explanation in brackets (), has been used to indicate that an IMS satisfies a respective requirement and the character '✗' has been used to indicate that the system does not satisfy the respective requirement.

Table 4.1: Functional and Security Requirements.

	Functional					Security				
	I-R	CD	LOG	HM	AA	AUTH	STO	CON	INT	NR
SAML	✓ (IdP Specific)	✓	✗	✗	✗	✓	✓	✓	✓	✓
OpenID	✓ (URI/XRI)	✓	✗	✗	✗	✓	✓	✓	✓	✓
OAuth	✓ (IdP Specific)	✓	✗	✗	✗	✓	✓	✓	✓	✓

Table 4.2: Privacy and Trust Requirements.

	Privacy					TRUST
	SAP		DM	UE		
	ANON	PSD		SD	EC	
SAML	✗	✓	✓	IdP Specific	IdP Specific	Federated
OpenID	✗	✗	✓	Provider Specific	Provider Specific	Open
OAuth	✓	✗	✗	✗	Provider Specific	Open

From Table 4.1, how the identifier is represented depends on the specific IdP for SAML and OAuth. The most general example of the identifier is the username or email address. However, there are other types of identifiers (e.g. URI/XRI for OpenID) as well. None of the systems has any mechanism to detect the context of the user apart from the application domain of the SP from where the user tries to access the service. There are other forms of contexts which will be discussed in later Chapter 7. None of the systems has any specific requirement in their specifications for transaction logging and history management. However, implemented systems based on a specific IMS may include these features, and hence their availability to the user is fully dependent on the specific implementation. Also, no system has any facility for aggregating attributes from multiple IdPs. As for the security requirements all IMSs have strong support to ensure that security requirements are properly satisfied.

OpenID does not have any support for anonymous or pseudonymous service accesses, SAML only supports pseudonymous service accesses and OAuth only supports anonymous accesses. Data minimisation in SAML is achieved via an SP requesting specific attributes from an IdP. Even though OpenID and OAuth also allow an SP to request specific attributes, how it is achieved depends on a specific implementation. OAuth does not have any support for selective disclosure of attributes. All other UE requirements are dependant on the specific provider. These issues will be analysed in greater detail in later chapters. As mentioned earlier, SAML is based on the federated trust and the other two systems are based on the open trust paradigm.

4.4 Conclusion

In this chapter, three popular Identity Management Systems have been compared using a taxonomy of requirements. The comparative analysis of the three popular IMSs highlight some major problems. In particular, no system has any facility for detecting the context apart from the application domain. Moreover, what defines a context with respect to the identity management is not properly clarified in the existing systems. None of the chosen systems has any mechanism for attribute aggregation from multiple IdPs. How data minimisation, selective disclosure and explicit consent requirements are met in these systems depends on a specific implementation and deserves further attention as it can drastically affect the privacy of a user. These issues will be further investigated in later chapters.

Among the analysed systems SAML is one of the most matured and widely used technologies. It already satisfies a good number of requirements as illustrated above. There is already one implementation of SAML (SimpleSAMLphp [11]) that has limited support for selective disclosure and explicit consent using its *Consent* module. The existing research on attribute aggregation has been based on SAML. For these reasons that SAML is a good candidate upon which this research can be built on. That is why SAML has been chosen as the basis for the proposed IMS in following chapters.

Chapter 5

Dynamic Identity Federation

5.1 Introduction

In the previous chapter, a comparative analysis of three IMSs has been presented using a taxonomy of requirements. Based on the analysis, SAML has been chosen for the subsequent research. It has been mentioned earlier that SAML is one of the most widely used technologies for creating federations. Despite its several advantages, there is one major limitation of SAML, the method by which trust is established and maintained, as mentioned in Chapter 2. To enable federations using SAML, trust among participating organisations has to be pre-configured by system administrators. Pre-configuring trust before any interaction hinders the establishment of a federation between two prior unknown parties in a dynamic fashion. This makes it hard to scale and maintain as the number of parties starts to increase in the federation. As discussed in Chapter 2, the existing works allow a federation to be created dynamically in a semi-automatic fashion and requires a major change in SAML. In addition, the focus of these works are on how an entity can join a federation and discussions on other aspects (maintenance, provisioning and revocation) are rare. It is essential to address all aspects covering the full life-cycle of a dynamic identity federation. In this chapter, the proposal and implementation for managing federations in a dynamic fashion are presented.

5.2 Dynamic Federation

Creating a federation in a dynamic fashion is an ongoing and active research topic and there are several works to address the issue as discussed in Chapter 2. However, all previously mentioned works have used the term *Dynamic Federation* literally without defining it formally. Without a formal definition, the term may be interpreted in multiple ways which can cause confusion. This is why the term *Dynamic Federation* is defined at first.

Definition 20 *A Dynamic Federation with respect to Identity Management is a business model which, in addition to entities of a traditional federation, consists of a group of two or more previously unknown entities federated together dynamically without any prior business and technical contract with the aim of allowing users to access services under certain conditions.*

This definition is a stark contrast with the traditional concept of the identity federation based on SAML in which there must be a legally binding technical and business contract between participating organisations before they can join any federation. The primary advantage here is the ability to join a federation in real time. However, the lack of any legally binding contract means that organisations must consider the possibility of negative consequences since no party is contractually bound to behave as it should and therefore take proper precautions. This leads to the topic of trust which will be explored below.

The life-cycle of a dynamic federation closely follows the steps in the life-cycle of a traditional identity federation: association, provisioning, maintenance and revocation. However, now each steps is carried out in a dynamic fashion. Providing the capability to create federations dynamically in real time will enable users to carry out some of these steps. This contrasts with traditional federations where all steps are carried out by the respective administrator. However, the task of *Provisioning and Maintenance* must be carried out by the respective administrator to ensure that a malicious user cannot update the metadata of an entity in the federation. These issues have not been considered in previous works.

5.2.1 Trust Issues

According to the proposed definition of a dynamic federation (see Definition 20), participating organisations may not trust each other entirely since they are previously unknown and there is no contract to make them accountable in case one acts maliciously. An IdP may not want to release sensitive attributes to an SP that has been added dynamically and an SP may not trust all attributes released by an IdP that has been added dynamically. This leads back to the open trust paradigm of OpenID where every organisation (IdP/SP) is assumed to be trusted even though it might behave maliciously.

Such trust issues have not been considered while drafting the Dynamic SAML [20] (see Chapter 2). It is believed that this is a serious flaw and trust should be reinstated in dynamic federations. Technically, joining a federation will require the parties to exchange and store each other's metadata. The SimpleSAMLphp implementation based on Dynamic SAML only allows one to create a *semi-automatic* federation (as discussed in Chapter 2) where an SP, to join with an IdP, requires that the IdP is pre-configured in the TAL (Trust Anchor List, see Chapter 2) of the SP to allow any user to choose that IdP. However, to harness the true potential of dynamic federations, it must be ensured that both parties can be added dynamically. Moreover, an IdP in SimpleSAMLphp does not distinguish between statically and dynamically added SPs. This allows the IdP to release the same level of (sensitive) attributes to both types of SPs. In summary, there are two requirements to fulfil:

1. fully automate the life-cycle in a federation for both parties, and
2. ensure that trust is established in a dynamic federation.

With these two goals in mind and based on how different entities are federated with each other, the notion of fully trusted, semi-trusted and untrusted entities are introduced.

Definition 21 *Fully trusted entities are IdPs and SPs in a traditional SAML federation which have legal contracts between them.*

They are so called since each IdP (or SP) inside a federation trusts any SP (or IdP) in the same federation to behave as intended and can be made accountable in case the other party behaves maliciously.

Definition 22 *Semi-trusted entities are SPs in a dynamic federation that have been added dynamically to an IdP inside the federation under* ***some conditions*** *without the presence of any contract between them and to whom at least a user of the IdP has released a subset of her attributes.*

They are so called since the user wants to release a subset of their attributes to these SPs inside the dynamic federation even though the IdP in the same federation may not fully trust such an SP to behave as intended. Therefore, such an SP might not be made accountable by an IdP in cases they behave maliciously with the absence of any contract between them.

Definition 23 *Untrusted entities are IdPs and SPs in a dynamic federation which have been added dynamically under* ***some conditions*** *without the presence of any contract.*

They are so called since any IdP (or SP) inside a dynamic federation may not trust at all any other dynamically added SPs (or IdPs) in the same federation to behave as intended.

It is important to understand that a dynamic federation may accommodate as many fully trusted entities as possible. As such, a dynamic federation is an extension of the traditional federation. Even though, different entities have been categorised based only on how they federate, these three types of entities have their own effect on other parts of the life-cycle of the federation which will be explored below.

The term “some conditions” in the definition of the semi-trusted and untrusted entities require further explanation. It can be the combination of several different conditions by which a federation between two entities can be created, updated or removed dynamically, the conditions for establishing and removing individual trust with each other in such a federation, the conditions by which attributes are released to a semi-trusted SP and the conditions by which an SP treats attributes of a user from an untrusted IdP. Semi-trusted and untrusted entities of different dynamic federations should have different sets of conditions suitable for their business models and service provisioning scenarios. Here, a set of conditions is presented that have been assumed for developing a proof of concept of managing a dynamic federation using SAML.

- Only a valid user of an IdP is allowed to add an SP to that IdP dynamically. This is to ensure that only those SPs that the users want to access for service provisioning are added in a dynamic federation. This is missing in the current implementation of SimpleSAMLphp.
- Once an SP is added to an IdP, the SP must add the IdP to its TAL to ensure that the user can select the IdP from its WAYF service. This nullifies the need to pre-configure the IdP in the SP.
- Only a valid user of an IdP who has added an SP to that IdP dynamically is allowed to initiate the procedure to revoke the SP from the federation with that IdP. In this case, the IdP will be removed from the TAL of the SP and the SP will be removed from the TAL of the IdP.
- Only the administrator of each entity (an IdP or SP) is allowed to dynamically update information in the metadata stored in the TAL of other entities. This is to ensure that users with malicious intent cannot update or corrupt such information of the metadata.
- A dynamically added SP must be tagged as untrusted in an IdP at the initial stage. Only after a user, after being authenticated at the IdP, has agreed to release a subset of her attributes to the SP, the SP should be re-tagged as semi-trusted.
- A dynamically added IdP should always be tagged as an untrusted entity for every SP.
- An IdP should ensure that it does not release any attributes to any untrusted entity.
- An IdP should ensure that crucial and sensitive attributes are not released to any semi-trusted entity since there is no guarantee that they will be handled as intended. Therefore, it should allow their administrators to configure what attributes should be released to a semi-trusted entity.
- It is up to the discretion of each SP as how they want to treat released attributes from an untrusted IdP. They could use the NIST (National Institute of Standards and Technology) LoA (Level of Assurance or Level of Authentication) guidance of 1 to 4 where

Level 1 conforms to the lowest assurance and 4 conforms to the highest assurance [104]. Usually, the LoA level comes from an IdP and is embedded inside a SAML assertion to provide the level of assurance for a certain authentication mechanism at the IdP. However, an SP should consider implicitly that LoA 1 is the maximum that can ever accompany a SAML assertion from any untrusted IdP. Since, how the released attributes will be handled depends on the individual SP, it can vary from one SP to another even inside the same federation.

To summarise, it is proposed that entities have to be federated/defederated in a fully automatic fashion without any administrative intervention in order to harness the full potential of dynamic federations and while doing so all entities must consider trust issues involved. Moreover, the proposal also outlines the conditions for managing such federations in a fully automated manner. It should be noted that the conditions above are some of the many ways to fulfil the proposal; other implementations may require different sets of conditions suitable for their use-case scenarios.

5.3 Proof of Concept: Dynamic Management

To illustrate the applicability of the proposal for managing a dynamic identity federation, a the proof of concept has been implemented. For this, SimpleSAMLphp has been used and modified to meet different requirements. In this section, the implementation is presented using three different scenarios: i) dynamic federations, ii) dynamic defederations and iii) dynamic update of federations.

5.3.1 Use-case: Dynamic Federation

The architectural setup for this scenario is that there is one IdP and one SP deployed with the modified version of SimpleSAMLphp. At the beginning, the IdP and SP are not part of a common federation (they individually may be part of separate federations) and they have no prior knowledge of the other party. The IdP is configured to use a MySQL database to store user attributes including identifiers (usernames) and credentials (passwords) in a table called *users*. In addition, the IdP uses two tables called *semitrusted* and *untrusted* to store the entity ID of the semi-trusted and untrusted SPs. Moreover, the IdP uses two additional tables called *idpadmin* and *spadmin*.

- The *idpadmin* stores the entity ID of SPs and the IdP-generated admin codes that can be used to update the metadata of the IdP stored at the TAL of the SP.

- On the other hand, the *spadmin* stores the entity ID of SPs with the SP-generated admin codes to update the metadata of the SP stored at the TAL of the IdP.

Similarly, the SP is also configured to use a MySQL database including a table called *untrusted* to store the entity ID of an IdP that has been federated dynamically. Likewise, the SP uses two additional tables called *idpadmin* and *spadmin*.

- The *idpadmin* table at the SP stores the entity ID of an IdP along with an IdP-generated admin code to update the metadata of the IdP stored at the TAL of the SP.
- The *spadmin* table at the SP stores the entity ID of an IdP along with an SP-generated admin code to update the metadata of the SP stored at the TAL of the IdP.

Note that only administrators of the IdP or the SP have access to the *spadmin* and *idpadmin* tables since general users are not allowed to make any changes to the metadata.

With this setup, the protocol flow, illustrated in Figure 5.1, for federating the IdP and the SP dynamically while trying to access a service from the SP is given below.

1. A user visits the SP for the first time to access one of its services. Assuming that the user is not authenticated, the user is redirected to the WAYF service to discover her IdP and a list of federated IdPs with the SP is shown.
2. Since the IdP and SP are not part of a common federation, the IdP list at the SP does not contain the IdP. However, since the SP supports (more precisely the SimpleSAMLphp that has been used to deploy the SP supports) the proposal of dynamic federations, it contains two additional text fields (Figure 5.2) which allow the user to enter the entity ID of her IdP and a code to ensure that only the valid users of the IdP have the ability of federating an SP with an IdP.
3. Since the user does not have the code, she logs in to her IdP and generates a code using the *Generate* button at the Generate IdP Code page (Figure 5.3). This page also checks the *semitrusted* table to see if there are any dynamically added SPs. Since there are none, it says so (Figure 5.3). Once the *Generate* button is clicked, a 4 digit random number is generated and displayed (Figure 5.4). This random number is also stored temporarily in a database table called *code* along with the user identifier. The code is used during metadata exchange for verification (see below). Here, a 4 digit code has been opted, other implementations may opt for other type of codes according to their own requirements.
4. After generating the code, the user inserts the entity ID of the IdP and the recently generated code to the WAYF page of the SP (Figure 5.5) and clicks the *Add* button.

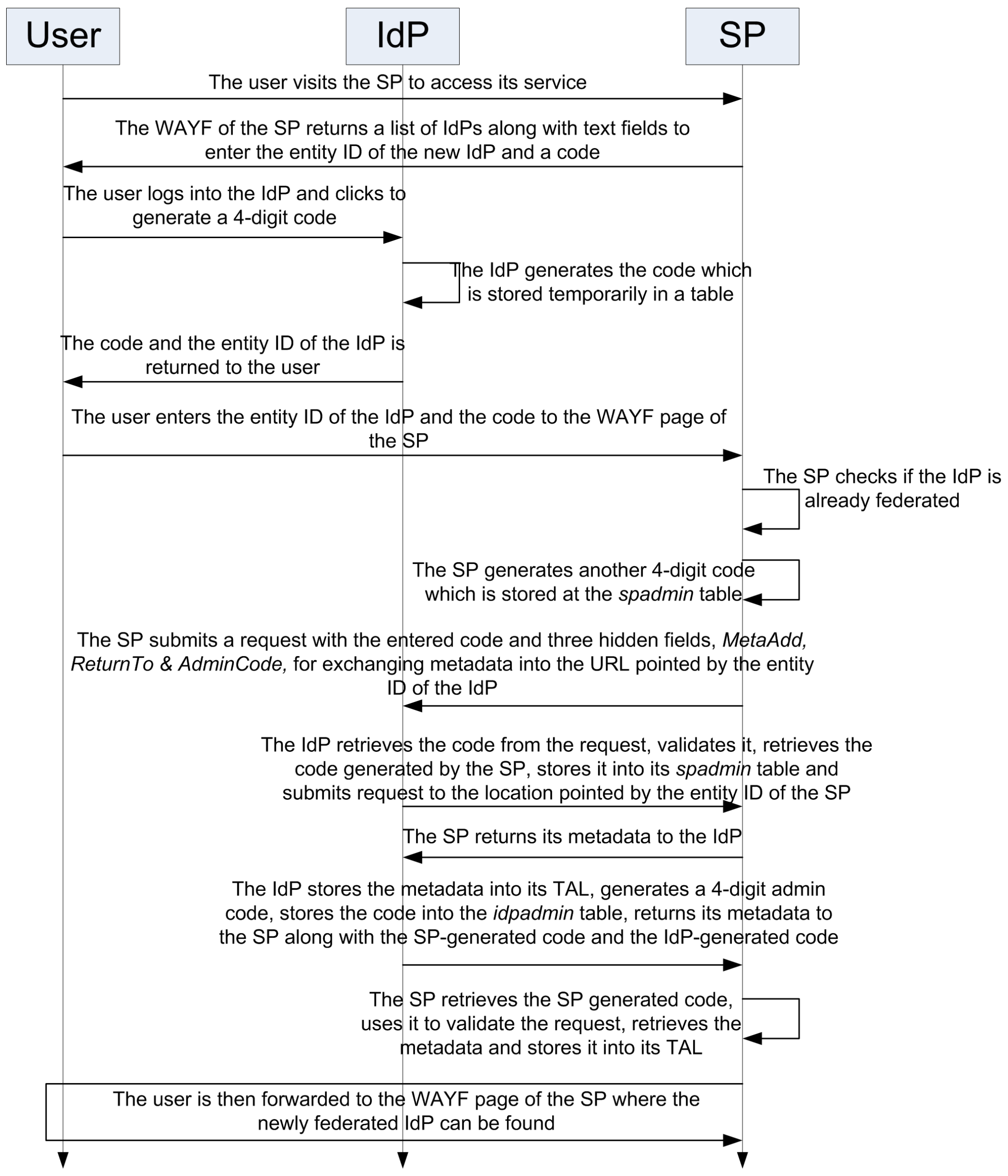
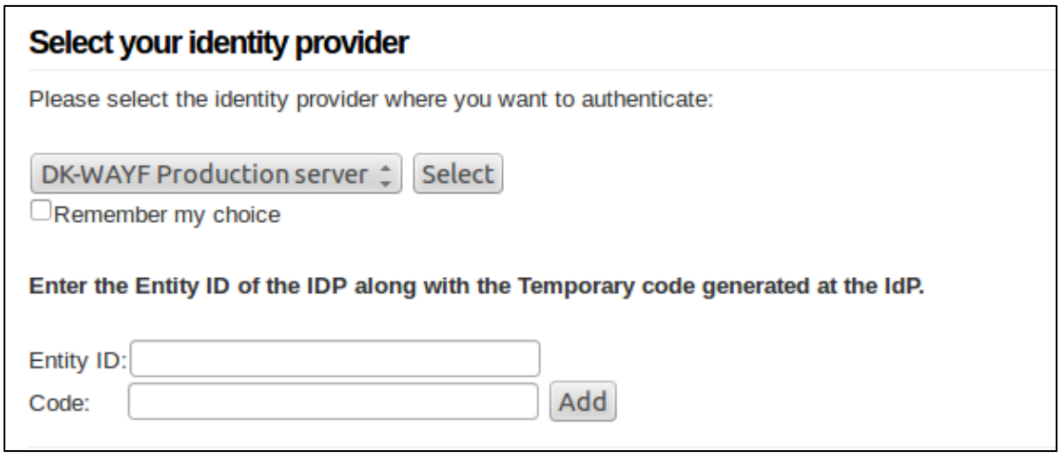

User
IdP
SP
The user visits the SP to access its service
The WAYF of the SP returns a list of IdPs along with text fields to enter the entity ID of the new IdP and a code
The user logs into the IdP and clicks to generate a 4-digit code
The IdP generates the code which is stored temporarily in a table
The code and the entity ID of the IdP is returned to the user
The user enters the entity ID of the IdP and the code to the WAYF page of the SP
The SP checks if the IdP is already federated
The SP generates another 4-digit code which is stored at the *spadmin* table
The SP submits a request with the entered code and three hidden fields, *MetaAdd, ReturnTo* & *AdminCode,* for exchanging metadata into the URL pointed by the entity ID of the IdP
The IdP retrieves the code from the request, validates it, retrieves the code generated by the SP, stores it into its *spadmin* table and submits request to the location pointed by the entity ID of the SP
The SP returns its metadata to the IdP
The IdP stores the metadata into its TAL, generates a 4-digit admin code, stores the code into the *idpadmin* table, returns its metadata to the SP along with the SP-generated code and the IdP-generated code
The SP retrieves the SP generated code, uses it to validate the request, retrieves the metadata and stores it into its TAL
The user is then forwarded to the WAYF page of the SP where the newly federated IdP can be found

Figure 5.1: Protocol flow for a dynamic federation.

5. Once the Add button is clicked, it is verified that the entity ID field or code is not null and that the inserted entity ID is not already part of the federation (dynamically or statically). If any part of the verification process fails, an appropriate error message is displayed and the user is redirected to the WAYF page where she can start again.

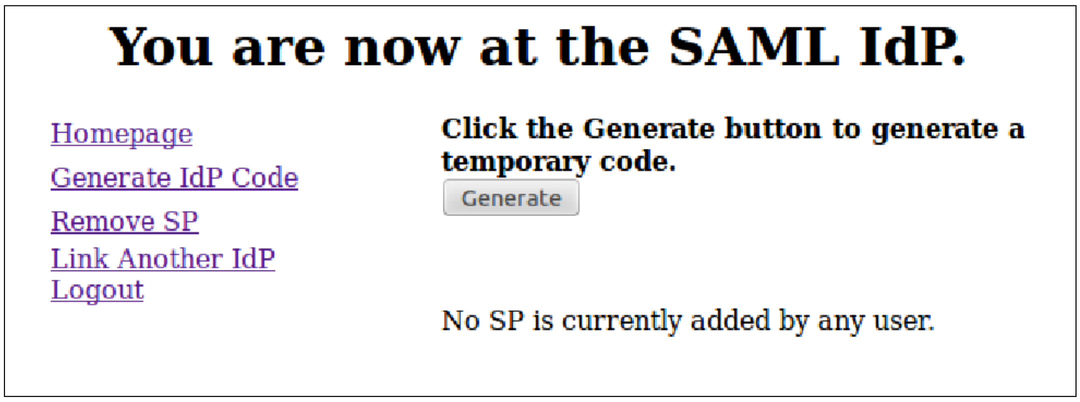
6. Assuming there is no error during verification, the SP generates a 4 digit random code which is then stored at the *spadmin* table along with the entity ID of the IdP. Then, a request to retrieve the IdP metadata from that entity ID with some specific values is posted. The request contains the entity ID and the code that the user has entered and three hidden fields called *MetaAdd, ReturnTo* and *AdminCode*. The values of the *MetaAdd* and *ReturnTo* fields contain the entity ID of the SP and the URL of the ser-


Select your identity provider

Please select the identity provider where you want to authenticate:

Enter the Entity ID of the IDP along with the Temporary code generated at the IdP.

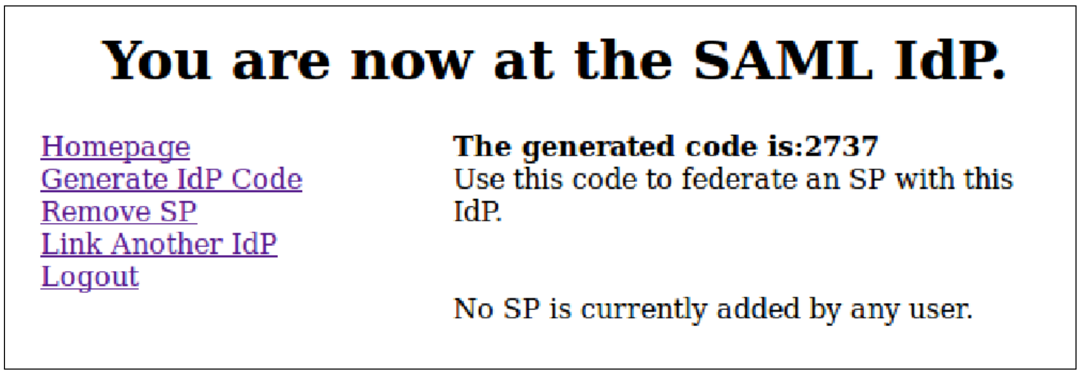
Figure 5.2: Additional text fields at the WAYF.


You are now at the SAML IdP.

Click the Generate button to generate a temporary code.

No SP is currently added by any user.

Figure 5.3: Code generate page at the IdP.


You are now at the SAML IdP.

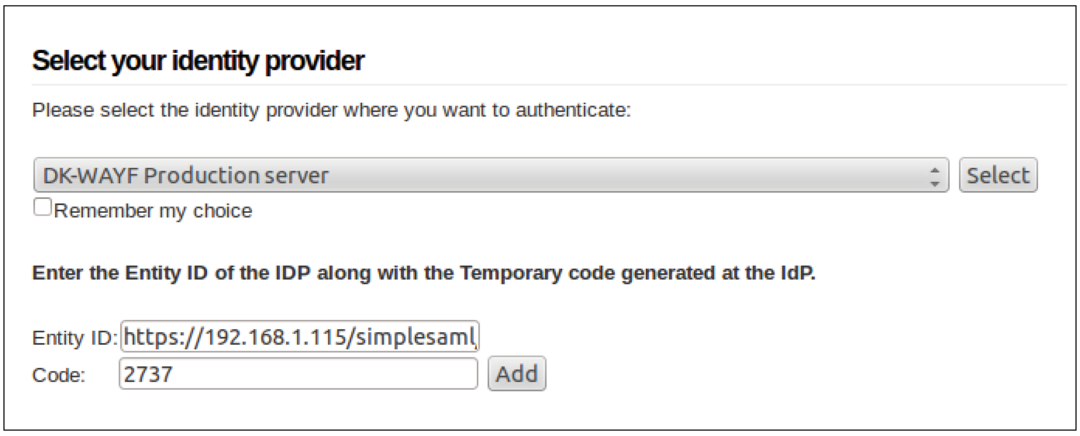
The generated code is:2737
Use this code to federate an SP with this IdP.

No SP is currently added by any user.

Figure 5.4: Generated code at the IdP.

vices that the user requested in the first place which initiated the SAML flow, whereas the *AdminCode* field contains the SP generated random admin code. Remember that SimpleSAMLphp implementation of dynamic SAML requires that entity ID should be an endpoint from where the metadata can be fetched.

7. Once the Appropriate end point of the IdP receives this request, it checks if there is a field called *MetaAdd*. If found, it knows that this is a special request for exchanging metadata with the requested SP. If not found, it assumes that it is a normal metadata fetch request and returns its metadata. Since the request contains a *MetaAdd* field, it


Select your identity provider

Please select the identity provider where you want to authenticate:

DK-WAYF Production server

Remember my choice

Enter the Entity ID of the IDP along with the Temporary code generated at the IdP.

Entity ID: https://192.168.1.115/simplesaml

Code: 2737

Figure 5.5: Entering values at the WAYF.

checks for a code field and retrieves its value and verifies if the same value can be found in the *code* table of its MySQL database. If found, it indicates that this request for exchanging metadata is valid. If not found, an error message is returned to the SP.

8. Assuming that the code field contains a valid code, the IdP retrieves the value of the *MetaAdd* and *AdminCode* fields which contain the entity ID of the SP and the 4 digit SP-generated admin code. The IdP stores this admin code in the *spadmin* table at the IdP along with the entity ID of the SP and sends an HTTP GET request to that location. GET has been used since there are no other parameters to pass during the metadata fetch process from the SP.

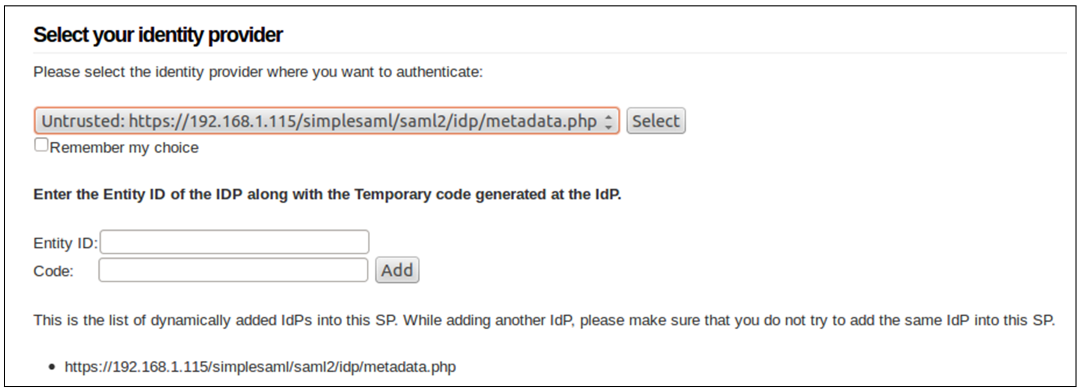
9. When the SP receives this request, it returns its metadata.

10. Once the metadata is retrieved, the IdP goes through the specified verification process for a dynamic SAML (verifying the embedded certificate and verifying the signature on the metadata using that certificate). If the verification is correct, the metadata is stored in its repository and the SP is added to its TAL. In addition, the user identifier for the code is retrieved from the *code* table and the entity ID of the SP is added into the *untrusted* table of the database along with the code and the user identifier. Next, the used code is removed from the *code* table to ensure that it cannot be used again. If the metadata verification is not correct, then the respective entry from the *spadmin* is removed and an error message is returned to the SP.

11. If everything goes accordingly at the IdP, it generates a 4 digit random admin code which is then stored in its *idpadmin* table along with the entity ID of the SP. Then, the metadata of the IdP along with the *ReturnTo* field and its value, a *code* field containing the previously generated value and the *AdminCode* containing the recently IdP-generated 4 digit admin code are returned to the requesting endpoint of the SP, where the metadata, *ReturnTo*, *code* and *AdminCode* values are separated. As before, the SP goes through the specified verification process for a dynamic SAML. If verification

is correct, the metadata is stored in its repository and the IdP is added to its TAL. In addition, the entity ID of the IdP along with the code is added into the *untrusted* table of its database making the IdP to be tagged as untrusted. Moreover, the entity ID of the IdP along with the value of the *AdminCode* parameter are stored in the *idpadmin* table.

12. Finally, the user is redirected to the URL retrieved from the *ReturnTo* field (the URL of the service requested initially) which in turn takes the user back to the IdP selection page of the WAYF. However, as the IdP has already been added, the list contains the list of the IdP and it is tagged as an untrusted IdP (Figure 5.6). Moreover, it is shown to the user that the IdP has already been added as an untrusted IdP so that no other users try to add it once again which will result in an error.


Select your identity provider

Please select the identity provider where you want to authenticate:

Untrusted: https://192.168.1.115/simplesaml/saml2/idp/metadata.php [Select]
☐ Remember my choice

Enter the Entity ID of the IDP along with the Temporary code generated at the IdP.

Entity ID:
Code: [Add]

This is the list of dynamically added IdPs into this SP. While adding another IdP, please make sure that you do not try to add the same IdP into this SP.

- https://192.168.1.115/simplesaml/saml2/idp/metadata.php

Figure 5.6: Added IdP at the WAYF.

In the traditional SAML scenario, it is assumed that both the IdP and SP are visible to each other as they are expected to be online. This assumption does not hold for all SAML IdPs which might be installed locally inside the user's computer from where she is trying to access a service. To accommodate this use-case, the implementation of the SP at the SimpleSAMLphp has been modified so that the SP can communicate with the local IdP. This has been done by embedding a JavaScript into an XHTML (Extensible Hypertext Markup Language) form which is submitted automatically at the entity ID of the IdP. Since the JavaScript runs inside the browser and the local IdP is visible to the browser, it can properly submit these requests. The SP, on the other hand, is assumed to be online and is visible to the IdP. Hence, no special treatment is required.

5.3.2 Use-case: Dynamic Defederation

The setup for this scenario is that an IdP and an SP have been federated dynamically using the steps described above and a user wants to defederate them dynamically. A user is only allowed to defederate those entities that she had federated previously.

With this setup, the protocol flow, illustrated in Figure 5.7, for defederating the IdP and the SP dynamically is given below.

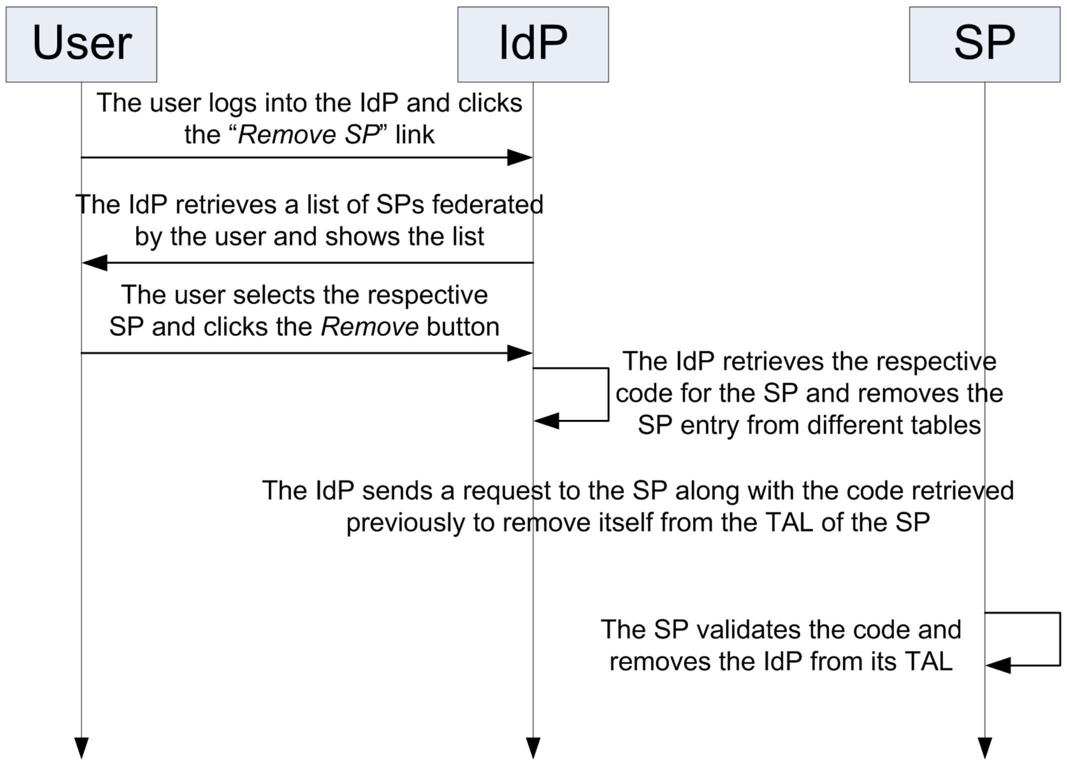
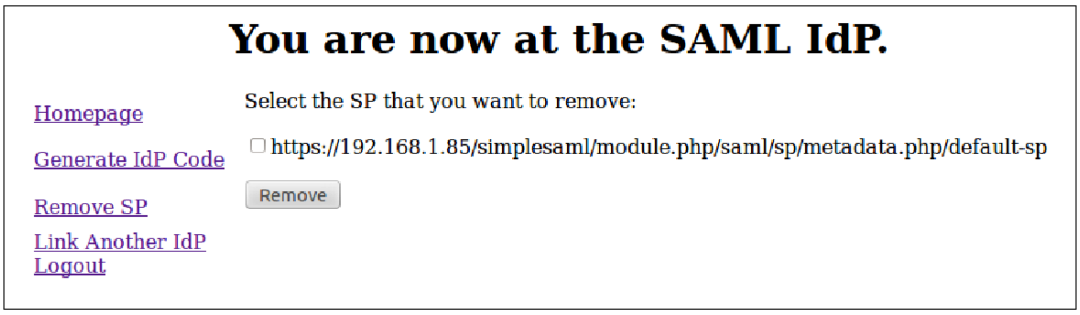

User
IdP
SP
The user logs into the IdP and clicks the "*Remove SP*" link
The IdP retrieves a list of SPs federated by the user and shows the list
The user selects the respective SP and clicks the *Remove* button
The IdP retrieves the respective code for the SP and removes the SP entry from different tables
The IdP sends a request to the SP along with the code retrieved previously to remove itself from the TAL of the SP
The SP validates the code and removes the IdP from its TAL

Figure 5.7: Protocol flow for a dynamic defederation.

1. The user logs in to the IdP and then clicks the *Remove SP* link.

2. If the user did not federate the IdP to any SP dynamically, a message will be shown to the user. Assuming, the user did already federate the IdP to some SPs, a list of SPs is retrieved from the *untrusted/semitrusted* tables using the user identifier to ensure that only the SP that has been federated dynamically by the user is retrieved. Then, a list of check boxes containing the entity ID of these dynamically added SPs is shown (Figure 5.8).



You are now at the SAML IdP.

Homepage
Generate IdP Code
Remove SP
Link Another IdP
Logout

Select the SP that you want to remove:

☐ https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

Figure 5.8: Removing an SP at the IdP.

3. The user selects one of the SPs and clicks the *Remove* button.

4. The IdP uses the entity ID of the selected SP to retrieve the code from the *untrusted/semitrusted* table.

5. Once the code is retrieved, the corresponding entry is removed from the respective table. The IdP also removes the corresponding SP entry from the *idpadmin* and *spadmin* tables.

6. Then, a request to remove the IdP entry from the TAL of the SP is sent to the entity ID of the SP. The request contains three parameters:

   - the *remove* parameter containing the entity ID of the IdP,
   - the *code* parameter containing the code retrieved in Step 4, and
   - the *ReturnTo* parameter containing the link of the Remove page at the IdP where the user will return later.

7. The SP checks if there exists the entity ID of an IdP along with the code containing the respective value in its *untrusted* table. If such an entry is found, the entry is removed from the table, otherwise an error message is shown. Moreover, the SP also removes the respective entry from the *idpadmin* and *spadmin* tables.

8. Finally, the user is redirected to the location retrieved from the *ReturnTo* parameter and the user is informed about the successful defederation.

As described in the previous use-case, the same approach has been adopted to defederate any locally installed IdP using a JavaScript embedded into an XHTML form which is submitted automatically to the entity ID of the IdP and like before the SP is assumed to be online.

Use-case: Dynamic Update

The setup for this scenario is that an IdP and an SP have been federated dynamically using the steps described in the first use-case. The *idpadmin* table at the IdP contains the entity ID of the SP along with the IdP-generated 4 digit *AdminCode* and the *idpadmin* table at the SP contains the entity ID of the IdP with the same IdP-generated 4 digit *AdminCode*. Now, the administrator of the IdP has made some changes. It is assumed that the changed entries are the *Binding Location* entries of the *SingleSignOnService* entry of the *IDPSSODescriptor* field of the metadata to the local metadata of the IdP and the administrator wants to propagate these changes into the metadata stored at the SP in a dynamic fashion. It is assumed that the IdP has provided an interface to allow the administrator to propagate these changes.

With this setup, the protocol flow, illustrated in Figure 5.9, for updating metadata dynamically is given below.

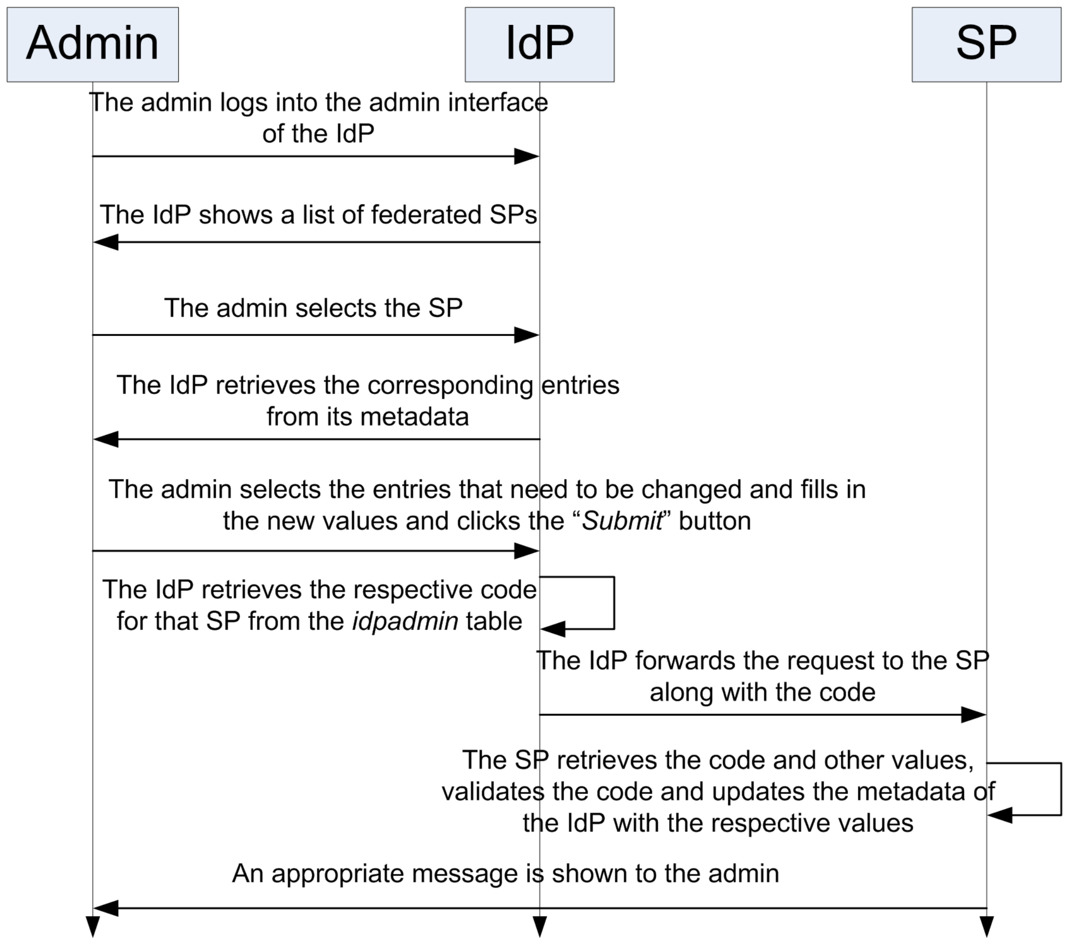

Admin
IdP
SP
The admin logs into the admin interface of the IdP
The IdP shows a list of federated SPs
The admin selects the SP
The IdP retrieves the corresponding entries from its metadata
The admin selects the entries that need to be changed and fills in the new values and clicks the "*Submit*" button
The IdP retrieves the respective code for that SP from the *idpadmin* table
The IdP forwards the request to the SP along with the code
The SP retrieves the code and other values, validates the code and updates the metadata of the IdP with the respective values
An appropriate message is shown to the admin

Figure 5.9: Protocol flow for dynamic update.

1. The administrator of the IdP visits the administrative interface of the IdP and logs in.

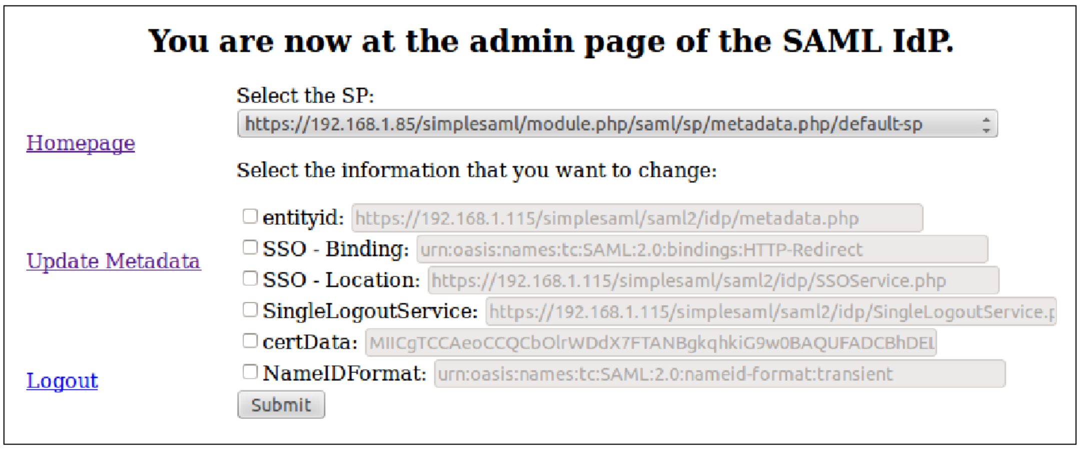
2. The administrator clicks the *Update Metadata* link at the interface and the update page is shown (Figure 5.10).

3. The update page contains a dropdown list containing the entity ID of all (dynamically or statically) SPs federated with the IdP (Figure 5.10).

4. The administrator selects the entity ID of the SP to which she wants to propagate these changes.

5. The selected entity ID of the SP is used to retrieve the respective entries of the metadata and those entries are displayed using check boxes to the administrator. The administrator selects the *Binding* and *Location* (SSO-Binding and SSO-Location entries in Figure 5.10) entries and clicks the *Submit* button. The IdP restores the *AdminCode* from the *idpadmin* table using the selected entity ID of the SP.

6. A request is sent to the entity ID of the SP. The request contains several parameters:


You are now at the admin page of the SAML IdP.

Homepage

Update Metadata

Logout

Select the SP:
https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

Select the information that you want to change:

☐ entityid: https://192.168.1.115/simplesaml/saml2/idp/metadata.php
☐ SSO - Binding: urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
☐ SSO - Location: https://192.168.1.115/simplesaml/saml2/idp/SSOService.php
☐ SingleLogoutService: https://192.168.1.115/simplesaml/saml2/idp/SingleLogoutService.
☐ certData: MIICgTCCAeoCCQCbOlrWDdX7FTANBgkqhkiG9w0BAQUFADCBhDEL
☐ NameIDFormat: urn:oasis:names:tc:SAML:2.0:nameid-format:transient

Submit

Figure 5.10: Metadata update page.

- the *update* parameter containing the entity ID of the IdP,
- the *AdminCode* parameter containing the code, and
- the name of the entries of the metadata that need to be updated each containing respective updated values and the *ReturnTo* parameter containing the location of the update page at the IdP where the administrator will return.

7. Once the SP receives this request, all values of the parameters are retrieved. The *update* parameter informs the SP that the IdP with the entity ID in that parameter wants to update its metadata. The *AdminCode* parameter is used to authorise the request by matching them with the values stored at the *idpadmin* table at the SP.

8. If they match, the metadata is updated with the requested values and an appropriate message is returned to the location retrieved from the *ReturnTo* parameter. If no match is found, an error message is returned to that IdP location.

A similar approach can be used to update the metadata of the SP stored at the TAL of the IdP. As described in the previous use-case, the metadata of any locally installed IdP can be updated using a JavaScript embedded into an XHTML form which is submitted automatically to the entity ID of the IdP.

5.4 Proof of Concept: Service Provisioning

To illustrate how users can access services in a federation created dynamically, a proof of concept has been implemented. In this section, the implementation is presented using two different scenarios: an IdP-SP Scenario to illustrate the concept for a Type 1 Federation (Figure 3.6(a)) and an IdP-IdP-SP Scenario to illustrate the concept for a Type 2 Federation (Figure 3.6(b)).

5.4.1 IdP-SP Scenario

The setup for this use-case is that an IdP and SP have been federated dynamically with the IdP tagged as untrusted at the SP and the SP is tagged as untrusted at the IdP. Before the protocol flow is discussed, underlying trust issues must be analysed.

A mechanism has been provided to ensure that an administrator of an IdP can configure which attributes are released to a semi-trusted SP. For this, a configuration parameter called *semitrusted.sp* has been added to the configuration file (called config.php) in SimpleSAMLphp. It is assumed that an IdP has asked all its users to provide values for these attributes during the registration process: *username, name, telephone, age, position, org, salarygrade* and *email*. A sample configuration parameter could be *'semitrusted.sp'=> array ('username', 'name', 'telephone', 'age', 'position', 'org')* which will configure the IdP to release only these attributes by excluding other attributes such as *salarygrade* and *email*, which the IdP normally releases to trusted SPs. The administrator can add as many or as few attributes as needed as per the requirements. This configuration parameter works like an attribute release policy. SimpleSAMLphp does not have the concept of an Attribute Release Policy like in Shibboleth, however, it has something similar called an Authentication Processing Filter (AuthProc) [105]. An AuthProc allows the system to do additional activities once user authentication is complete. For example, it can be used for filtering out attributes. There are several authentication processing filters bundled with the SimpleSAMLphp implementation. One of them is the *Consent* module that is used to display the list of attributes to the user just before they are released. This module has been modified to allow the IdP to show only those attributes that can be found on the *semitrusted.sp* parameter. From the displayed list, the user can choose which attributes she wants to release to the SP. Note that the illustrated use-cases are applicable for any locally installed IdP as well as for any traditional IdP using the mechanisms discussed previously.

With this setup, the protocol flow, illustrated in Figure 5.11, for the scenario is given below.

1. A user visits the SP for the first time to access one of its services. Assuming that the user is not authenticated, the user is redirected to the WAYF service to discover a list of IdPs federated with the SP. The list also contains the untrusted IdP federated in the previous use-case.

2. Now, if the user selects the untrusted IdP, the usual SAML flow will take place. A SAML authentication request will be sent to the selected IdP and if the user is not already authenticated at the IdP, she will be prompted for login.

3. Once the user is logged in, the *Consent* module is called internally. It reads the *semitrusted.sp* configuration parameter and displays the attributes automatically. Figure 5.12 shows the consent form. The consent page also states which attributes are

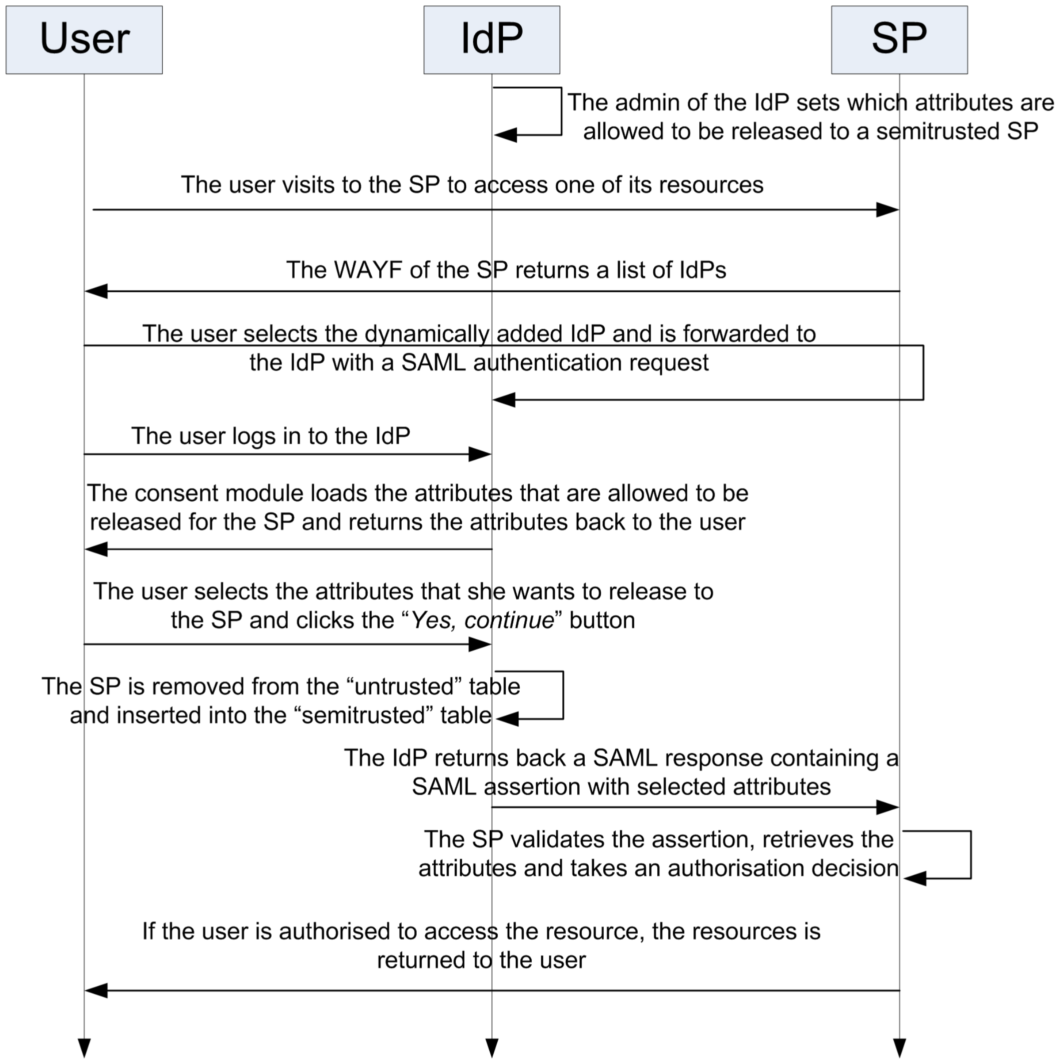

User
IdP
SP
The admin of the IdP sets which attributes are allowed to be released to a semitrusted SP
The user visits to the SP to access one of its resources
The WAYF of the SP returns a list of IdPs
The user selects the dynamically added IdP and is forwarded to the IdP with a SAML authentication request
The user logs in to the IdP
The consent module loads the attributes that are allowed to be released for the SP and returns the attributes back to the user
The user selects the attributes that she wants to release to the SP and clicks the "Yes, continue" button
The SP is removed from the "untrusted" table and inserted into the "semitrusted" table
The IdP returns back a SAML response containing a SAML assertion with selected attributes
The SP validates the assertion, retrieves the attributes and takes an authorisation decision
If the user is authorised to access the resource, the resources is returned to the user

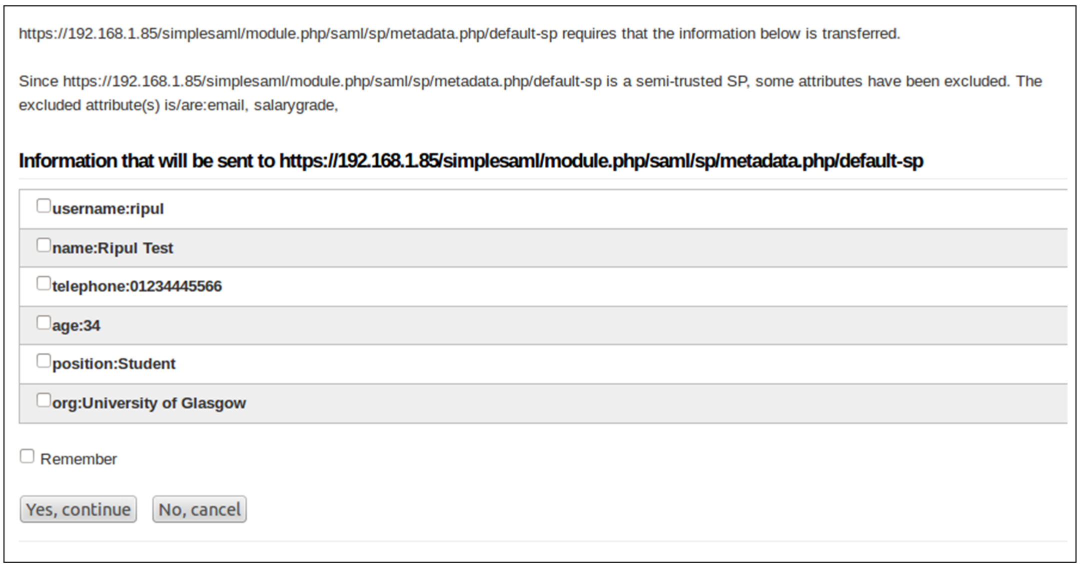
Figure 5.11: Protocol flow for IdP-SP service provisioning.

filtered out from the full set and why. At this point, the user can choose which attributes she wants to release to the SP.

4. If the user has chosen to release any attribute(s) by clicking the *Yes, continue* button, the entity ID of the SP is removed from the *untrusted* table and inserted into the *semitrusted* table indicating that the SP will be tagged as a semi-trusted entity hereafter and the chosen attributes would be released to the SP using a SAML response. If the user chooses not to release any attributes, the entity ID of the SP will remain in the *untrusted* table.

5. Once the SP receives the response, the assertion is extracted and validated. Then, attributes are retrieved from the assertion.

At this point, the SP knows from the *untrusted* table of its database that these attributes have been released by an untrusted IdP. Therefore, the respective SP will treat these attributes


https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp requires that the information below is transferred.

Since https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp is a semi-trusted SP, some attributes have been excluded. The excluded attribute(s) is/are:email, salarygrade,

Information that will be sent to https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

- ☐ **username:ripul**
- ☐ **name:Ripul Test**
- ☐ **telephone:01234445566**
- ☐ **age:34**
- ☐ **position:Student**
- ☐ **org:University of Glasgow**

☐ Remember

Figure 5.12: Attributes to be released at the Consent page.

coming from an untrusted IdP having a LoA value of 1 and authorise the user accordingly.

Evidently, the notion of trust plays a crucial role in dynamic federations and goes through different transformations while different entities interact during a use-case. The summary of these transformations is as follows. When an IdP and an SP are added dynamically, they initially consider each other as untrusted entities. When a user of the IdP, after being authenticated, releases a few attributes to the SP, the SP is considered as semi-trusted. However, a dynamically added SP is never considered as a fully trusted entity to an IdP. On the other hand, a dynamically added IdP is never considered as a fully trusted or semi-trusted entity to an SP.

5.4.2 IdP-IdP-SP Scenario

The protocol flows described above will allow the creation of federations in a fully dynamic fashion and also consider the trust issues involved at both ends. However, the main problem is that an SP may not trust all attributes coming from an untrusted IdP even though the IdP is honest. The problem can be resolved if it is possible to link the untrusted IdP with an IdP which is fully trusted by the SP. In such a case, the fully trusted IdP would act like a *Proxy IdP* as described in [106]. A Proxy IdP can delegate the authentication task to another IdP which is hidden from the SP. Thus, the SP will think that it is talking to the fully trusted IdP while in fact it is another IdP performing the authentication. The untrusted IdP would release the user attributes to the fully trusted IdP once the user is authenticated which will be then retrieved and returned to the SP in such a way that the SP will think that the attributes have been released by the fully trusted IdP. Another advantage is that the SP no longer needs to support dynamic federations, since the Proxy IdP can, in a trustworthy manner, add new IdPs indirectly. As before, SimpleSAMLphp has used and modified to demonstrate this scenario.

SimpleSAMLphp allows multiple authentication sources (including another SAML IdP) for authenticating a user. This can be enabled by the *MultiAuth* module of SimpleSAMLphp. This feature has been used to add the untrusted IdP (from the perspective of an SP) as one of the authentication sources in the fully trusted IdP. To the untrusted IdP, the fully trusted IdP would be treated as a normal SAML SP, however, the fully trusted IdP would treat the untrusted IdP as the SAML IdP. To enable this configuration, it needs the authentication source to be pre-configured by exchanging metadata just like a federation. As in the previous scenario, SimpleSAMLphp has been modified to automate this procedure so that two prior unknown SAML IdPs can be linked (in other words federated) in a fully dynamic fashion. However, this approach introduces an inconsistency which a malicious user may abuse for elevation of privileges. Since the SP would think that the attributes from the linked IdP have come from a trusted source (the Proxy IdP), they might be tricked in trusting them. To ensure that it does not happen, the trusted IdP must add a LoA value of 1 into the assertion in cases it has received any attributes from a dynamically linked IdP and return the assertion with that lower LoA. This would help the SP to decide how much trust they can put in those attributes.

With this setup, the protocol flow, illustrated in Figure 5.13, for this scenario is given below.

1. A user logs in to the untrusted IdP and generates a code as described in Section 5.3.1 . This code will be used to link the Proxy IdP.

2. Now, the user needs to log in to the Proxy IdP. Since the IdP has enabled the *MultiAuth* module, the IdP shows all authentication sources (Figure 5.14). As there is no other authentication sources, only the *example-userpass* source (an authentication source based on locally stored username/password in a database) is shown, which allows the user to log in to the IdP by using username/password. The user selects this source and logs in using her username and password. After login, the user clicks the ‘Link Another IdP’ option.

3. The user is presented with a page which has three fields:

    - IdP ID field for entering the entity ID of the untrusted IdP,
    - Code field to enter the generated code from step 1, and
    - a Name field to enter a *Nick Name* for the untrusted IdP.

    This *Nick Name* will help the user to remember the IdP once it is added as one of the authentication sources. This field is not needed at the SP because each IdP is listed using its entity ID, whereas at the Proxy IdP, IdPs are listed as authentication sources using a user friendly name. After entering all this information, the user clicks the *Submit* button.

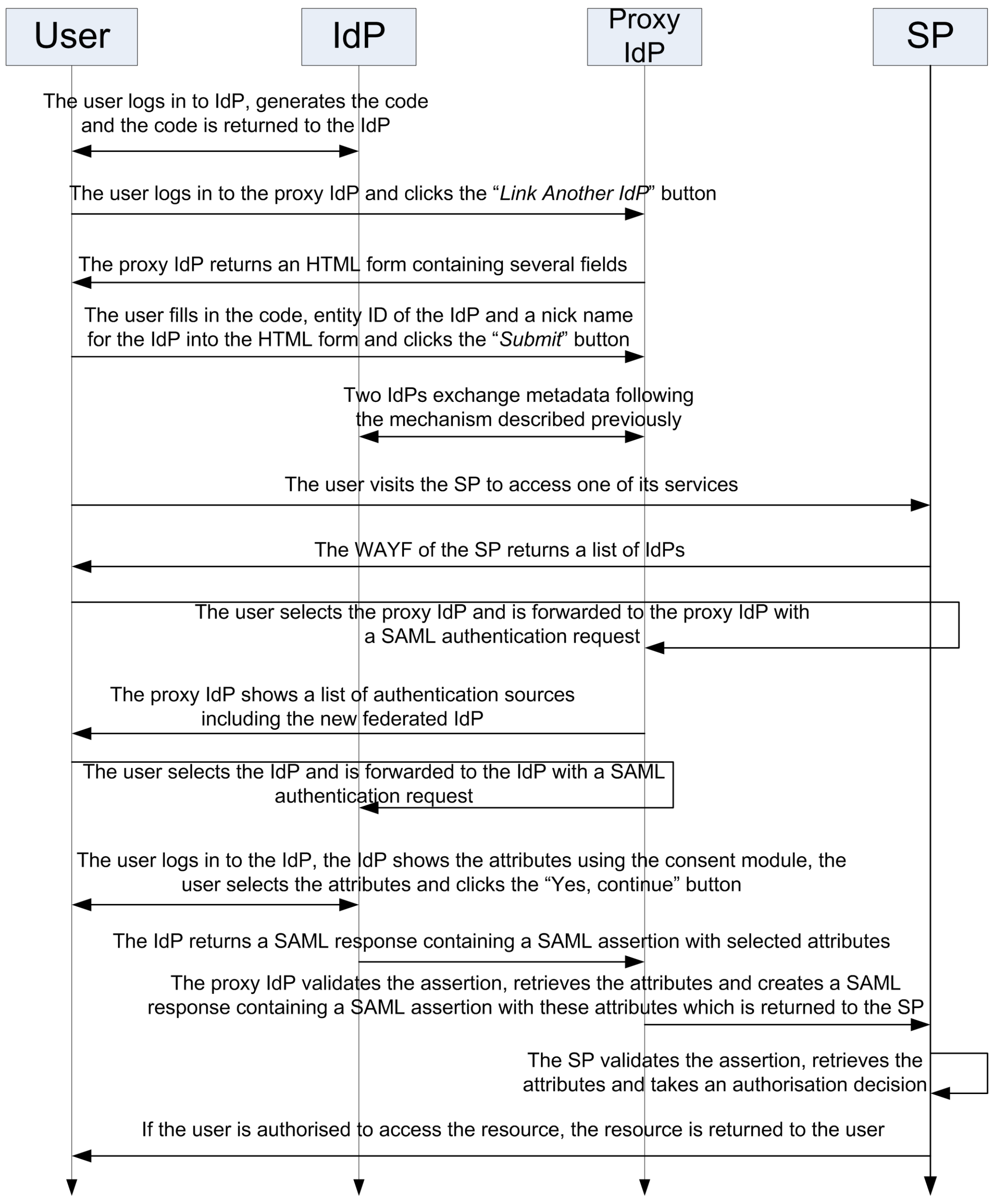
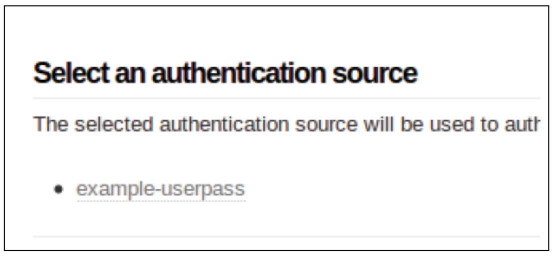

User
IdP
Proxy IdP
SP
The user logs in to IdP, generates the code and the code is returned to the IdP
The user logs in to the proxy IdP and clicks the "Link Another IdP" button
The proxy IdP returns an HTML form containing several fields
The user fills in the code, entity ID of the IdP and a nick name for the IdP into the HTML form and clicks the "Submit" button
Two IdPs exchange metadata following the mechanism described previously
The user visits the SP to access one of its services
The WAYF of the SP returns a list of IdPs
The user selects the proxy IdP and is forwarded to the proxy IdP with a SAML authentication request
The proxy IdP shows a list of authentication sources including the new federated IdP
The user selects the IdP and is forwarded to the IdP with a SAML authentication request
The user logs in to the IdP, the IdP shows the attributes using the consent module, the user selects the attributes and clicks the "Yes, continue" button
The IdP returns a SAML response containing a SAML assertion with selected attributes
The proxy IdP validates the assertion, retrieves the attributes and creates a SAML response containing a SAML assertion with these attributes which is returned to the SP
The SP validates the assertion, retrieves the attributes and takes an authorisation decision
If the user is authorised to access the resource, the resource is returned to the user

Figure 5.13: Protocol flow for IdP-IdP-SP service provisioning.

4. Once the Submit button is clicked, it is checked to make sure that all information has been entered in the three fields. If not, appropriate error messages are displayed. If yes, a request to retrieve metadata from that entity ID is submitted with some specific values just as discussed in the previous scenario.

5. At this point, the code is verified, metadata between the two IdPs are exchanged, verified and stored as discussed previously. At this point, the two IdPs are linked.

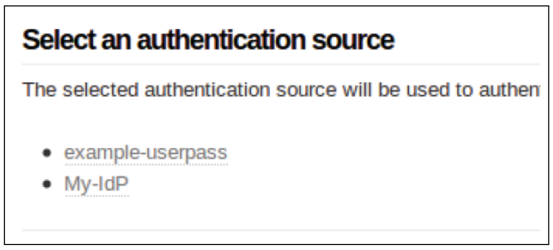
6. The user visits the SP to access one of the services. Assuming that the user is not authenticated, she is redirected to the WAYF Service where the list of federated IdPs is


Select an authentication source

The selected authentication source will be used to auth

- example-userpass

Figure 5.14: Initially, one authentication source *Username/password* at the IdP.


Select an authentication source

The selected authentication source will be used to authen

- example-userpass
- My-IdP

Figure 5.15: Linked untrusted IdP as the authentication source in the Proxy IdP.

shown. The user selects the fully trusted IdP (Proxy IdP) and a SAML Authentication request is submitted to that IdP.

7. The IdP displays the list of authentication sources. As the untrusted IdP is already linked, the user can see the nick name (My IdP) of the untrusted IdP which was given during the linking phase (Figure 5.15).

8. Once the user selects the untrusted IdP, she is redirected to the IdP where the user logs in and the consent page with attributes is shown. As the Proxy IdP is not tagged as the semi-trusted SP, all attributes that the user chooses will be released to the Proxy IdP.

9. Once the user clicks the *Yes, continue* button, a SAML assertion with all attributes is sent back to the Proxy IdP. The Proxy IdP validates the assertion and then retrieves all attributes. The Proxy IdP builds a SAML assertion with these attributes with a LoA value of 1 and returns a SAML response with the assertion to the SP.

What the SP will do with all these attributes is not further explored here.

Now, the notion of trust among different entities is summarised. The Proxy IdP and the SP consider each other fully trusted. Since, the Proxy IdP and the other IdP have been federated dynamically, they initially consider each other as untrusted entities. In addition, the Proxy IdP acts as an SP to the other IdP. Following the restrictions of dynamic federations, the Proxy IdP will be considered as an semi-trusted entity to the other IdP once a user of the other IdP releases her attributes to the Proxy IdP. On the other hand, the other IdP will always be considered as an untrusted entity to the Proxy IdP.

5.5 Discussion

The approach for creating dynamic federations offers several advantages:

- Users can create federations dynamically and whenever required. Even though it was not considered in the implementation, any IdP or SP can decide on how long it would allow the other party that has been added dynamically to remain in the federation by using a time threshold. Once the threshold is reached, the respective entry can be removed automatically from the TAL list, thus defederating the entity.

- By using separate trust domains inside the same federation for fully trusted, semi-trusted and untrusted entities, a federation can host all types and leverage the advantages of all in the same configuration. However, one must keep in mind that the types of treatment semi-trusted entities may receive will fully depend on a particular implementation.
- The proposal illustrates how the life-cycle of any identity federation can be managed dynamically in a fully automatic fashion. This approach can be extremely useful to automate the task of federation management which is currently done manually.

Currently, the implementation only deals with one level of trust for linking IdPs: IdP-IdP. It will be interesting to see if it can be extended for multi-levels so that the trust hierarchy looks something like this: IdP-IdP-IdP-...-IdP. Technical challenges for exchanging metadata may not be very difficult, however, establishing the transitive trust can be challenging. Also, the current model is based only on the SimpleSAMLphp implementation of the SAML. It is necessary to investigate how the proposals can be implemented in other SAML implementations as well to ensure its widespread adoption. The best way to achieve this is by converting the proposals into specifications and then integrating it with the official specifications to ensure consistent implementations.

5.6 Conclusion

In this chapter, the proposal for managing dynamic federations in SAML has been explored: how the life-cycle of such a federation can be fully managed in an automatic fashion and how such federations can be used to provide services to a user. Managing an identity federation, especially when it is very large, can be a laborious job if it is done manually. The proposed approach can be beneficial in such aspects as it can be used for creating, removing and updating federations dynamically and in a fully automatic fashion. Furthermore, the proposed approach requires no modification of the SAML Protocol, rather it depends on the implementation and hence, can be easily adopted to any SAML implementation.

The proposal has been implemented using the SimpleSAMLphp SAML library. To outline the applicability of the proposal, several use-cases with detailed protocol flows have been illustrated using the implementation. In addition, the trust issues involved in such scenarios have been examined. Finally, it should be noted that the issues of trust are very complex. The way the trust issues have been outlined here may not be suitable for all scenarios. For example, an IdP may be reluctant to trust any SP which is not pre-configured in the traditional way and thereby hesitant to release any attributes to it. In such cases, it will be difficult to create federations in a dynamic way and the involved IdPs and SPs will need to relax the trust requirements if they want to allow their users to take advantage of dynamic federations. However, how much relaxation it will require will depend entirely on a specific use-case.

The implementation of dynamic federations has been extensively used to realise different proof of concepts in the subsequent chapters of this thesis, and hence, becomes an integrated part in realising the concept of the proposal of User-controlled Identity Management which is explored in the next chapter.

Chapter 6

User-controlled Identity Management Systems

6.1 Introduction

In previous chapters, a number of limitations of the current Identity Management Systems have been identified. At first, the major limitations are highlighted below:

- **The effect of multiple Partial Identities.** The number of partial identities of a user keeps growing and the user finds it increasingly difficult to manage her identities which are spread across multiple IdPs.
- **Data Ownership and Controllability.** Users have limited control over their own data and there is no way for users to know if their attributes are abused or shared with third-parties.
- **Privacy Issues.** The support for the Selective Disclosure and the Explicit Consent requirements is not universal and depends entirely on the specific provider.

In addition to these, there have been a number of incidents in which sensitive user attributes were stolen from some major global companies (which also act as the IdPs for their respective services) such as Sony (77 million user accounts were affected [99]), Linkedin (more than six million passwords were stolen [100]), Apple (over 1 million Apple IDs fell into the wrong hands [101]) and users are rightfully worried about storing crucial data at these IdPs.

In this chapter, the proposal for a *User-controlled Identity Management System* is introduced to tackle these limitations. The main motivation for a User-controlled Identity Management is to empower users by giving them more control over their own attributes. To allow the desired control, a novel type of user-controlled IdP called *Portable Personal Identity Provider*

(PPIdP) has been introduced. A PPIdP is a special type of IdP hosted in a mobile device of a user and acts as the central component in User-controlled Identity Management. How the proposal of PPIdP can tackle a few of the stated limitations has been investigated. Additionally, the advantages and the current limitations of the proposal have been discussed. To show the applicability of the approach, several use-cases have been illustrated. A PPIdP imposes several functional, security, privacy and trust requirements. Therefore, it is also analysed how these requirements have been satisfied while designing and developing a PPIdP.

6.2 Definitions and Requirement Analysis

User-controlled Identity Management can be considered as an extension of the traditional concept of Identity Management. In essence, User-controlled Identity Management empowers users with the authority and control over their data which have been missing in the current setting of IdM. A few existing extensions of Identity Management may seem to offer a similar level of authority and control. One such extension known as the *User-centric Identity Management* [4, 107] will be investigated. The term User-centric Identity Management itself has two interpretations. In one interpretation, it refers to the IdM technology that will ensure that a user is placed at the centre of all protocol flows using that technology [107]. The argument is that putting a user at the centre of all protocol flows will enable the user to control the data flow ensuring that the user knows what data is released to which entities. Modern IdM technologies such as SAML and OpenID are based on this concept. Unfortunately, this does not solve the problem of data ownership and the provider has the ultimate control over user data. In addition, even though such technologies are supposed to provide more control (e.g. the selective disclosure of attributes) during the data flow, it has been found that it entirely depends on a specific implementation and many implementations simply do not offer such a facility. The findings regarding this will be presented in Chapter 8. The second interpretation of User-centric Identity Management refers to the theoretical use-case of utilising a personal device to collect and store data from different IdPs. Then, the personal device can be used to authenticate and access different services [4]. Since this is only a theoretical model, this is not clear how such a model can be realised and how such a model can be used to tackle the stated problems. Also, this interpretation does not deal with the issue of data ownership since all data is stored at an IdP. In summary, the User-centric Identity Management moves toward the right direction to ensure users have more control over their data, unfortunately, does not handle all issues regarding data ownership and controllability. User-controlled Identity Management builds upon the concept of User-centric Identity Management while adding the additional functionalities. In that sense, the User-controlled Identity Management can be thought of as an evolution of the User-centric Identity Management.

6.2.1 Definitions

At first, a definition of User-controlled Identity Management is provided.

Definition 24 *User-controlled Identity Management consists of all the functions and capabilities of the traditional Identity Management as defined in Definition 15 while allowing users the following additional functionalities:*

- *full ownership of their data so that no other entities can impose ownership of it;*
- *ultimate controllability over their data including the selective disclosure of attributes, and thus ensuring no entity can get hold of any subset of it without her consent;*
- *availability of their data so that it can be used for online services and applications in a secure manner.*

A system that can be used for User-controlled Identity Management is regarded as the User-controlled Identity Management System. Like any traditional IMS, such a system consists of three classes of entities - IdPs, SPs and Users - and respective protocols to enable interactions among entities of these classes. To equip the proposed system with capabilities of a traditional IMS, any existing model (e.g. the Federated or Open Unfederated Identity model) can be used. Unfortunately, the IdPs and the SPs in these models fall short of realising the additional functionalities. The two ways by which the additional functionalities can be realised are discussed next.

- The first way is to allow users to encrypt their data at source (e.g. at the device from where the data is being uploaded) before storing it at a provider. Then a third party (including a provider itself) will only be able to access the data when the user allows it by decrypting the data before sharing. In this manner, the provider will have no way to know which value for any attribute is being stored at their end. Ultimately, this will also undermine the possibility of abusing user data or sharing user data with any third party and thus solving a few of the existing problems (the second issue in Definition 24) and allowing users to have the ultimate controllability over their data. Unfortunately, this does not tackle the problem that arises due to the effect of multiple partial identities since users will still need to manage those partial identities across different providers as described previously. Moreover, this does not ensure data ownership. For example, a provider has the ultimate control on how and when the data is available to a user, even though the data is unusable by it. Also, if a provider is unavailable (it may go through a maintenance phase or may be forced out of business), users will not be able to access any data stored at the provider when required. In essence, this approach is promising, unfortunately, it does not solve all issues.

- The second way is to introduce a novel type of IdP that will be under the full control of a user. One way to achieve this is to ensure that users are the owners of this type of IdP. This enables a user to own any data stored in the IdP. However, the IdP must be equipped with appropriate technologies to allow users the full controllability of their data and to make these data available whenever required. In addition, such an IdP can be used to replace a significant number of IdPs where a user's data is currently stored. This will reduce the difficulty that arises from the effect of multiple partial identities.

Clearly, the second approach is more advantageous for a user and that is why this approach has been adopted. With this goal in mind, a novel type of IdP called the ***Portable Personal Identity Provider*** is introduced and defined below.

Definition 25 *A Portable Personal Identity Provider (PPIdP) is an Identity Provider that is under the full control of a single user and resides in a mobile device owned by the user. It is the user who must decide what attributes should be stored in such an IdP and which attributes should be released to which SP by this IdP.*

Since a PPIdP is hosted inside a mobile device of a user, all attributes stored in it will be owned by the user. Hence there is no chance of any abuse by an IdP. However, it must be ensured that a PPIdP is equipped with appropriate technologies to allow a user to fully control the stored attributes. Apart from this, the IdP must provide an interface to allow users to add, update and remove attributes into this IdP as well as to set a Attribute Release Policy (ARP) that dictates which attributes will be released to which SP. Moreover, it is the responsibility of the IdP to ensure that all user attributes are stored safely and securely.

Interestingly, such an IdP adds the feature of portability allowing a user to use it while on the move. Windows CardSpace, a former initiative from Microsoft, could act as a type of personal IdP [108] and was included with several versions of the Windows operating system such as Windows 7 and Windows Vista. However, it was not portable since it was only available for desktop computers. Currently, there is no such personal yet portable IdP and the implementation described here aims to fill this gap.

In short, additional functionalities of the User-controlled IMS will be realised using a PPIdP. Other entities (such as an SP and user) must be equipped with capabilities so that they can interact with the PPIdP using a protocol. Thus, the PPIdP takes the central stage for realising the concept of the User-controlled IMS.

6.2.2 Requirement Analysis

To ensure that additional functionalities are properly met and different key issues are effectively addressed, the following set of requirements, in addition to those given in the taxonomy of Chapter 4), have been formulated.

Functional Requirements.

- **Online Service Integration.** All IdM protocols are based on standard web technologies. Therefore, it needs to be ensured that a PPIdP is compatible with such web technologies and is integrable with online services.

- **Visibility.** The existing API of different IdM protocols (e.g. SAML and OpenID) have not been developed with a locally hosted IdP in mind. The general assumption is that both an IdP and SP are online and visible to each other. Since a PPIdP will be hosted inside a mobile device, this assumption does not hold. Therefore, there must be a mechanism to make a PPIdP and SP visible to each other.

- **Federation.** As discussed previously, SAML requires every entity to be a part of a federation before they can interact with each other and such a federation is established by exchanging metadata. Since a PPIdP is hosted inside a user's mobile device and is maintained by the user herself, the traditional approach of exchanging metadata cannot be used to create federations. A mechanism needs to be provided to federate a PPIdP with other entities.

- **Attribute Storage and Update.** A PPIdP must store and handle attributes of a user locally and must provide an interface to allow the user to provide new attributes and update existing ones.

- **Attribute Release Policy.** There should be an interface to set up any attribute release policy which will dictate what attributes can be released to which SPs.

- **Attribute Synchronisation.** Nowadays, many users have more than one mobile device. Some have a smart phone as well as a tablet and some have more than one smart phone and tablet. To enable a user to access the same set of online services using a PPIdP across multiple devices, a PPIdP must be installed on each of these devices. While using a PPIdP across multiple devices, a user may add, remove or update different attributes in different PPIdPs in different devices during different interactions. Thus, she may end up with inconsistent sets of attributes in different PPIdPs. To ensure consistency, there must be a mechanism to allow a user to synchronise attributes in these PPIdPs across multiple devices.

Security Requirements.

- **Secure User Authentication.** A user should be securely authenticated at a PPIdP.
- **Confidentiality, Integrity and Authenticity.** Attributes released by a user using a PPIdP should be transmitted between different entities maintaining their confidentiality, integrity and authenticity.
- **Secure Storage and Data Theft.** A PPIdP should ensure that a user's attributes are stored safely and securely having mechanisms to guard against any data theft possibilities.
- **Intentional or Accidental Corruption of Data.** A PPIdP must guard against the intentional or accidental corruption of data.
- **Name Qualification and Impersonation Issues.** A PPIdP is a locally hosted IdP and a user can have her own IdP hosted on her mobile devices. Therefore, a mechanism is required to ensure that each PPIdP is name-qualified with respect to an SP so that the SP can uniquely differentiate between different PPIdPs. Similarly, many users may end up using the same identifier (a username) in their PPIdP. Hence, it must be ensured that one user cannot impersonate others.

Privacy Requirements. The privacy of a user in an IMS can be affected if the privacy requirements of the taxonomy presented in Chapter 4 are not fulfilled. Even though the introduction of a new IMS introduces novel functional, security and trust requirements, the privacy requirements remain unchanged. This is because the proposed system will not affect the privacy of a user in any way. Hence, it needs to be ensured that the existing requirements - Support of Pseudonym, Data Minimisation, Selective Disclosure and Explicit Consent - are satisfied.

Trust Requirements. Implicit in every Identity Management is the issue of trust. Hence, it is crucial to closely examine the trust requirements that may arise while using the proposed system. The trust requirements essentially outline the level of trust each entity may have over another. A trust metric can be used to indicate the level of trust of each entity over another entity. For the proposed system, the NIST proposed LoA (Level of Assurance) value of 1 to 4 will be used as a trust metric.

6.3 Architecture

To realise the proposal, a model of PPIdP has been designed. The architecture of the model is illustrated in Figure 6.1. A PPIdP has two major components called the Personal Attribute Store (PAS) and the IdP Component. Each component is discussed in turn.

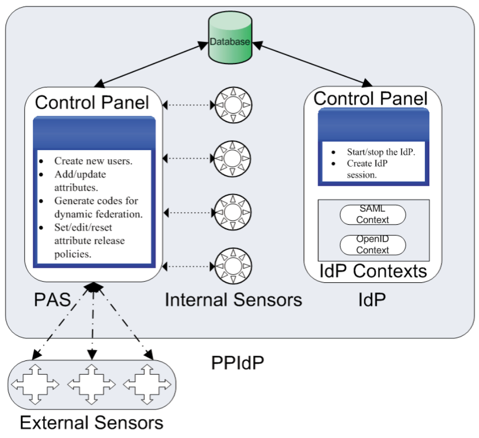


Figure 6.1: Architecture of the PPIdP.

As the name suggests, the Personal Attribute Store (PAS) is where a user's attributes are stored in encrypted format and is a necessary companion of PPIdP. It consists of a back-end database to store the attributes and an interface for users to update attributes. It also has interfaces to communicate with internal sensors (such as the GPS receiver) of mobile devices in order to collect contextual information and store them as dynamic attributes.

The IdP component is responsible for providing IdP services using different protocols (SAML, OpenID, etc.). It has two sub-components: the IdP Context and the Control Panel. The IdP context may consist of several endpoints where each endpoint serves a specific protocol. Depending on the requirements for a particular implementation, as many or as less endpoints can be added to the IdP context component. The IdP component can access the same back-end database of the PAS allowing it to retrieve user attributes while engaging in an interaction of a protocol The control panel acts as the controller allowing a user to start or stop the IdP Component whenever required.

6.4 Prototype Implementation

To show how the proposed architecture can be realised, a proof of concept prototype has been implemented. The current implementation of the prototype is based on SAML with an experimental support of OpenID. There are several popular platforms for mobile devices, namely Android, iOS, Windows and Blackberry. Since both SAML and OpenID have a Java API, Android has been chosen due to its strong support for Java. In addition, SAML SPs have been deployed with the capabilities to interact with the PPIdP. There are different implementations of SAML SPs such as Shibboleth [109], SimpleSAMLphp [11] and ZXID [110]. However, SimpleSAMLphp has been chosen because of its strong support for the selective disclosure of attributes using the *Consent* module. Next, it is discussed how different components of PPIdP have been developed in Android. Then, the challenges faced while developing these components are explored.

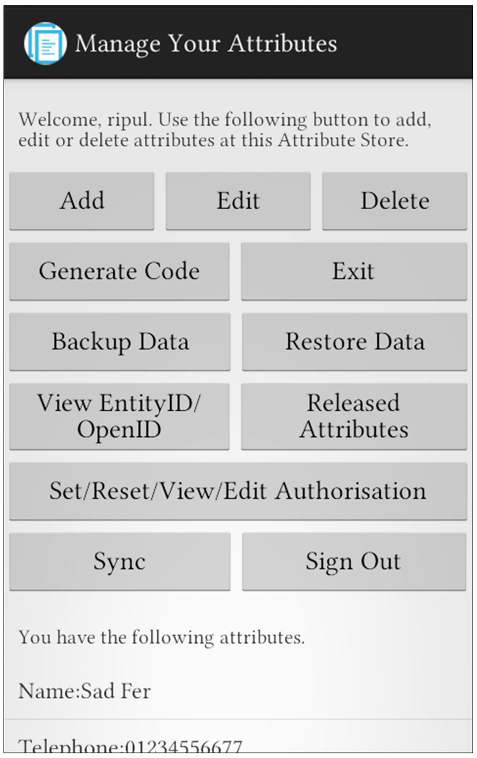
6.4.1 Personal Attribute Store

The PAS has been developed as an Android application. Android is bundled with the support for the SQLite Database [111] which is used to store user's attributes. To ensure the security of stored attributes, they are encrypted before storage. An open source library called SQLCipher [112] has bee used to encrypt attributes before storage and decrypt during usage. SQLCipher provides fast and secure 256-bit AES (Advanced Encryption Standard) encryption of SQLite database files and thus alleviating the need for manual encryption and decryption of attributes.

The first-time user of a PPIdP must create an account by providing an identifier (username) and a credential (password). The user must use that identifier (and credential) to add or manipulate other attributes and also to start the IdP. Once the user is logged in, a session is created and she is presented with the main control panel of PAS which has several options (Figure 6.2). The user can choose the *Add* option to add a new attribute, the *Edit* and *Delete* options to edit and delete existing attributes respectively, the *Exit* option to exit the PAS and the *Sign Out* option to sign out by deleting the existing session. If the user exits the PAS without signing out, the session will be kept for an hour. When the user launches the PAS next time and the session has not expired, the user will be automatically taken to the main control panel of the PAS. This saves the user having to log in every time the PAS is launched. The panel also shows the already added attributes, if any, in the bottom part of the screen. In addition, a secret key is generated as a configuration parameter by hashing the identifier and the credential of the user. The secret key is used for encryption/decryption of the user's attributes.

6.4.2 Identity Provider Component

The IdP component also has been developed as an Android app. The IdP context hosts several servlets which actually act as the endpoints. The SAML servlet is responsible for handling SAML requests and responding to them. The OpenID servlet is to handle OpenID requests. To develop the SAML servlet, the OpenSAML library [113] was considered at first. OpenSAML is a set of open source Java and C++ libraries that can be used for SAML development. However, it has quite a large number of external dependencies (more than 20 external jar files) which would increase the size of the app considerably and many of these jar files could not be compiled with the Dalvik VM (Virtual Machine) of the Android platform. To overcome this problem, a subset of the SAML protocol based on the Web Browser SSO Profile and HTTP Post binding has been developed. In this way, it has been possible to



Welcome, ripul. Use the following button to add, edit or delete attributes at this Attribute Store.

You have the following attributes.

Name:Sad Fer

Telephone:01234556677

Figure 6.2: Main control panel of the PAS.

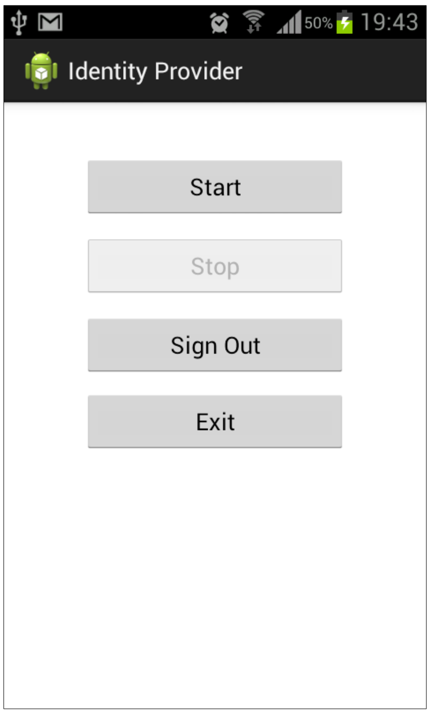


Figure 6.3: Main control panel of the IdP.

reduce the size of the app considerably. The OpenID servlet has been developed using the OpenID4Java Libraries [114].

The user can use the username/password pair of the PAS to log in to the IdP component and access the control panel. The control panel provides the option to Start/Stop the IdP component using the *Start/Stop* button (Figure 6.3). When the *Start* button is touched, the IdP component starts running in the background. Then, the user can use their preferred browser of the mobile device to access federated services. When an authentication request arrives at one of the servlets, it accesses the same back-end database (see Section 6.3.1) to retrieve the user's attributes which are then used to create responses. SimpleSAMLphp [11] and the JanRain PHP OpenID libraries [115] have been used for deploying SAML and OpenID service providers respectively.

6.4.3 Implementation Challenges

While developing the PPIdP, a number of key challenges has been handled. The challenges are discussed below.

Web Server

The first challenge was to find a suitable web container for the Android platform. After a thorough search, Jetty Web Server, an open source HTTP Server and Java Servlet Container [116], seems to be the only candidate that meets the requirements.

Visibility Issues

The second challenge was to solve the problem of visibility between a PPIdP and an SP. In the traditional setting, both IdPs and SPs are online and visible to each other. In this case, the PPIdP is installed in the device as a local web server and therefore is not visible to an SP. This problem has been solved by embedding an XHTML (EXtensible HyperText Markup Language) form submitted to the IdP using JavaScript. Since JavaScript runs on the browser, it is possible to contact the respective IdP component using the *localhost* URL. This is a standard technique used in the SAML SSO Browser profile. The same behaviour has been adopted by modifying the code of SimpleSAMLphp and JanRain PHP libraries for the implementation. Note that a PPIdP has no problem interacting with an SP since the SP has to be online to offer services.

SAML Trust Issues and Dynamic Federation

As discussed in Chapter 2, trust between an IdP and an SP in SAML needs to be established through exchanging metadata before they can interact with each other. This is accomplished by the system administrators. This imposes a problem with respect to a PPIdP since a PPIdP will be used by a user who does not have the provision to exchange metadata as an administrator. This means that a mechanism needs to be provided for federating a PPIdP with an SP by exchanging metadata, and thus establishing the required notion of trust. In addition, the mechanism must be accomplished in an automated fashion so that users are not burdened with the task of manual metadata exchange. Both of these issues are addressed by allowing users to create federations in a dynamic fashion as discussed in Chapter 5.

Essentially, the proposed approach for creating and maintaining a dynamic federation can be used to federate a PPIdP with an SP. Once an SP is federated with a PPIdP, it creates a novel type of federation which is called the **Personal Identity Federation** (PIF). In a way, this is an extension of the traditional Identity Federation and also can be of two types (Type 1 and Type 2 PIF). Figures 6.4 and 6.5 illustrate two types of PIF. In the first type (Type 1), a traditional IdP is replaced by a PPIdP. On the other hand, the second type allows a user to federate two IdPs (a PPIdP and a traditional IdP) in such a way that the user can import dynamic attributes from the PPIdP to the IdP. Then those attributes can subsequently be passed as attributes to an SP. The middle IdP (IdP1 in Figure 6.5) in this setting is known as the *Proxy IdP* and is assumed to be fully trusted by the SP with a mutual trust agreement. On the other hand, the PPIdP will be considered as untrusted by both the Proxy IdP and SP following the condition of the dynamic federation as explained in Chapter 5.

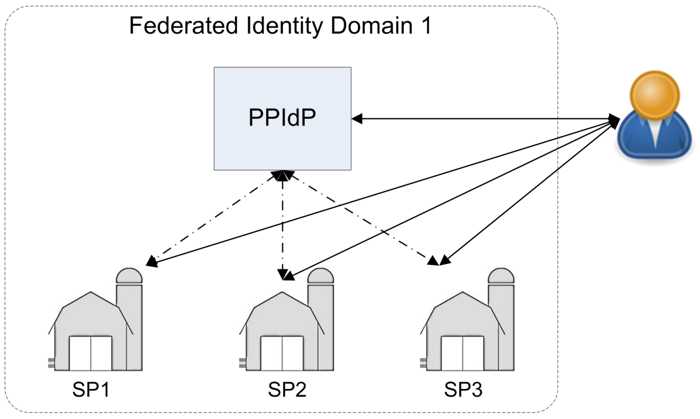


Figure 6.4: PIF Type 1.

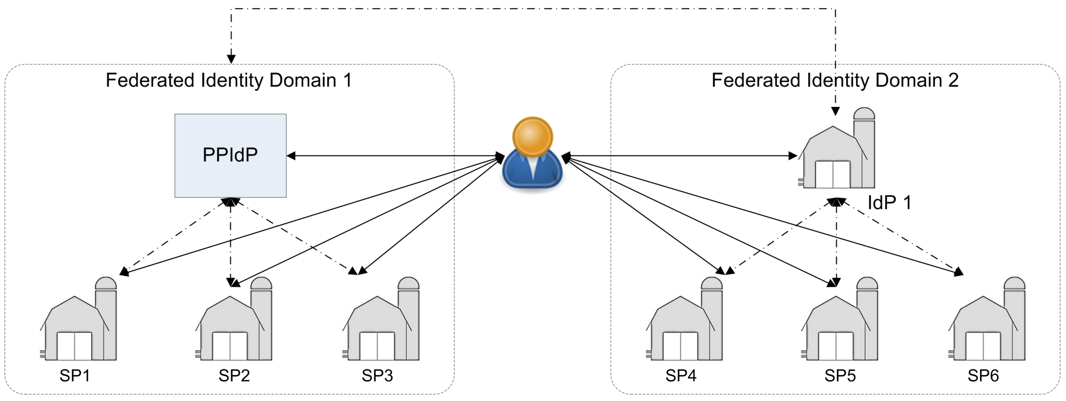


Figure 6.5: PIF Type 2.

6.4.4 SAML Protocol Flow

For accessing federated services from a SAML SP using a PPIdP, the PPIdP needs to be federated with the SP by utilising the previously described mechanisms for creating a dynamic federation. The protocol flow for creating such a federation and accessing services from the PPIdP is illustrated in Figure 6.6 and is discussed below:

1. First, a PPIdP needs to create a Type 1 PIF with an SP. For this, the user visits the PAS of the PPIdP and generates a temporary code.

2. The user visits the SP and she is forwarded to the WAYF page of the SP. Therefore, the user is presented with a form to allow her to create a dynamic federation.

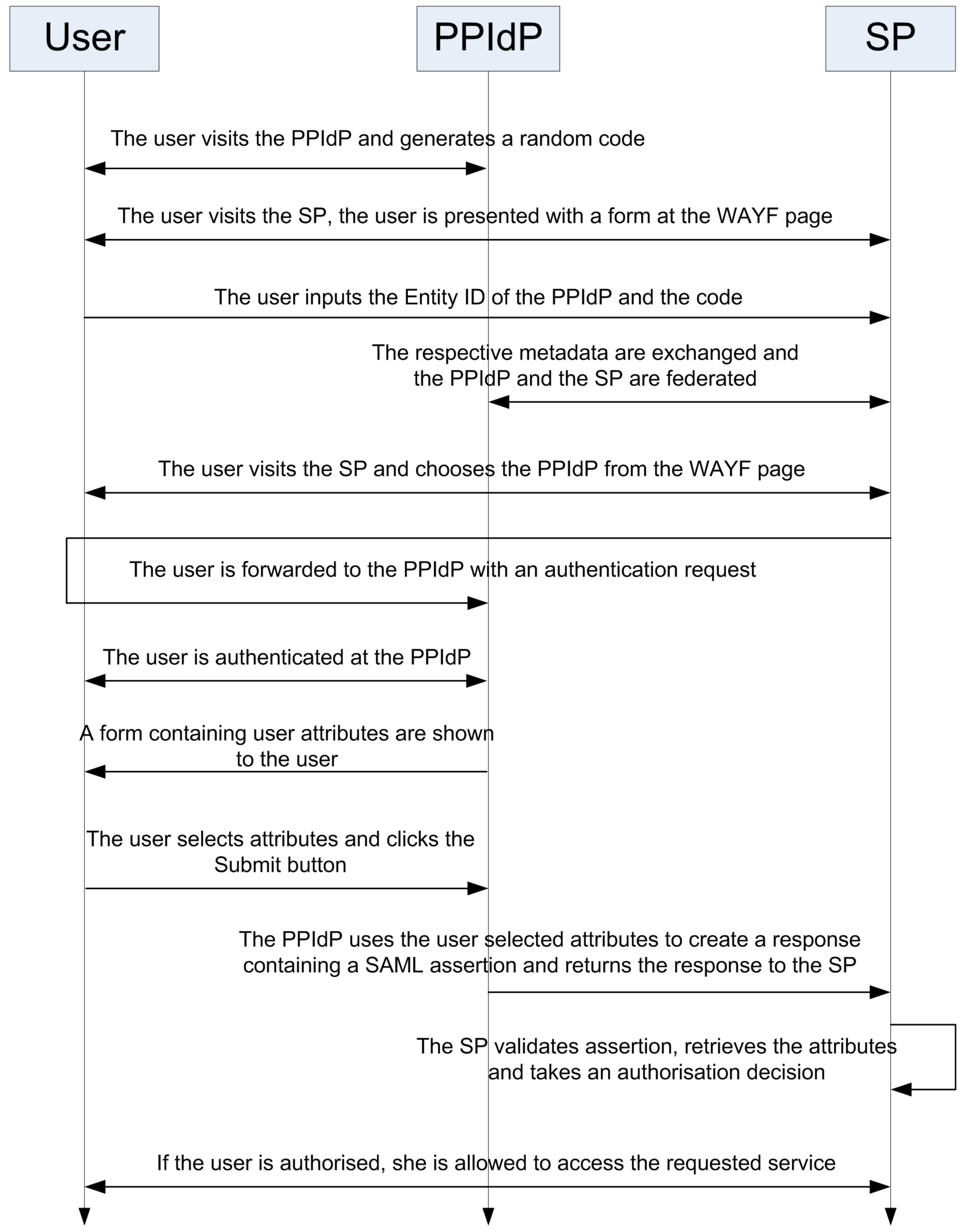
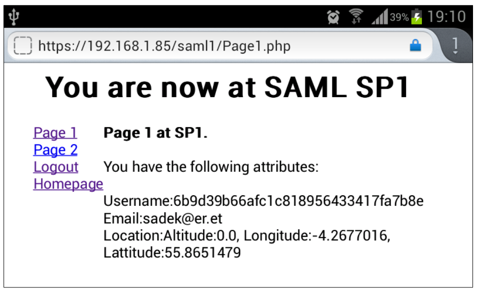

User
PPIdP
SP
The user visits the PPIdP and generates a random code
The user visits the SP, the user is presented with a form at the WAYF page
The user inputs the Entity ID of the PPIdP and the code
The respective metadata are exchanged and the PPIdP and the SP are federated
The user visits the SP and chooses the PPIdP from the WAYF page
The user is forwarded to the PPIdP with an authentication request
The user is authenticated at the PPIdP
A form containing user attributes are shown to the user
The user selects attributes and clicks the Submit button
The PPIdP uses the user selected attributes to create a response containing a SAML assertion and returns the response to the SP
The SP validates assertion, retrieves the attributes and takes an authorisation decision
If the user is authorised, she is allowed to access the requested service

Figure 6.6: Protocol flow in SAML.

3. The user inputs the entity ID of the PPIdP along with the temporary code.
4. The previously described mechanism for exchanging metadata takes place and in the end, the PPIdP and the SP becomes a part of the PIF.
5. The user visits the SP using her preferred mobile browser.
6. The SP forwards the user to the WAYF Page where she chooses the PPIdP.
7. The SP creates a SAML authentication request and forwards the request to the PPIdP. As discussed before, the request is submitted automatically to the locally hosted PPIdP using an embedded XHTML form and JavaScript.
8. The respective end point of the PPIdP receives the request and presents a login form to the user.
9. The user logs in using her username/password.
10. Once the authentication is successful, a form with all user attributes is presented to the user. The PAS has the option to create a pre-configured Attribute Release Policy (ARP) for any specific SP using the *Set/Reset/View/Edit Authorisation* option of the PAS main control panel (Figure 6.2). The PPIdP will use the ARP to filter out any attributes before they are shown to the user. The user selects the attributes that she wants to release to the SP and clicks the *Submit* button.
11. A SAML assertion including the selected attributes is created and returned to the SP.
12. The SP validates the assertion, retrieves the user attributes and makes an authorisation decision with respect to the user accessing the requested service (Figure 6.7).

Since the PPIdP has been federated with the SP using the mechanism of a dynamic federation, the SP must treat it as an untrusted entity even if the IdP acts honestly. In this way, the SAML implementation of the IdP will essentially act equivalently to OpenID. Thus, this technique will be suitable for general SPs which do not require user attributes coming from a highly trusted IdP. Interestingly, some attributes (such as credit card information) are self-certifiable. It does not matter if they are coming from a highly trusted IdP or an untrusted IdP since the SP can verify the authenticity of those attributes, and hence use them to decide whether to allow users to access sensitive services.
Another scenario has been deployed in which the PPIdP can be federated with a trusted IdP to create a Type 2 PIF. Here, the fully trusted IdP would act like a Proxy IdP as described in [106] and would delegate the authentication task to the PPIdP which is hidden from the SP. The PPIdP will essentially act as an authentication source for the trusted IdP (Figure 6.8). A SAML IdP and a SAML SP have been deployed using simpleSAMLphp. Their metadata has


https://192.168.1.85/saml1/Page1.php

You are now at SAML SP1

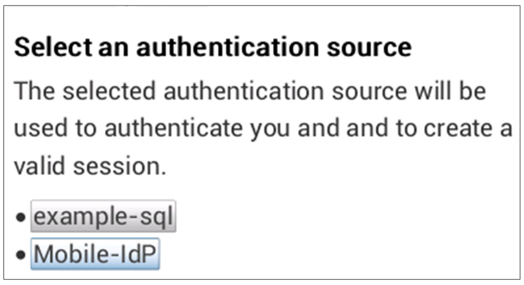
Page 1
Page 2
Logout
Homepage

Page 1 at SP1.

You have the following attributes:

Username:6b9d39b66afc1c818956433417fa7b8e
Email:sadek@er.et
Location:Altitude:0.0, Longitude:-4.2677016, Lattitude:55.8651479

Figure 6.7: Released attributes in Type 1 Federation.


Select an authentication source

The selected authentication source will be used to authenticate you and and to create a valid session.

- example-sql
- Mobile-IdP

Figure 6.8: Two authentication sources at the trusted IdP.

been exchanged beforehand to ensure they are considered fully trusted to each other. The PPIdP has been added to the trusted IdP as an authentication source using the mechanism of dynamic federations. With this setting, the protocol flow for this deployment is illustrated in Figure 6.9 and discussed below:

1. The user visits the SP using her preferred mobile browser to access its services.
2. The user is forwarded to the WAYF page where the user selects the trusted IdP.
3. A SAML authentication request is sent to the trusted IdP and the user is presented with a selection of authentication sources: The PPIdP (*Mobile IdP* in Figure 6.8) or the trusted IdP itself (*example-sql* in Figure 6.8).
4. Assuming the user selects the PPIdP, a SAML authentication request is forwarded to the PPIdP and the usual SAML protocol flow takes place.
5. A SAML assertion with the selected attributes is sent back to the trusted IdP.
6. The trusted IdP validates the assertion, retrieves the attributes and creates a SAML assertion with those attributes. To reflect the fact that these attributes have come from the PPIdP, it also inserts a LoA value of 1 for this assertion.
7. A response with the assertion is sent back from the trusted IdP to the SP.
8. The SP validates the assertion, retrieves the attributes and determines if the user is authorised to access the requested service using the released attributes with a LoA value of 1 (Figure 6.10). If the user had chosen the trusted IdP, the assertion would contain a LoA value of 2 to indicate a higher trust on the trusted IdP. This could have allowed users to access more restricted services (Figure 6.11).

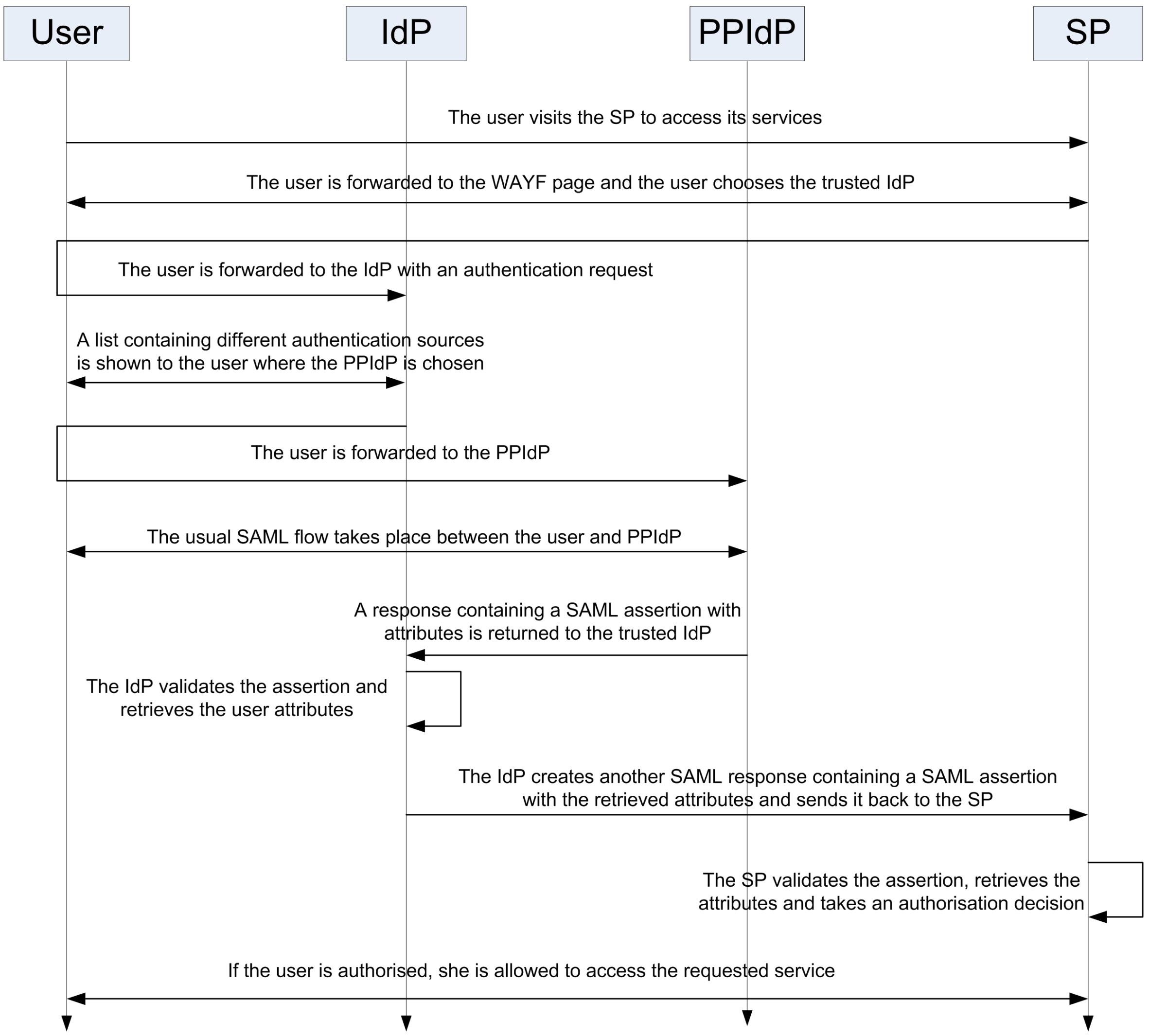
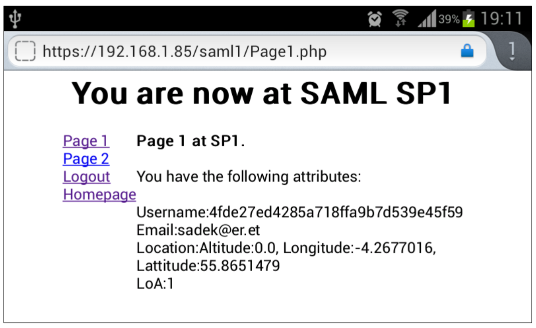

User
IdP
PPIdP
SP
The user visits the SP to access its services
The user is forwarded to the WAYF page and the user chooses the trusted IdP
The user is forwarded to the IdP with an authentication request
A list containing different authentication sources is shown to the user where the PPIdP is chosen
The user is forwarded to the PPIdP
The usual SAML flow takes place between the user and PPIdP
A response containing a SAML assertion with attributes is returned to the trusted IdP
The IdP validates the assertion and retrieves the user attributes
The IdP creates another SAML response containing a SAML assertion with the retrieved attributes and sends it back to the SP
The SP validates the assertion, retrieves the attributes and takes an authorisation decision
If the user is authorised, she is allowed to access the requested service

Figure 6.9: PIF Type 2 Protocol flow in SAML.



https://192.168.1.85/saml1/Page1.php

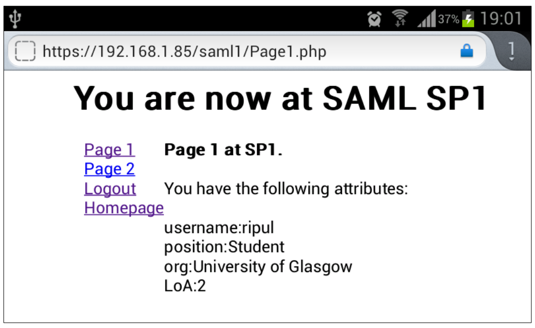
You are now at SAML SP1

Page 1 at SP1.

You have the following attributes:

Username:4fde27ed4285a718ffa9b7d539e45f59
Email:sadek@er.et
Location:Altitude:0.0, Longitude:-4.2677016,
Lattitude:55.8651479
LoA:1

Figure 6.10: LoA 1 for attributes from the PPIdP.



https://192.168.1.85/saml1/Page1.php

You are now at SAML SP1

Page 1 at SP1.

You have the following attributes:

username:ripul
position:Student
org:University of Glasgow
LoA:2

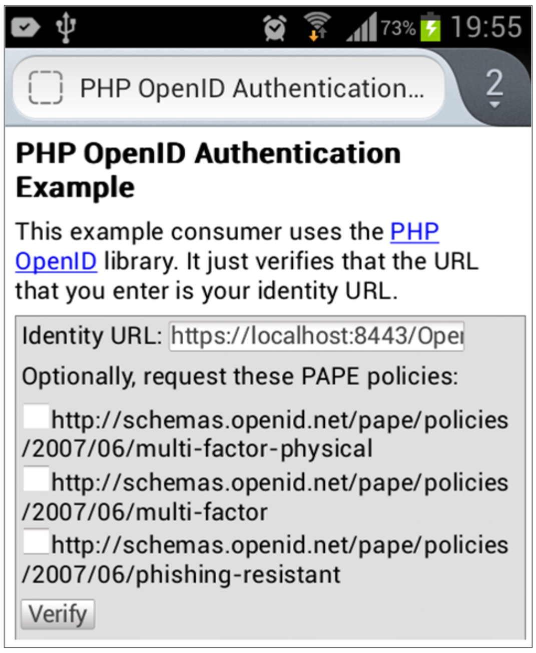
Figure 6.11: LoA 2 for attributes from the trusted IdP.

6.4.5 OpenID Protocol Flow

Compared to SAML, OpenID is a simple protocol. There are no complex trust assumptions to consider as every OpenID provider is considered trusted. The only problem that needs to be considered is the visibility issue between a PPIdP and an SP (RP in OpenID) as discussed earlier. To handle this problem, a specific implementation of OpenID has been modified so that the SP can communicate with the IdP in much the same way that the metadata exchange is accomplished in SAML. The Janrain OpenID Libraries for PHP [115] has been used to implement the OpenID SP. For this, the code has modified so that IdP discovery, authentication requests and verification requests are submitted to the IdP using JavaScript. Note that the responses from the IdP do not require any special treatment since the SP is visible to the IdP and hence the responses can be returned in the normal way.

The servlet of the OpenID context of a PPIdP has been developed using the OpenID4Java Libraries which is a set of Java libraries to develop an OpenID Provider and SP. The flow for accessing OpenID services using a PPIdP is similar to what was illustrated in Chapter 2 (Figure 2.2) and is as follows:

1. The user visits the SP using her preferred mobile browser to access one of its services and she is given the option to enter her OpenID.
2. The SP needs to authenticate the user. As it has support for OpenID, it allows the user to enter her OpenID and it has a special button tagged as *Sign in with OpenID*. Once clicked, it will allow the user to enter her OpenID.
3. The user enters her OpenID (https://localhost:8443/OpenIDTest/Server) and touches the *Verify* button (Figure 6.12(a)).
4. The PPIdP is discovered using the mechanism described above (via JavaScript). During the discovery process, the optional key exchange step has been skipped.
5. The SP creates an authentication request which is submitted to the IdP using the same approach.
6. The PPIdP receives the request and forwards it to the appropriate servlet.
7. The authentication of the user takes place using a username and password.
8. Once authenticated, an authentication response is generated and returned to the SP.
9. The SP then sends a signature verification request to the PPIdP where the signature is verified as per the OpenID specification.
10. A verification response is generated and is sent back to the SP.
11. At this point, the user is considered as authenticated (Figure 6.12(b)). What the SP will do with the authenticated user is not explored further.


PHP OpenID Authentication...

PHP OpenID Authentication Example

This example consumer uses the PHP OpenID library. It just verifies that the URL that you enter is your identity URL.

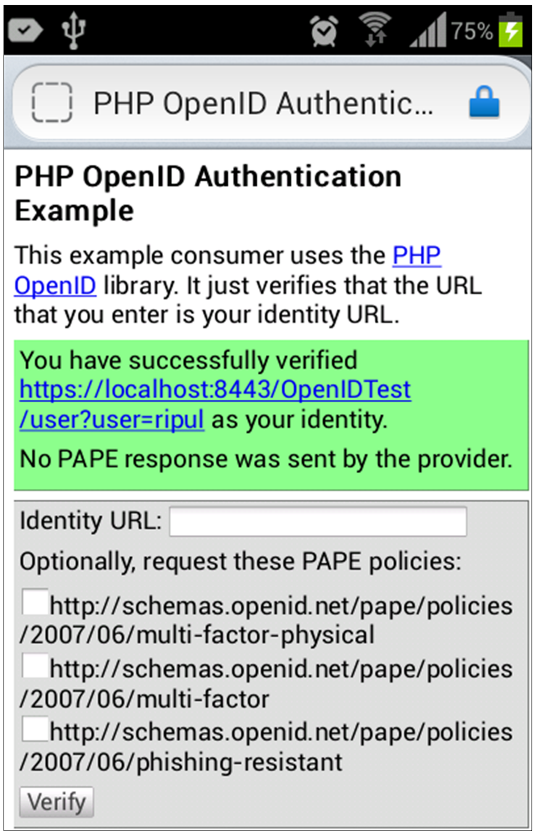
Identity URL: https://localhost:8443/Ope

Optionally, request these PAPE policies:

- http://schemas.openid.net/pape/policies/2007/06/multi-factor-physical
- http://schemas.openid.net/pape/policies/2007/06/multi-factor
- http://schemas.openid.net/pape/policies/2007/06/phishing-resistant

Verify

(a) Entering the OpenID.


PHP OpenID Authentic...

PHP OpenID Authentication Example

This example consumer uses the PHP OpenID library. It just verifies that the URL that you enter is your identity URL.

You have successfully verified https://localhost:8443/OpenIDTest/user?user=ripul as your identity.

No PAPE response was sent by the provider.

Identity URL:

Optionally, request these PAPE policies:

- http://schemas.openid.net/pape/policies/2007/06/multi-factor-physical
- http://schemas.openid.net/pape/policies/2007/06/multi-factor
- http://schemas.openid.net/pape/policies/2007/06/phishing-resistant

Verify

(b) OpenID Verified.

Figure 6.12: OpenID service access.

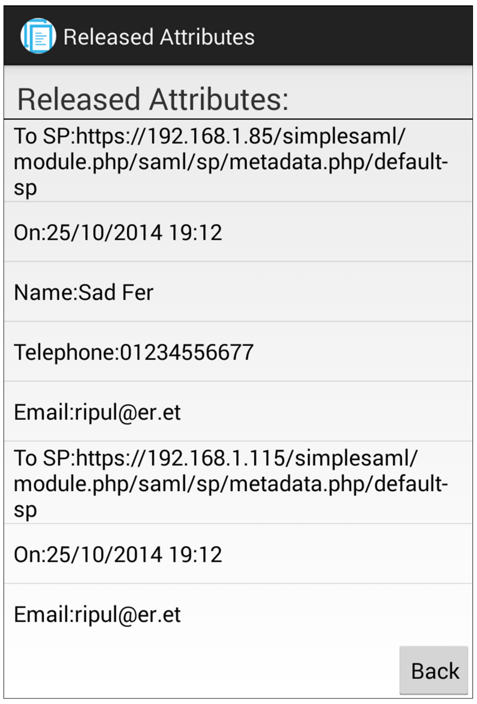
6.5 Analysis

This section investigates whether PPIdP satisfies the requirements of Section 6.2.2.

6.5.1 Functional Requirements

Since the PPIdP is based on SAML, it automatically satisfies the functional requirements (of the taxonomy given in Chapter 4) that SAML satisfies. Each user in the PPIdP is identified and represented using a username and a context is identified as an application domain. A PPIdP also satisfies two other requirements - Transaction Logging and History Management - which are not satisfied by SAML. How these two requirements as well as other additional functional requirements outlined in this chapter are satisfied by the PPIdP is discussed below.

- **Transaction Logging and History Management.** Each transaction using a PPIdP is logged into the database. Whenever any attributes are released by a PPIdP to an SP, the name of the SP and the list of attributes along with the time are stored in the database by the IdP component. The user can view each transaction by clicking (touching) the *Released Attributes* button from the control panel of the PAS. The log shows which attributes have been released to which SP at what time. Figure 6.13 illustrates an example of the logged transactions at the PAS. The user also has the option of deleting any of the stored transactions whenever she wishes to.


Released Attributes

Released Attributes:

To SP:https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

On:25/10/2014 19:12

Name:Sad Fer

Telephone:01234556677

Email:ripul@er.et

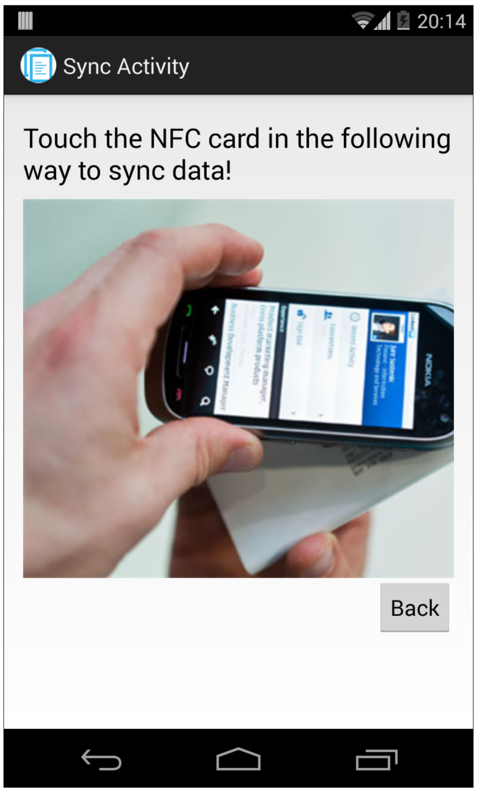
To SP:https://192.168.1.115/simplesaml/module.php/saml/sp/metadata.php/default-sp

On:25/10/2014 19:12

Email:ripul@er.et

Back

Figure 6.13: Logged transactions at the PAS.


20:14

Sync Activity

Touch the NFC card in the following way to sync data!

Back

Figure 6.14: Attribute synchronisation at PAS.

- **Online Service Integration.** Using the proposed approach for dynamic federations and the modified SimpleSAMLphp implementation, a PPIdP can be federated with any SP that provides online services. This enables a user of a PPIdP to access online services provided by any SP using the PPIdP.

- **Visibility.** As explained previously, the visibility issues between a PPIdP and an SP have been tackled using the automatic submission of the XHTML form with JavaScript.

- **Federation.** A PPIdP is federated with any SP using the proposed approach of Dynamic SAML as discussed previously.

- **Attribute Storage and Update.** A PAS is used to provide an interface for a user to add, update and remove any user attributes which are then stored in a database hosted in the mobile device.

- **Attribute Release Policy.** A PAS provides an interface for a user to set up attribute release policies which can be accessed by touching the *Set/Reset/View/Edit Authorisation* button from the control panel of the PAS (Figure 6.2).

- **Attribute Synchronisation.** To enable attribute synchronisation across multiple devices, a contactless smartcard has been utilised. A contactless smartcard is a pocket-sized card with an embedded micro-chip allowing the secure storage of data and has the capability to communicate with a reader via radio waves using Near Field Communication (NFC) technology [117]. The NFC is a standard technology for wireless communication between NFC-enabled devices which are a few centimetres apart and is used by contactless smartcards for communication [118]. The technologies for such smartcards have been defined in the ISO/IEC 14443 standard [119, 120, 121, 122]. The wireless smartcard itself can be of two types: Type A and Type B. The main differences between the two types are the variation in modulation techniques, protocol initialisation methods and coding schemes.

  Many modern smart devices are NFC-enabled allowing them to communicate with wireless smartcards and other NFC-enabled devices. In particular, since 2011 Android OS has had extensive support for NFC and has been the primary platform for popularising NFC. It allows any app to leverage the NFC capability of a device and supports both Type A and Type B smartcards. Apple made the NFC-enabled iPhone 6 in 2014, however, the NFC chip is only accessible for a specialised application scenario called *Apple Pay* which allows a user to use her iPhone to make payments wirelessly. The extensive support for NFC on Android makes it an ideal platform to explore the possibility of attribute synchronisation in different PPIdPs across multiple devices.

  Since a PAS is the central place from where attributes are managed, the support for attribute synchronisation with NFC has been added to the PAS. A user can access this feature by touching the *Sync* button from the control panel of the PAS (Figure 6.2). Once touched, a new view is shown to the user (Figure 6.14) from where the user can touch their smartcards to the back of their mobile devices to initiate the synchronisation step. During synchronisation, attributes (and their values) are encrypted using the 256-bit AES algorithm and the same secret key that is used for storing encrypted attributes at the database. Then, the encrypted attributes are transferred to the smartcard. This ensures that the attributes in the smartcard remain secure even if the card gets stolen or falls into the wrong hands. To enable synchronisation, an additional table, called *timestamp*, is maintained in the database of the PAS which stores the timestamp of the last action regarding the addition, removal or updating of attributes in the database. With this setup, the synchronisation step is discussed below using several use-cases.

  Use-case 1: Non-empty database – Empty smartcard. This depicts the scenario when the smartcard is used for the first time and hence it is empty; meaning that it does not contain any user attributes. Once the card is touched to the back of the device from the Attribute Synchronisation view (Figure 6.14), user attributes are concatenated along with the timestamp (retrieved from the *timestamp* table), then encrypted and finally transferred to the card. Once completed, a message is shown to the user.

Use-case 2: Empty database – Non-empty smartcard. This depicts the scenario when all data from the database of the PAS has been accidentally deleted or when the user wants to synchronise attributes to a new mobile device where the PPIdP and the PAS have just been installed and hence the database is empty. For the first scenario, it is assumed that a user account and a secret key have already been created and the user has already synchronised attributes from the database to the smartcard using the first use-case. Once the attributes have been synced to the card, all attributes in the database were mistakenly deleted. Now, the user wants to synchronise the data back to the database. To achieve this, the card is touched to the back of the device from the Attribute Synchronisation view (Figure 6.14) and encrypted user attributes and the timestamp are transferred from the card to the PAS where they are decrypted. Then, the timestamp and user attributes are retrieved from the decrypted text. The timestamp is used to update the *timestamp* table and attributes stored in encrypted format in the database using mechanisms discussed before. For the second scenario, the user must use the same username and password for the second PAS to ensure that the secret key generated for attribute synchronisation is same as that of the PAS of the first device. This can be achieved without problems since two devices essentially emulate two different application domains where the same user can have the same username; just like a user can have multiple accounts in different web domains using the same identifier. With this assumption, the attribute synchronisation process from the card to the PAS is exactly same as the first scenario. An appropriate message is shown at the end of each synchronisation step.

Use-case 3: Non-empty database – Non-empty smartcard. This depicts two scenarios:

1. when a user has accidentally deleted an attribute (or multiple attributes) from the database of the PAS of one of her devices and wants to restore the attribute (or attributes).
2. when a user has added, updated or removed attributes from the database of the PAS of one of her devices and hence the card or the PAS of another device needs to be updated accordingly.

This is not a trivial step like the first two use-cases as there must be an indicator that can be used to determine the latest state between the card and the database of the PAS. The value of the timestamp is used as the indicator. Once the card is touched to the back of the device from the Attribute Synchronisation view (Figure 6.14), a dialog appears with the question: “Would you like to restore attributes in PAS?”. If a user touches *Yes*, then the first scenario is assumed. Encrypted user attributes and the timestamp are transferred from the card to the PAS where they are decrypted. Then, the timestamp and user attributes are

Table 6.1: Different properties of the used NFC card.

Property	Value
Type	ISO/IEC 14443 Type A
Chipset	Topaz NFC IC
Size	512 Byte

retrieved from the decrypted text. Also, user attributes are retrieved from the database. These two sets of attributes are compared side by side to determine which attributes in the database need to be restored and based on the comparison, the respective attributes are restored. Also the *timestamp* table of the database is updated with the timestamp value retrieved from the card.

If a user touches *No*, then the second scenario is assumed. Encrypted user attributes and the timestamp (denoted t^c) are transferred from the card to the PAS where they are decrypted and retrieved as before. Also, the timestamp value (denoted t^m) is retrieved from the *timestamp* table of the PAS. With these two timestamps, there are three possible use-cases.

– If $t^m > t^c$, it depicts the scenario that the card needs to be synchronised with attributes from the database, user attributes are concatenated along with the timestamp (retrieved from the *timestamp* table), encrypted using the approach discussed previously and finally stored on the card.

– If $t^m < t^c$, it depicts the scenario that the user is trying to synchronise the database of another device with the value from the first device using the card, encrypted user attributes and the timestamp are transferred from the card to the PAS where they are decrypted. Then, the timestamp and user attributes are retrieved from the decrypted text. Also, user attributes are retrieved from the database. These two sets of attributes are compared side by side to determine which attributes in the database need to be added, updated or removed. Based on the comparison, the respective attributes are added, removed or updated. Also the *timestamp* table of the database is updated with the timestamp value retrieved from the card.

– If $t^m = t^c$, it depicts the scenario that both databases are synchronised, no further step is taken. An appropriate message is shown at the end of each synchronisation step.

The current implementation of the PAS is functional for both Type A and Type B cards. The functionalities have been tested by installing the PPIdP in two Nexus 4 (NFC-enabled Android) devices and by emulating the outlined use-cases with a Type A NFC card. The different properties of the used NFC card are given in Table 6.1.

6.5.2 Security Requirements

Next, it is analysed how the security requirements for PPIdP are satisfied.

- **Secure User Authentication.** A PPIdP is installed in a mobile device of a user, meaning that the user is authenticated locally inside the device and the identifier and the related credential of the user for the PPIdP are never transmitted over the network. This ensures the secure authentication of a user at a PPIdP.
- **Confidentiality, Integrity and Authenticity.** PPIdP uses encrypted communication channels (HTTPS) during different protocol flows as per the specification of SAML and OpenID. The encrypted channel guarantees the confidentiality, integrity and authenticity of information passed between involved entities.
- **Secure Storage and Data Theft.** As mentioned previously, all user attributes are stored in encrypted formats using the SQLCipher library in the database as well as in the NFC card. As all user attributes are stored centrally in a database, it needs to be ensured that user attributes are stored safely and securely. This is ensured with the help of the SQLCipher library which encrypts every attribute stored in the database. The database is saved in the external storage of the android phone to allow both the PAS and PPIdP to access the database. Because of the use of the SQLCipher Library with a secret key (as described in Section 6.4.1) that only the PAS and PPIdP know, the whole database will be inaccessible by other applications. This removes any threats involving other applications or third parties accessing the database. Similarly, user attributes are stored on the smartcard in encrypted formats. The security of the system can be increased furthermore by storing attributes in a hardware/software based TPM (Trusted Processing Module). However, the availability of such TPMs for the Android platform is currently limited. Therefore, it will take some time for any wide-spread adoption of this mechanism.
- **Intentional or Accidental Corruption of Data.** Since the database acts as the central repository, it is essential to guard against any intentional or accidental corruption of the database. To ensure redundancy, a backup database is always maintained in the internal storage of the PPIdP and PAS which are private to the applications and are inaccessible to other applications or attackers. The data in the backup database are stored in encrypted format just like the main database. Every single operation (addition/updating of attributes) is synchronised across both databases. In case the main database gets corrupted, the PAS provides the option of allowing the user to restore attributes from the backup database using the *Restore Data* option of the PAS control panel (Figure 6.2). Furthermore, users also can create a secondary backup of the

database to allow them to copy the database onto some other places (e.g. PC or Laptop) using the *Backup Data* option of the PAS control panel (Figure 6.2). Moreover, storing data onto smartcards provides another layer of protection against the intentional or accidental corruption of data. Once the data is stored onto a smartcard and even if the mobile device (along with the database) gets stolen, all data can be easily recovered by installing a new PPIdP and the PAS in the device and then synchronising the device using the NFC smartcard as discussed previously.

- **Name Qualification and Impersonation Issues.** The most difficult problem in terms of security for implementing a PPIdP is how to name-qualify different PPIdPs installed on different users' phones. In the general setting of SAML and OpenID, the entity ID of an IdP and OpenID are name-qualified based on the Domain Naming System (DNS). This ensures that no two IdPs have the same entity ID and no two users have the same OpenID in OpenID. Since the PPIdP installed on different phones will have the same entity ID (https://localhost:8443/saml/idp) and the same OpenID IdP name (https://localhost:8443/OpenIDTest) and different users in different phones might have the same username, malicious users might use this technique to impersonate users from other phones. The problem has been solved by devising a request/response mechanism to name-qualify each IdP installed on different phones from the viewpoint of each SP. The mechanism is described below for a PIF Type 1 federation.

  1. When a user provides the entity ID (https://localhost:8443/saml/idp) of a PPIdP and the respective code while creating a dynamic federation, the SP generates a random hash and a password. The random hash is used as an SP-specific identifier for the PPIdP and is included in the provided entity ID so that it takes the form of: *https://localhost:8443/saml/idp/random_hash*. The modified entity ID and the password are also exchanged during the metadata exchange period. The metadata of the PPIdP is based on the modified entity ID which will be used for any subsequent SAML protocol flow. Once the metadata is successfully exchanged and a dynamic federation is created, both the PPIdP and the SP store four pieces of information: the modified entity ID, the password, the entity ID of the SP and the code used for the dynamic federation.
  2. When the user wishes to access one of the restricted resources, just before the WAYF Page is shown to the user, the SP sends a message to the PPIdP providing the entity ID of the SP and the code, and ask for the PPIdP's entity ID and corresponding password.
  3. The PPIdP retrieves the saved entity ID along with the password using the SP's entity ID and the provided code and send them back to the SP.
  4. The SP matches the supplied entity ID and password with its database entry.

Then, the supplied entity ID is used to filter out other PPIdP entity IDs (e.g. the entity ID that belongs to the https://localhost:8443/ domain) when the TAL is presented. Note that other entity IDs belonging to other trusted or untrusted IdPs are not filtered out.

5. When the user chooses one of the IdPs, the usual SAML protocol flow resumes.

This mutual authentication between an IdP and SP serves two purposes:

- it helps the SP to name-qualify each PPIdP using the SP-specific random hash, and
- a malicious user of one PPIdP will not be able to impersonate other users of other PPIdPs (in different phones) using the entity ID.

An SP has very little to gain by acting maliciously since a malicious act by an SP will expose its services to unauthorised users. However, a malicious PPIdP may gain considerably if it can impersonate another PPIdP. Using the stated mechanism, a malicious PPIdP would be required to know both the modified entity ID and the respective password. Since all communications take place on encrypted channels using HTTPS, the only option for a malicious PPIdP is to guess both the modified entity ID and password. As both the hash of the entity ID and password consist of 12 randomly generated alphanumeric characters, they will be difficult to guess. A similar technique can be used to create a PIF Type 2 federation using a PPIdP.

The whole mechanism is invisible to users which alleviates them from any associated complexities. Interested users, however, have the option to view the modified entity ID and OpenID provided by each SP using the *View EntityID/OpenID* option of the PAS control panel (Figure 6.2). In cases where a user is accessing the federated SP from another devices where the PPIdP is not installed, the intermediate request/response mechanism will time out and the user cannot view any PPIdP in the WAYF page. The presented TAL will contain only other trusted or untrusted IdPs. The proposed mechanism has been successfully tested with PPIdPs installed in three different Android phones.

In SAML, an IdP has the option to generate an unique identifier (known as the *Transient Identifier*) for each SP. This transient identifier is passed on to the SP as the unique identifier for a user. The PPIdP uses the same approach. However, to ensure that different users from different PPIdPs do not generate the same transient identifier, the generated entity ID (in the format of https://localhost:8443/saml/idp/random_hash) for the requested SP as well as the entity ID for the SP are used to generate the transient identifier. This guards against the possibility of a user maliciously impersonating another user.

6.5.3 Privacy Requirements

The core privacy requirements for any Identity Management System is to ensure that it supports pseudonymity (or anonymity) and empowers users by giving them full control over their own data as discussed in Chapter 4. SAML supports pseudonymity with the concept of transient identifiers. A PPIdP adopts the same technique that generates different transient identifiers for different SPs as discussed above. This enables each SP to maintain a session with a specific user and prevents the possibility of linking the same user across different SPs.

How the Selective Disclosure and the Explicit Consent requirements are satisfied depend on the particular IdP, developing a system based on SAML does not automatically satisfy these requirements. Therefore, it is analysed how these requirements are satisfied by a PPIdP. Firstly, the PAS allows one to set a general ARP that will dictate which attributes to release to any SP. Users can also set a policy for each SP. Secondly, a user is presented with a form, called the *Consent* form, containing the attributes (excluding the attributes that have been filtered out based on the ARP) just before they are released. This allows the user to select the attributes that she wants to release to that SP. Moreover, the PAS also provides an interface that the user can use to examine the history of what attributes have been released to which entities by using the *Released Attributes* option of the PAS control panel (Figure 6.2). All these features enable the PPIdP to meet the Selective Disclosure and Explicit Consent requirements of an IMS. Data minimisation can be achieved by releasing only the required attributes to an SP. For this, users must be informed which attributes are required for accessing a service. SAML does not have any mechanism to inform users regarding required attributes. In next two chapters, it will be explored how data minimisation with a PPIdP can be achieved.

6.5.4 Trust Requirements

Since a PPIdP is federated with an SP using the mechanism of dynamic federations, the PPIdP, once federated with a SAML SP, must be treated as untrusted by the SP and the SP will be treated as semitrusted by the PPIdP. Therefore, all attributes released by the PPIdP have a LoA value of 1 at the SP.

6.6 Discussion

The PPIdP offers a number of advantages that can overcome many of the limitations of the current IMSs. Here, these advantages are outlined and analysed:

- As stated before, this approach empowers users to have full control over an IdP and the attributes stored. Users have ultimate control over their attributes and they dictate which attributes are released to which SP. This essentially solves problems related to Data Ownership and Controllability.

- A PAS acts as the central repository for storing a user's attributes. Since the attributes are not scattered over many IdPs, their management becomes easier. Furthermore, a PPIdP can replace all low trusted and untrusted IdPs as the LoA values provided by those IdPs will be the same as that provided by the PPIdP. By replacing all low trusted and untrusted IdPs with a PPIdP, the problem that arises from the effect of multiple partial identities can be effectively counteracted. This will allow the identity space to look like Figure 6.15. By comparing this figure with Figure 3.3, this reduced identity space with the PPIdP is easier to maintain and offers complete control of attributes that none of the previous low trusted and untrusted IdPs would offer.

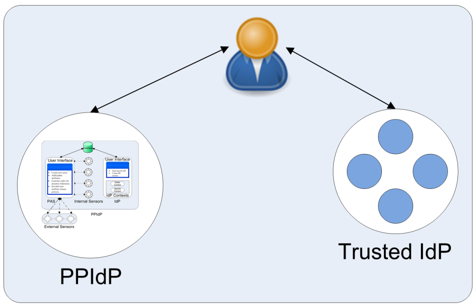


Figure 6.15: Identity space with the PPIdP.

- Users can use as many mobile devices as they want and keep attributes in all of them synchronised using a NFC smartcard. This alleviates the need for a user to manually update attributes across multiple devices and also provides the protection against any corruption of data or database.

- Usually attributes stored in traditional IdPs are static in nature. Mobile devices can be used to generate dynamic attributes such as spatial values that can be used to provide context-aware services in a secure way using SAML.

- By accommodating other IdM protocols, the PPIdP can provide a single interface to interact with all types of SPs.

Unfortunately, this approach also has some limitations. It currently supports only the SAML Web Browser SSO Profile and HTTP Post binding and is incompatible with other SAML profiles and bindings. Currently, the implemented PPIdP does not support other advanced OpenID extensions such as Provider Authentication Policy Extension (PAPE) [123] and

OpenID Attribute Exchange [124]. These limitations can easily be tackled by adding these required functionalities.

Another problem of the current implementation is the limited memory size of the NFC cards. With only 512 bytes of memory size, the card cannot store a large number of attributes. One way to tackle this problem is to zip the concatenated attributes and timestamp before they are encrypted. Even so, the number of attributes that can be stored will be limited. Another option is to use a dual interface smartcard. A dual interface card is based on the ISO/IEC 14443 and ISO/IEC 7816 standards [117]. The ISO/IEC 7816 is the standard for contact smartcards, e.g. debit/credit cards, that require to be in contact with a reader for communication. The standard defines technologies and mechanisms to establish communication between the reader and the smartcard. Since a dual interface card has the ability to communicate wirelessly using NFC and the memory capacity for most of them is in the range of 144K byte, they can be a better choice for the scenario. Unfortunately, there is a cost difference between contactless and dual interface cards. A contactless smart card costs less than a pound whereas a dual interface card costs approximately £10 depending on the memory size (the price quotation has been retrieved from `http://www.smartcardfocus.com/`). However, it should be noted that, as the technology grows, the memory size of the contactless card will grow and at certain point of time, they might offer enough memory to offer a viable option for storing attributes for the scenario.

6.7 Conclusion

The User-controlled Identity Management is all about empowering users and establishing a firm control over their attributes. A PPIdP takes the central stage in realising this concept. Since it is under the full control of its user, the user has the ultimate authority to dictate which attributes will be stored in it and which attributes will be released to different SPs. Moreover, a PPIdP can replace different low trusted IdPs, allowing a user to minimise the number of partial identities that need to be managed. Hence, some major issues in the current setting of Identity Management can be tackled effectively using a PPIdP as evident from the discussion of this chapter.

In addition, a PPIdP provides a solid foundation to explore new ideas that can improve the ways online services can be provided by an SP and accessed by a user. Especially, it will be explored how a PPIdP can be leveraged to provide context-aware federate services and how it can help to realise privacy-preserving attribute aggregation in federated services. The next two chapters will investigate these two issues.

Chapter 7

Context-aware Federated Services using PPIdP

7.1 Introduction

In the previous chapters, the User-controlled Identity Management System has been proposed. It also has been discussed how the system is realised using a novel type of IdP called Portable Personal Identity Provider (PPIdP). How the proposed system can be used to overcome many limitations of the existing Identity Management Systems has been analysed. In this chapter, it will be explored how PPIdPs can be used to offer context-aware federated services.

In the last decade or so, an incredible number of mobile devices equipped with an array of sensors have been proliferated. These devices have been the driving force behind the ever-increasing popularities of context-aware services, mainly in the form of location-based services. A study by MarketsandMarkets predicted US$120 billion worth of market share for context-aware computing by 2018 [125]. There are numerous works that have explored how context can be used to provide online services. However, there are a few major problems that are yet to be addressed.

- Firstly, most of these works focus on the authentication or the authorisation of users neglecting other aspects of identity management. In that sense, the issue of Context-aware Identity Management has not been explored in detail, as discussed in Chapter 2.

- Secondly, the existing works are mostly based on the SILO Model where the users need to authenticate themselves, if needed, separately to access each service which could be inconvenient for the user. The usability of such services could be improved by introducing the capability of federated services allowing users to access services with just a single authentication using the Singe Sign On (SSO) feature [81].

- Thirdly, the security and privacy issues in the existing context-aware frameworks have not been addressed properly. As such, unresolved security and privacy issues need to be addressed to harness the full potential of context-aware services [126].

In this chapter, the proposal of a novel framework called Context-aware Federated Service (CAFS) is introduced which can tackle the stated problems. In this regard, a concrete definition of Context-aware Identity Management is provided. Next, the definition has been used to develop a framework for Context-aware Federated Services. The framework has been developed by utilising the PPIdP of Chapter 6. To show the applicability of the approach, a number of use-cases has been illustrated. The framework imposes several functional, security, privacy and trust requirements. Therefore, how the framework meets these requirements are investigated as well as the advantages of the framework over existing works are discussed.

7.2 Context-aware Identity Management

In the literature of Context-aware Services, the meaning of the term *Context* is highly debated and has been defined in numerous ways. Schilit and Theimer [98] used the term *context-aware* for the first time describing contexts as locations, identities of nearby people, objects and changes to those objects. Similarly, Ryan et al. [127] regarded context as the user's location, environment, identity and time. Hull et al. [128] represent contexts as different aspects of the current situation. One of the most accurate and widely-used definition is given by Abowd et al. [129]: "*any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves*". This definition is preferred since it considers the application itself as a context which is related to Identity Management. Interestingly, the term *Context* has a specific meaning in Identity Management. A context in identity management is the application domain or namespace under which an entity exists, operates and is identified with a specific identifier. The identity of an entity within that application domain is defined as the partial identity of that entity for that domain (see Chapter 3). Combining these, the context for Identity Management can be defined in the following way.

Definition 26 *Context in Identity Management is any information that represents the user's partial identity (using an identifier), the physical location from where the user tries to access any service, the time and date of the service request and the application domain in which* *the partial identity of the user is valid (the IdP) or where the user is requesting access to a service (the SP).*

Such information can also be regarded as the *Contextual Information* or *Dynamic Attributes*. The terms *Context, Contextual Information* and *Dynamic Attributes* will be used interchangeably throughout this chapter.

Using the above definition, the term *Context-aware Identity Management* is defined in the following way:

Definition 27 *Context-aware Identity Management is a specific type of Identity Management that can use contexts for at least one of the following:*

1. *registering and de-registering the partial identity of a user containing identifiers, partial identifiers, credentials, etc. to an IdP;*
2. *to aid users in selecting appropriate attributes to release to a specific SP;*
3. *to aid users in selecting appropriate contexts to be used as dynamic attributes;*
4. *for utilising dynamic attributes during the authorisation and service provisioning step; and*
5. *for supporting business and security applications in providing innovative services.*

7.3 Requirement Analysis

The goal is to design and develop a framework for context-aware federated services. At first, a list of requirements is formulated that the framework should satisfy. Based on the taxonomy of the requirements in Chapter 4, the following set of requirements are identified.

Functional Requirements.

F1 **Online Services.** The framework should be integrable with online services.

F2 **Single Sign On Capability.** The framework should provide the SSO capability which allows users to access services from different SPs of the same federation.

F3 **Indoor/Outdoor Location Tracking.** The framework should have the capability to locate users both indoors and outdoors.

F4 **Showing Required Attributes.** The framework should inform users what attributes are required to access a particular service so that they can make informed choices.

F5 **Advanced Authorisation Capabilities.** The framework should provide advanced policy-based authorisation capabilities which will allow administrators of SPs to write advanced yet flexible authorisation policies to enable complex use-cases.

Security Requirements.

S1 **Confidentiality, Integrity and Authenticity.** Contextual data and other user attributes transmitted between different entities should maintain their confidentiality and integrity. The authenticity of contextual data should be verifiable when required.

S2 **Secure User Authentication and Authorisation.** The framework should ensure that services can be offered only to authenticated and authorised entities.

S3 **Preventing Relay Attacks.** The framework should be able to prevent any relay attack involving contextual data.

S4 **LoA Calculation.** The LoA value for a specific IdP is properly determined.

Privacy Requirements. There is no additional privacy requirements other than the ones outlined in the taxonomy of Chapter 4. Hence, it must be ensured that the existing requirements - Support of Pseudonym (denoted as *P1*), Selective Disclosure (denoted as *P2*), Explicit Consent (denoted as *P3*) and Data Minimisation (denoted as *P4*) - are satisfied.

Trust Requirements. The federated services are bound by implicit trust requirements. Therefore, the trust requirements that may arise while using the proposed system must be closely examined. As these trust issues are dependent upon the way a framework is designed and implemented, they will be discussed after the implementation is introduced.

7.4 CAFS Framework

Designing a context-aware framework that allows federated services should be based on the concept of context-aware identity management and satisfy all requirements given in Section 7.3. At first, an IdP with the capability to provide fine-grained contextual information is required. This is because the traditional IdP cannot provide such information.

A novel framework to utilise contexts for federated services, called ***CAFS*** or Context-aware Federated Services, has been designed and developed. The basic idea is very similar to the existing federated architecture where there are three classes of entities: users, IdPs and SPs. SPs are considered to have the capabilities of providing context-aware services where contextual information is received from IdPs. However, the main difference of CAFS and the existing framework is the inclusion of PPIdPs, as introduced in Chapter 6. PPIdPs can be integrated with SPs to create Personal Identity Federations (PIFs) in a dynamic fashion, as described in Chapter 6.

The authorisation of users in CAFS is handled by the eXtensible Access Control Markup Language (XACML). XACML is an OASIS standard that defines a general-purpose access control and authorisation system [50]. It consists of a policy language based on XML and a processing system that knows how to interpret the policy with respect to the relevant application. The policy language is used to create policies where each policy enlists the requirements to access a resource in a protected environment. The major components of XACML are (Figure 7.1):

- Policy Administration Point (PAP) which is responsible for creating and managing all policies,
- Policy Enforcement Point (PEP) which is responsible for intercepting users' requests and enforcing XACML decisions received from the Policy Decision Point (PDP, see below),
- Policy Decision Point (PDP) which is responsible for evaluating users' requests based on existing policies and returning XACML decisions to the PEP, and
- Policy Information Point (PIP) which gathers additional attributes of a user.

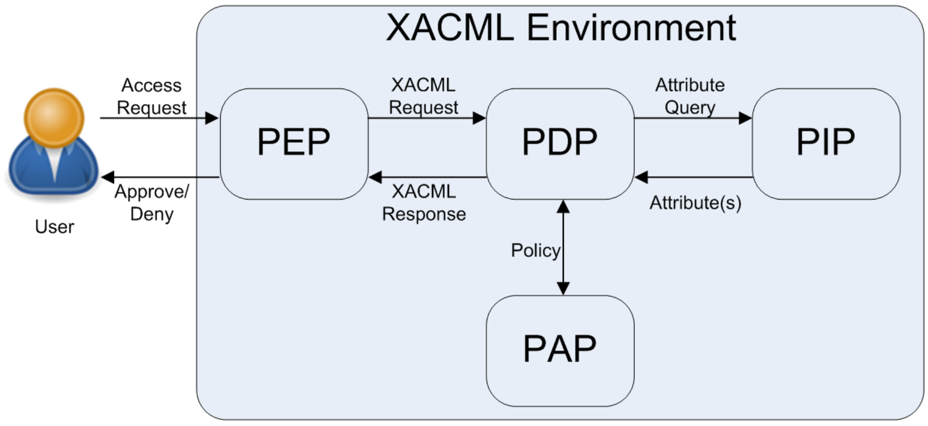


Figure 7.1: A simpler architecture of XACML.

XACML is an example of a request/response language where a user submits a request to the PEP to access a protected resource. The PEP generates an XACML request based on the user attributes, the resource the user wants to access and the action (read/write) the user wants to perform on that resource; it then forwards that request to the PDP. At this point, additional user attributes may be retrieved from the PIP to aid the PDP in making a decision. Using these attributes, the PDP evaluates the existing policies to determine if the user can perform that particular action on that resource. Then, the PDP creates an XACML response and returns it to the PEP. Based on the response, the PEP approves or rejects the user's request.

The existing XACML standard cannot handle location data. Geospatial XACML or GeoXACML is an Open Geospatial Consortium (OGC) standard that has been introduced as an extension to the XACML Version 2.0 standard to enforce access restrictions based on geographic information [130]. The geographic information is encoded in Geography Markup Language (GML) [131] and can be used inside a policy as a condition. Based on the user's location information, the PDP will evaluate the existing policies considering user attributes and user's location and determine if the user is allowed to access the resource. The GeoXACML architecture has been incorporated with the PIF to complete the architecture for the CAFS Framework. The architecture for CAFS using Type 1 PIF is given in Figure 7.2. The PIF consists of a PPIdP and a number of SPs federated with the PPIdP. The CAFS consists of such SPs where each SP incorporates the GeoXACML architecture. The protocol flow using CAFS is as follows. When a user, following the SAML authentication phase at the PPIdP, returns to the SP with a SAML assertion, she is forwarded to the PEP. Essentially, the PEP acts as the entry point for the GeoXACML architecture. The PEP retrieves attributes from the assertion, creates an XACML request with those attributes and forwards the request to the PDP. There is no PIP in this architecture since the GeoXACML architecture of CAFS relies only on the user attributes retrieved from the PPIdP (or the Proxy IdP for Type 2 PIF, see below). The PDP evaluates the user's request using the existing policies and returns an XACML response to the PEP. Based on the XACML response, the user is granted or rejected by the PEP. This architecture can be easily extended for the Type 2 PIF using a Proxy IdP as shown in Figure 7.3.

7.5 Implementations

The existing open source XACML APIs for Java such as SunXACML [132], HERASAF XACML Core [133] and enterprise-java-xacml [134] do not provide any support for the GeoXACML. There does however exist a commercial proprietary implementation of the GeoXACML in [135]. An incomplete implementation of GeoXACML has been located in the GeoServer SVN Repository [136]. That project has been exported and integrated it with the SunXACML API by going through some major modifications to a number of its classes. The combined API has been bundled with the Jetty Web Server [116] that runs on a local web server.

Next, a number of use-cases is illustrated to demonstrate the applicability of the proposed approach.

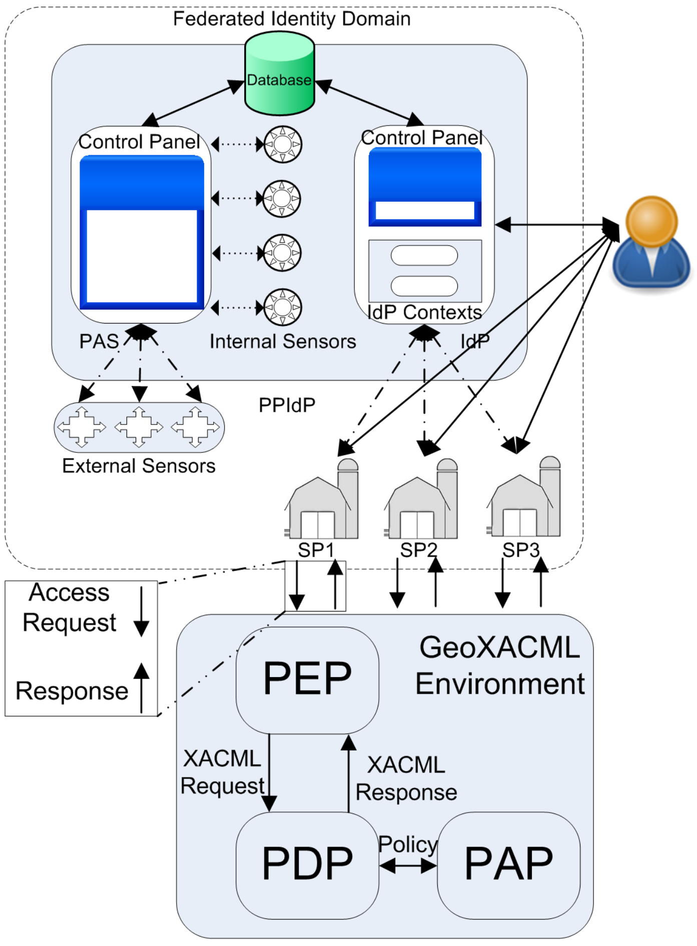


Figure 7.2: CAFS Type 1 Architecture.

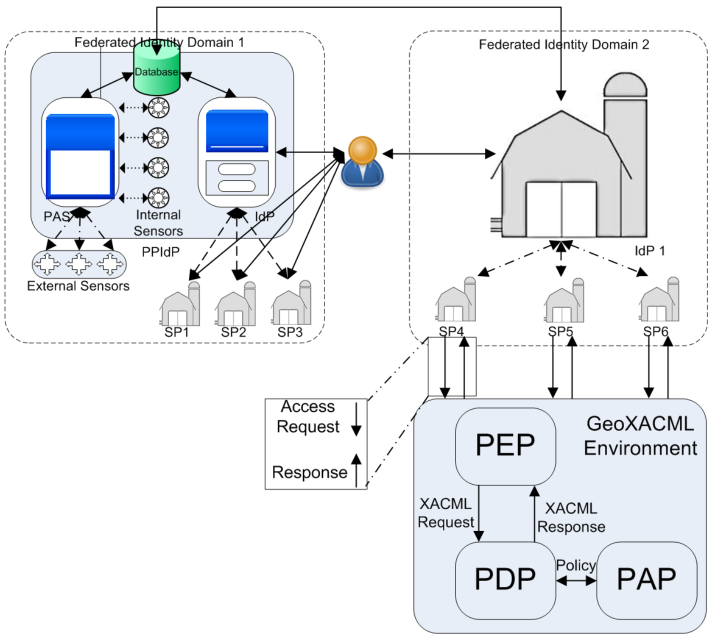


Figure 7.3: CAFS Type 2 Architecture.

7.5.1 Configurations

The SimpleSAMLphp library has been used for deploying a SAML SP. The PEP has been developed in PHP. There are two service providers (hereafter denoted as *SP1* and *SP2*) in the deployment. Each SP has a few web pages that have been protected using the PEP. Each page has different requirements that need to be fulfilled to access that page. These requirements have been used as conditions inside the policies in the GeoXACML environment. Table 7.1 specifies the list of attributes required for each page in each SP. In the table, the value *val* refers to the rectangular bounding box of these geographic values: (55.865318, -4.267437), (55.865185, -4.2660857), (55.864722, -4.266203), (55.864806, -4.268220). The *gte* condition signifies that the user attribute should be greater than or equal to the stated value, *equals* signifies that the user attribute should match the stated value, *within* signifies that the location of the user should be within the specified value and the *between* signifies that the request should be made during the specified duration. The value *ST or HT* signifies that the IdP has to be semitrusted or highly trusted respectively. The value *a* for the *org* attribute signifies a specific organisation.
The location attribute will be provided by the PPIdP and can locate a user outdoor within a certain geographical location, as in the case to access the *Page1.php* of SP1. However, some services or resources (as in the cases to access the *Page2.php and Page3.php* of SP1) require the system to locate a user inside a building (e.g. in a room of a particular floor). Verbally, the requirements for accessing the *Page2.php* can be expressed as: "*a user can access Page2.php only if she is within a geographic location and has a loa value greater than or equal to 1 and has submitted the request from Building 1, Floor 3, Room 5 in between 14:00 to 14:30 on 4 May 2013*". There have been a few works to locate a user inside a building, however, the technology is not precise yet. That is why an alternative approach using Dynamic QR codes introduced in [40, 56] has been adopted. Their work has been used as a basis for the proposed deployment with one major difference: they did not consider the usage of federated services meaning if the user needs to access more than one service, they will need to authenticate again whereas the deployment alleviates the user from any further authentication by leveraging the SSO capability of the federation.

A QR (Quick Response) code is a type of two dimensional bar code. It can display information in a pictorially encoded format which can be read by a device having a digital camera and a specific piece of software to decode the information [137]. It has gained considerable popularity with the rise of smart mobile devices as most of these are capable to decode QR codes. Traditional QR codes are static in nature meaning that once one is generated its contents cannot be altered. In fact, it is not normally required that the contents of the QR code change after they are generated since they are used for marketing purposes via printed or digital media. With a computer system running in the background, it is possible to easily

Table 7.1: Attributes required for each page

Name	Resource	Attributes	Conditions	Values
SP1	Page1.php	loa	gte	1
		location	within	*val*
	Page2.php	loa	gte	1
		location	within	*val*
		building	equals	1
		floor	equals	3
		room	equals	5
		time	between	14:00 - 14:30
		date	equals	2013-05-04
	Page3.php	idp	equals	HT/ST
		location	within	*val*
		building	equals	4
		floor	equals	5
		room	equals	2
		time	between	14:00 - 15:30
		date	equals	2013-05-04
		loa	gte	1
		idp	equals	HT
		pos	equals	student
		org	equals	*a*
		loa	equals	2
SP2	Page1.php	loa	gte	2
		salarygrade	gte	6
		age	gte	33
	Page2.php	loa	gte	2
		salarygrade	gte	6
		age	gte	33
		position	equals	admin

generate and display dynamic QR codes which can be refreshed dynamically. This approach has been adopted in the implementation where different information such as the Building Number, Floor Number, Room Number, Date and Time are encoded into QR codes. In this implementation, this scenario has been emulated by developing an Android app. The assumed administrator provides the required information (Building no., Floor no., etc.). The information is combined and then encrypted. The secret key is shared between the app and the PEP. The user scans a QR code using the PAS (Figure 7.4) upon entering the specific room from where she has to access the specified resource. The encrypted information retrieved from the QR code is saved as an attribute on the PAS that the user must release to access that resource. This means that users must know beforehand which attributes they need to release to different SPs; otherwise they might not release the required attributes to access a particular resource. This is also one of the requirements of CAFS. That is why when a user

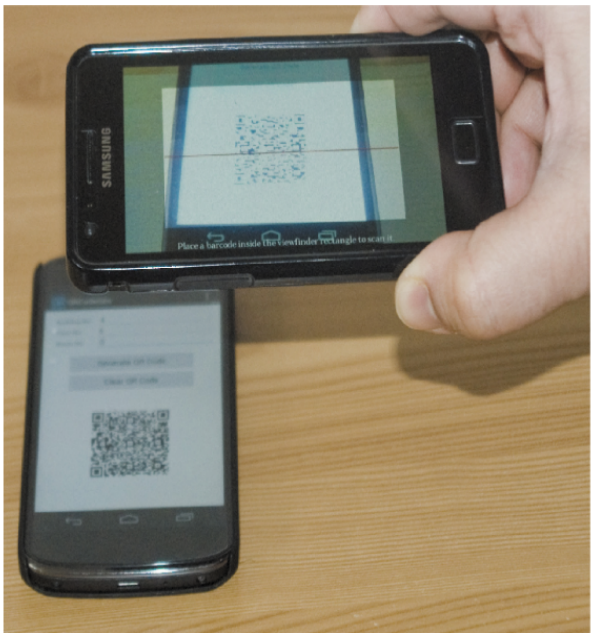
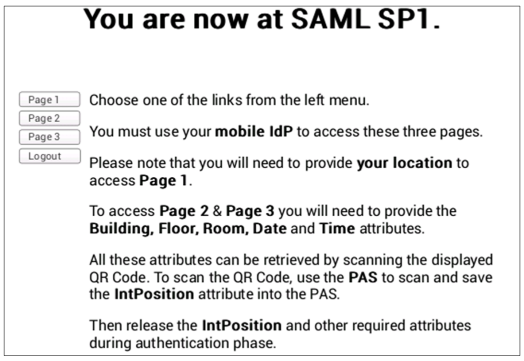


Figure 7.4: Scanning a QR code.

visits the homepage of a service provider, she is informed which IdP they will need to use and which attributes they will need to release. Figure 7.5(a) and 7.5(b) show homepages of the service providers.



You are now at SAML SP1.

Choose one of the links from the left menu.

You must use your **mobile IdP** to access these three pages.

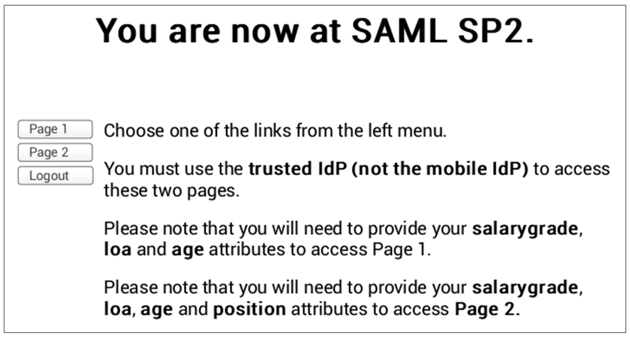
Please note that you will need to provide **your location** to access **Page 1**.

To access **Page 2** & **Page 3** you will need to provide the **Building, Floor, Room, Date** and **Time** attributes.

All these attributes can be retrieved by scanning the displayed QR Code. To scan the QR Code, use the **PAS** to scan and save the **IntPosition** attribute into the PAS.

Then release the **IntPosition** and other required attributes during authentication phase.

(a) SP 1.



You are now at SAML SP2.

Choose one of the links from the left menu.

You must use the **trusted IdP (not the mobile IdP)** to access these two pages.

Please note that you will need to provide your **salarygrade**, **loa** and **age** attributes to access Page 1.

Please note that you will need to provide your **salarygrade**, **loa**, **age** and **position** attributes to access **Page 2**.

(b) SP 2.

Figure 7.5: Homepages of Service Providers.

For accessing *Page3.php*, the user, in addition to attributes similar to *Page2.php*, will need to release the *pos* and *org* attributes from a highly trusted IdP to testify that she belongs to an organisation. This means that the user will need to combine attributes from two SAML IdPs: the location, building, floor, room, time and date attributes from the PPIdP and the *pos* and *org* attributes from a highly trusted IdP. Here, it has been assumed that the *pos* and *org* attributes are stored at the highly-trusted IdP and thus will have a LoA value of 2. This interaction can be accomplished using the use-case 2 illustrated in Chapter 6 where the highly-trusted IdP acts as the Proxy IdP. It needs to be federated with the SP in the traditional way so that it is considered as highly trusted by the SP. The PPIdP needs to be federated with the Proxy IdP as an authentication source as discussed in Chapter 6.

With these settings, a number of protocol flows is discussed. It is assumed that users have already federated the PPIdP with the respective SP and with the trusted (proxy) IdP to create a Type 1 and Type 2 PIF respectively.

7.5.2 Use-Case: SP1

The requirements for accessing different pages have been given in Section 7.5.1. Now different protocol flows are outlined in turn.

Accessing Page1.php

The protocol flow for accessing *Page1.php* is illustrated in Figure 7.6 and is discussed below:

1. The user visits the homepage of SP1 using a browser on her mobile device where a PPIdP is installed. She notes that she will need to release the location attribute to access *Page1.php*.
2. The user clicks the *Page 1* button in order to access *Page1.php*.
3. She is redirected to the WAYF Page of SP1 and she selects the PPIdP.
4. A SAML Authentication Request is sent to the PPIdP.
5. The user is authenticated there with her username and password.
6. At this point, the user is provided with the consent form containing the list of attributes and she selects the location attributes and clicks the *Submit* button.
7. A SAML response with a SAML assertion is sent back to the SP where the PEP receives the response.
8. The PEP validates the assertion and retrieves the attributes from the SAML response. Since the response is from an untrusted IdP, the PEP assigns a LoA value of 1 to this response.
9. An XACML request with the *location* and *loa* attributes are sent to the PDP.
10. The PDP uses existing policies to evaluate the user's request. Depending on the location of the user, an XACML response with the *Permit/Deny* value is sent back to the PEP.

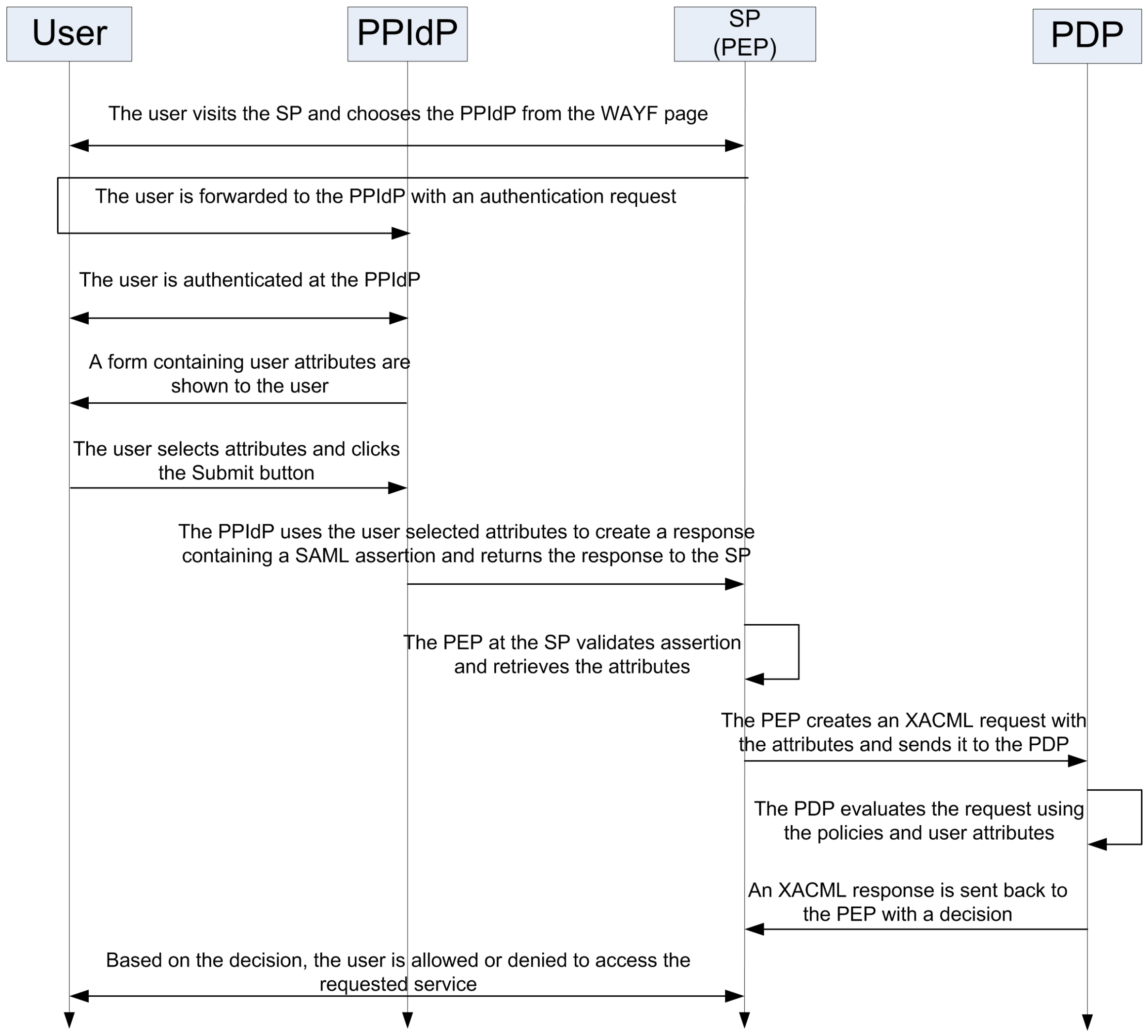

User
PPIdP
SP
(PEP)
PDP
The user visits the SP and chooses the PPIdP from the WAYF page
The user is forwarded to the PPIdP with an authentication request
The user is authenticated at the PPIdP
A form containing user attributes are shown to the user
The user selects attributes and clicks the Submit button
The PPIdP uses the user selected attributes to create a response containing a SAML assertion and returns the response to the SP
The PEP at the SP validates assertion and retrieves the attributes
The PEP creates an XACML request with the attributes and sends it to the PDP
The PDP evaluates the request using the policies and user attributes
An XACML response is sent back to the PEP with a decision
Based on the decision, the user is allowed or denied to access the requested service

Figure 7.6: Protocol flow for accessing the Page1.php.

11. The user is granted or denied accordingly to access the *Page1.php*.

Accessing Page2.php and Page3.php

At this point, if the user tries to access *Page2.php*, there is no need for the user to authenticate again due to the Single Sign On (SSO) capability offered by the PPIdP or the Proxy IdP. However, the user will not be authorised to access the service since the user does not have the required additional attributes. To access either *Page2.php or Page3.php*, the user must logout. Then, she needs to be in that specific room of the floor in the building from where she can access the service. Assuming that the user is in that specific room (e.g. Building 1, Floor 3 and Room 5) and there is a Dynamic QR code displayed inside the room, the user uses the PAS to scan the QR code and saves the information as an attribute, called *IntPosition*, into the PAS. With this setting, the protocol flow is very similar to what is illustrated in Figure 7.6 and is discussed below.

1. The user visits the homepage of SP1 with a browser of her mobile device and notes that she needs to release the *location* and *IntPosition* attributes to access *Page2.php*. The user clicks the *Page2.php* button.

2. The user is forwarded to the WAYF page. She chooses the PPIdP from the WAYF page and is forwarded to the PPIdP with an authentication request.

3. The user is authenticated there with her username and password.

4. The user is presented with the consent form in which she chooses to release the *location* and *IntPosition* attributes (Figure 7.7).

5. A response containing a SAML assertion with the selected attributes are sent back to the SP where the PEP receives the response.

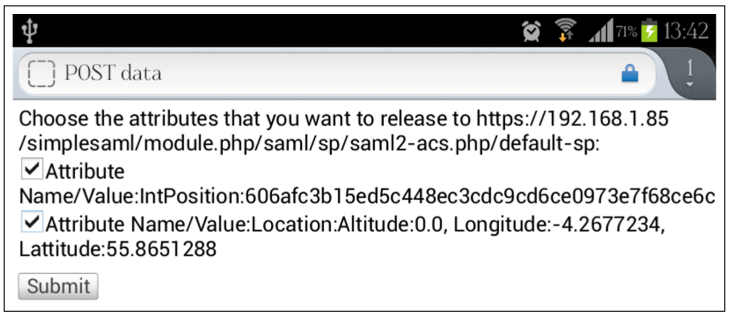
6. Since the assertion contains the *IntPosition* attributes, the PEP decrypts its value and retrieves the *building, floor, room, date and time* attributes.

7. These attributes are used to create an XACML request to send to the PDP.

8. The PDP uses the existing policies to evaluate the user's request using the provided attributes.

9. If the user scans the QR code while being inside the room, an XACML response with the *Permit* value is sent back to the PEP.

10. The user is granted to access the *Page2.php*.



Choose the attributes that you want to release to https://192.168.1.85 /simplesaml/module.php/saml/sp/saml2-acs.php/default-sp:
✓ Attribute Name/Value:IntPosition:606afc3b15ed5c448ec3cdc9cd6ce0973e7f68ce6c
✓ Attribute Name/Value:Location:Altitude:0.0, Longitude:-4.2677234, Lattitude:55.8651288

Figure 7.7: Releasing location attributes at the PPIdP.

To access *Page3.php*, the user needs to release attributes from two IdPs as discussed above, meaning that she needs to combine attributes from two IdPs. The protocol flow is illustrated in Figure 7.8 and is outlined below.

1. The user visits the homepage of SP1 using a browser on her mobile device and notes that she needs to release the *pos* and *org* attributes from the Proxy IdP and the *location*

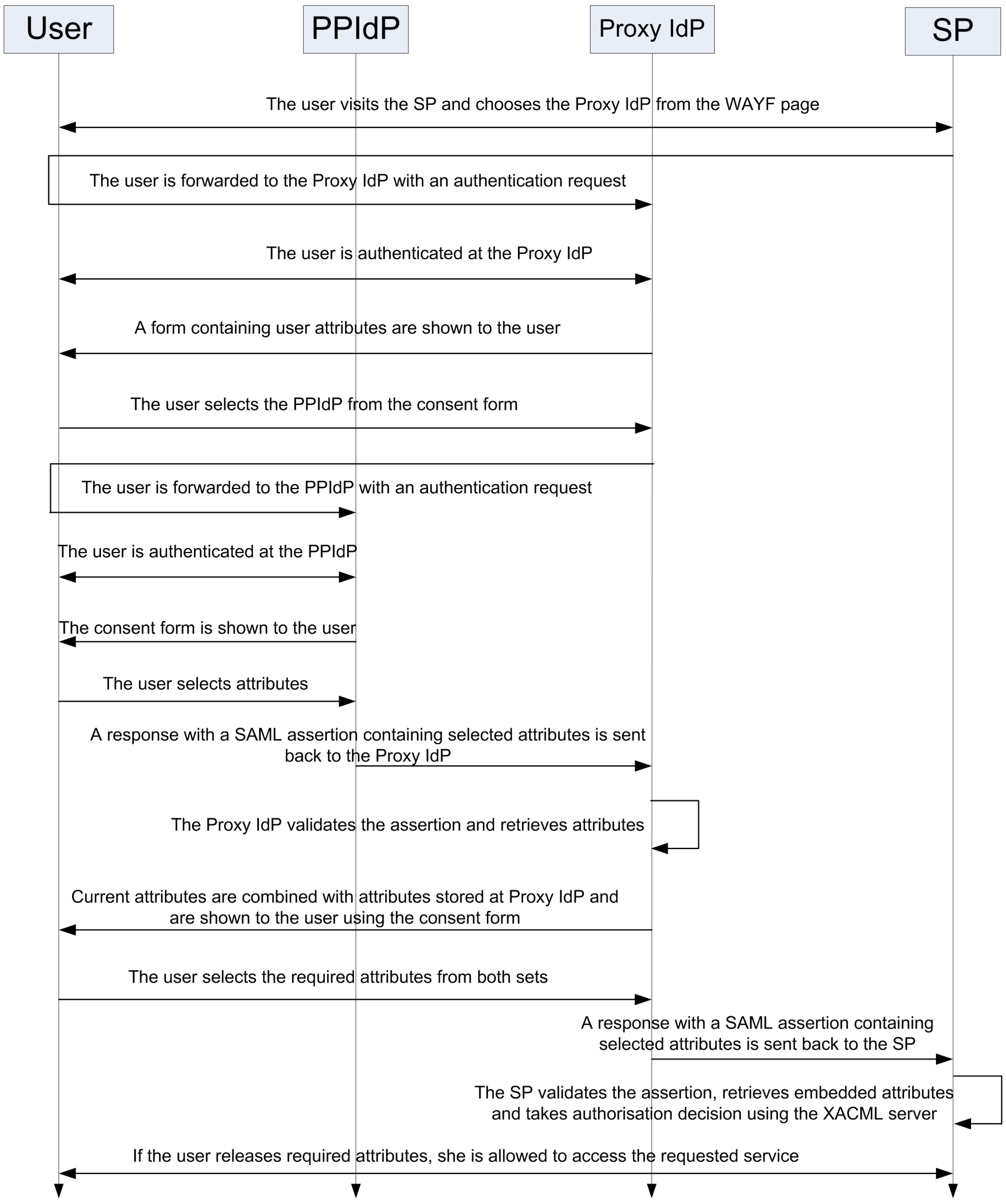

User
PPIdP
Proxy IdP
SP
The user visits the SP and chooses the Proxy IdP from the WAYF page
The user is forwarded to the Proxy IdP with an authentication request
The user is authenticated at the Proxy IdP
A form containing user attributes are shown to the user
The user selects the PPIdP from the consent form
The user is forwarded to the PPIdP with an authentication request
The user is authenticated at the PPIdP
The consent form is shown to the user
The user selects attributes
A response with a SAML assertion containing selected attributes is sent back to the Proxy IdP
The Proxy IdP validates the assertion and retrieves attributes
Current attributes are combined with attributes stored at Proxy IdP and are shown to the user using the consent form
The user selects the required attributes from both sets
A response with a SAML assertion containing selected attributes is sent back to the SP
The SP validates the assertion, retrieves embedded attributes and takes authorisation decision using the XACML server
If the user releases required attributes, she is allowed to access the requested service

Figure 7.8: Protocol flow for accessing the Page3.php.

and *IntPosition* attributes from the PPIdP to access *Page3.php*. The user clicks the *Page3.php* button.

2. The user is forwarded to the WAYF page. She chooses the Proxy IdP from the WAYF page and is forwarded to the Proxy IdP with an authentication request.

3. The user authenticates herself at the Proxy IdP.

4. The user is presented with a consent form that contains the attributes stored at the Proxy IdP.

5. The consent form also contains a button called *Aggregate More Attributes* (Figure 7.9). The user clicks the button.

6. A session is created and the user is forwarded to the IdP selection page where other SAML IdP (including the PPIdP) linked to the Proxy IdP are listed.

7. The user chooses the PPIdP and she is forwarded to the PPIdP with a SAML authentication request.

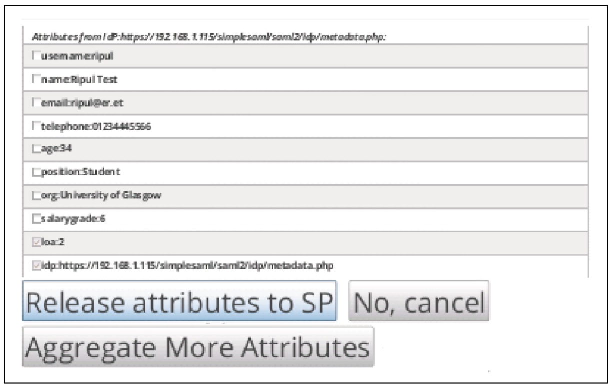
8. The user authenticates herself at the PPIdP.

9. The user is presented with the consent form at the PPIdP (Figure 7.7). The user selects the required attributes and clicks the *Submit* button.

10. A response with a SAML assertion with the attributes are returned to the Proxy IdP. The Proxy IdP validates the assertion and retrieves the attributes.

11. The Proxy IdP retrieves attributes stored at its end and combines both sets and presents them in the consent form (Figure 7.10). Note that the Proxy IdP also inserts the IdP and LoA value for each corresponding set of attributes from a single IdP.

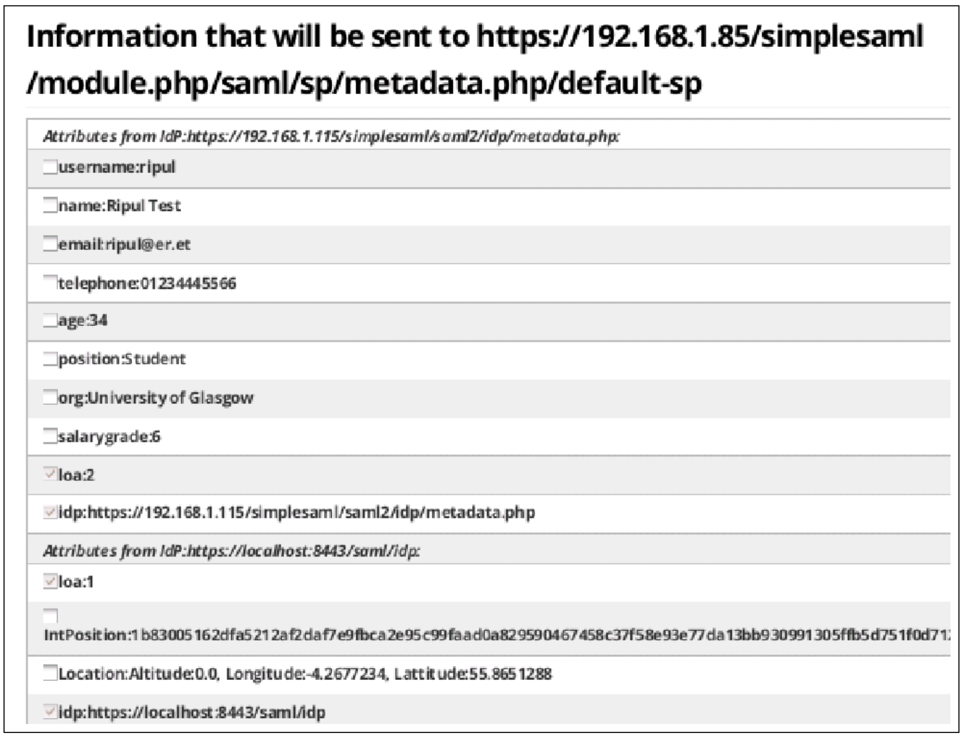
12. Assuming the user chooses the required attributes for accessing the *Page3.php*, a response with a SAML assertion containing the attributes will be sent back to the PEP and the same XACML protocol flow takes places to determine if the user can access the *Page3.php*.


Attributes from IdP:https://192.168.1.115/simplesaml/saml2/idp/metadata.php:

☐ username:ripul
☐ name:Ripul Test
☐ email:ripul@er.et
☐ telephone:01234445566
☐ age:34
☐ position:Student
☐ org:University of Glasgow
☐ salarygrade:6
☑ loa:2
☑ idp:https://192.168.1.115/simplesaml/saml2/idp/metadata.php

Figure 7.9: The consent form at the Proxy IdP allowing the user to combine attributes.

Note that, the user can access (*Page1.php* and *Page3.php*) or (*Page1.php* and *Page2.php*) when she can access *Page3.php* or *Page2.php* respectively, since the entities in both pairs require the same value for the common *location* attribute.


Information that will be sent to https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

Attributes from IdP:https://192.168.1.115/simplesaml/saml2/idp/metadata.php:

☐ username:ripul
☐ name:Ripul Test
☐ email:ripul@er.et
☐ telephone:01234445566
☐ age:34
☐ position:Student
☐ org:University of Glasgow
☐ salarygrade:6
☑ loa:2
☑ idp:https://192.168.1.115/simplesaml/saml2/idp/metadata.php

Attributes from IdP:https://localhost:8443/saml/idp:

☑ loa:1
☐ IntPosition:1b83005162dfa5212af2daf7e9fbca2e95c99faad0a829590467458c37f58e93e77da13bb930991305ffb5d751f0d71
☐ Location:Altitude:0.0, Longitude:-4.2677234, Lattitude:55.8651288
☑ idp:https://localhost:8443/saml/idp

Figure 7.10: Combined attributes from the Proxy IdP and the PPIdP.

7.5.3 Use-Case: SP2

Since the pages of the SP2 requires an LoA value of 2, the user cannot use the PPIdP (either directly or indirectly via the Proxy IdP). The protocol flow is similar to what is illustrated in Figure 7.6 and is described below.

1. The user visits the homepage of SP2 and notes the list of attributes that she needs to release to access *Page1.php* and *Page2.php* (Table 7.1). The user clicks the *Page 1* button.
2. She is redirected to the WAYF Page of SP2 and she selects the Proxy IdP.
3. A SAML Authentication Request is sent to the Proxy IdP.
4. The user is authenticated at the Proxy IdP.
5. The user is provided with the consent form containing a list of attributes.
6. The user selects the required attributes and clicks the *Submit* button.
7. A response with a SAML assertion containing the selected attributes is sent back to the PEP.
8. The SP validates the assertion and retrieves the attributes.

9. These attributes are used to create an XACML request to send to the PDP.
10. The usual XACML flow takes place. If the XACML response contains a *Permit* value, the user is allowed to access the resource otherwise, she is denied access.

7.6 Analysis

This section investigates whether the CAFS framework has satisfied different requirements outlined in Section 7.3.

7.6.1 Functional Requirements

The current implementation of the PPIdP does not use any contexts for registering and deregistering the partial identity of a user. However, a trusted IdP can impose the requirement that the user must visit a particular place physically to register for different services and thus utilising the contexts for registration. The whole framework has been designed and developed with online services in mind and provides a seamless Single Sign On capability across different service providers in the same federation. Therefore, it meets requirements *F1* and *F2*.

As illustrated in previous use-cases, the framework has the ability to locate users indoors as well as outdoors and thus meets requirement *F3*. To meet *F4*, a mechanism will be needed to allow an SP to request attributes from an IdP. For the use-cases, an option has been chosen where the deployed SPs publish on their homepages what attributes are needed to access each service. This is not a very user-friendly option as the user must keep note of what attributes are required for which service. In the next chapter, a more user-friendly way of accomplishing this is presented. As evident from the use-cases, CAFS supports an advanced policy-based authorisation infrastructure XACML and thus also meets the requirement *F5*.

7.6.2 Security Requirements

The whole framework has been developed using standardised SAML and XACML protocols. The communication channel in SAML is fully encrypted. CAFS also uses the encrypted channel (HTTPS) to communicate with the GeoXACML environment. The encrypted channel guarantees the confidentiality and integrity of information, including contextual information, passed between involved entities. The discussion regarding the authenticity of contextual data, e.g. location, deserves further discussion since location information can be falsified in mobile devices using available application. For example, Fake GPS location [138] and Location Spoofer [139] are two popular Android apps for spoofing location data for the Android platform. Therefore, it is advisable to provide location-based services in such a way that users need to be physically present to avail the service thereby undermining the motive of spoofing location data. An alternative approach is to deploy external sensors that can be used to prove that a user is in vicinity of the sensor at a certain time. This approach has been adopted by using dynamic QR codes as external sensors. To ensure that the information retrieved from the QR code cannot be spoofed, the information is encrypted and then the QR code is generated. Here, it is assumed that the dynamic QR codes are deployed by the SP, and thus the SP can decrypt the encrypted QR code to retrieve the location data. This ensures the authenticity of location data and thus meets *S1*. The CAFS also authenticates users securely over the HTTPS channel and provides the services only to the authorised users which is determined by the respective XACML policy and thus meets *S2*. To limit relay attacks, the date and time of the day are also embedded inside the information in the QR code. This ensures that the QR code cannot be used after that date and time, thereby ensuring *S3*. The Proxy IdP determines the LoA value of the PPIdP for the second use-case, thereby satisfying *S4*.

7.6.3 Privacy Requirements

During the authentication phase, the PPIdP and the Proxy IdP generate a pseudonymous identifier (similar to what was discussed in Chapter 6) which is unique to each SP. In this way, users can maintain a session with an SP, however, several SPs cannot collude to build a profile of a user. The Proxy IdP can also provide such a unique identifier. The user can release such unique identifiers if it is required. This satisfies *P1*.

A user can set up a pre-configured ARP to decide which attributes she wants to release to a particular SP. Once a user is authenticated, she can choose, using the consent form, which attributes she wants to release to an SP. All these features help users to select attributes and contextual information, control data flow and to provide her consent during a service access scenario and hence *P2* and *P3* are satisfied. As mentioned in Chapter 6, data minimisation can be achieved if a user knows which attributes are required for accessing a service and then releasing only those attributes. The *F4* requirement enables users to know which attributes are required for a service and this implicitly satisfies *P4*.

7.6.4 Trust Requirements

Discussion of any federated system will be incomplete without analysing the trust issues involved. As the Proxy IdP and the SP have been in the traditional way, they trust each other, whereas the PPIdP is considered as untrusted to the IdP and the SP. If the authentication is delegated by the Proxy IdP to another IdP (as in the case of CAFS Type 2), an SP depends on the judgement of the Proxy IdP following the rule of the transitive trust property. That is, if the Proxy IdP considers other IdPs as trusted then they will be trusted by the SP and if other IdPs are considered as semitrusted (or untrusted), then, they will be considered as semitrusted (or untrusted) as well. As mentioned earlier, the LoA value is used as a trust metric. For the CAFS Type 1 architecture, any attributes released by the PPIdP will be implicitly considered by an SP as attributes having a LoA value of 1. For the CAFS Type 2 architecture, the Proxy IdP holds the responsibility to assign the LoA value of 1 for attributes retrieved from the PPIdP and a LoA value of 2 for attributes retrieved from another trusted IdP.

7.7 Discussion

The CAFS provides several key advantages which are highlighted below.

- CAFS is the first framework to consider context-aware identity management in a comprehensive way focusing all aspects of identity management and satisfying all requirements as outlined in Section 7.3.

- CAFS is the first framework to demonstrate how location data can be used to provide federated services. This has been possible by creating Type 1 and Type 2 PIFs. This brings benefits to both users and SPs. A user can utilise the SSO capability of the IdP, thereby reducing the need for further logins for each separate location-based service hosted on different identity domains. SPs can offload the associated cost of managing and maintaining user identities to IdPs and instead can concentrate on providing services.

- CAFS illustrates how geographical location data can be utilised by integrating the GeoXACML API with the SAML and the existing XACML libraries. In addition, the deployment shows how the user can be located precisely inside a building and how the authenticity of the location can be verified using external sensors and the readily available sensing technologies of smart mobile devices.

- CAFS framework empowers users by informing them which attributes they will need to release. This enables users to create fine-grained ARPs and control the flow of their attributes.

- This is the first work to explore the possibility of combining attributes from two different IdPs to provide context-aware services. The discussed use-cases illustrate how this can be achieved using the CAFS framework to formulate innovative and advanced other use-case scenarios.

Table 7.2: Comparison of CAFS with existing works.

Name	FR					SR				PR			
	F1	F2	F3	F4	F5	S1	S2	S3	S4	P1	P2	P3	P4
Bardram et al. [39]	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Jansen et al. [40]	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Malek et al. [41]	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Hsieh et al. [42]	✗	✗	✓	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗
Hayashi et al. [43]	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Covington et al. [46]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Zhang et al. [47]	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Hulsebosch et al. [48]	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
Jean-Yves et al. [49]	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Covington et al. [51]	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Hu et al. [52]	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗
Nishiki et al. [54]	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Covington et al. [55]	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
Goel et al. [56]	✓	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗
CAFS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 7.2 provides a comparative analysis of the framework with existing works on context-aware services against the set of requirements compiled in Section 7.3. The table provides a side-by-side comparison between this work and other existing works. The "✓" symbol has been used in the table to signify that the particular work has considered the respective requirement. The "✗" symbol has been used to signify that either the system has failed to meet that respective requirement. As evident from the table, the proposed framework satisfies more requirements than the existing works for accessing access context-aware services.

Many services will not require any identifying information, providing the location information would be enough to access such services as in the case of *Page1.php* and *Page2.php*. To offer secure services only to an authorised subset of users (based on the Role Based Access Control (RBAC) or Attribute Based Access Control (ABAC)) that might require any identifying information (e.g. the user's identity or the user's group membership) from a trusted source as well as location information (as in the case of *Page3.php*), the user will need to combine attributes from two IdPs as illustrated in the use case. The *Page3.php* access control in the use-cases has only illustrated the RBAC scenarios (e.g. if the role of a user is student). Even though the *Page3.php* use-case required to know the role of the user (e.g. a student), it can be easily adopted for scenarios requiring the user's identity. Moreover, the LoA values provided by the Proxy IdP is a key mechanism for the SP to ensure that the level of assurance for attributes are properly calculated; in turn this could be used to undermine any privilege escalation.

7.8 Conclusion

As current trends suggest, the popularity of federated services and context-aware services will only increase. An architecture that can provide context-aware federated services will be advantageous for users due to the SSO capability, and thus reduce the number of online accounts they need to manage and use to access different services. Such services can also be advantageous for SPs as well, since it allow them to offload the burden of managing user information to IdPs. In this chapter, a novel framework, based on the concept of Portable Personal Identity Provider and Personal Identity Federation, has been presented for context-aware federated services using the SAML, XACML and GeoXACML standards. It also has been shown how combining attributes from different sources can play a crucial role in providing innovative use-case scenarios. The mechanism of combining attributes from different providers is commonly known as *Attribute Aggregation*. There are different such mechanisms. The discussed approach in this chapter is based on the *Identity Proxying* model [58]. The topic of Attribute Aggregation has potential to stand out as a research topic in its own. In the next chapter, the issue of attribute aggregation is treated thoroughly.

Chapter 8

Attribute Aggregation in Federated Identity Management

8.1 Introduction

It is evident from the discussions in previous chapters that federated services have improved the usability and experience of online services. It also has been shown how the proposed User-controlled Identity Management Systems using PPIdPs can offer context-aware federated services while allowing users to have ultimate control over their attributes. Unfortunately, there exists a serious limitation in the current setting of federated services: users can use only a single IdP in a single service session [61]. This is an unrealistic restriction that assumes that one IdP would be able to provide all the required attributes to an SP where, in reality, users have different attributes stored in different providers. To illustrate the restriction, a real-life scenario in the setting of FIM (Federated Identity Management) can be considered. Imagine a user, named *Alice*, wants to buy an age-restricted product from her favourite online shop. She might need to use her Governmental IdP (e.g. her passport or driving licence from passport or driving authorities) to prove her age, her bank details to pay for the product and her loyalty card information to collect loyalty points during this purchase. This information is stored at different IdPs. Even if it is assumed that the online seller has a federated agreement with all these IdPs, the current setting of federated services does not allow Alice to provide all this information in a single session (e.g. during her purchase). Allowing users to aggregate attributes from different providers will enable them to release these attributes to an SP in a single service session which can be used to provide innovative services.

The concept of attribute aggregation has been introduced to tackle this very problem. A number of novel approaches exist for aggregating attributes targeted for federated services. Even though each model has its own strength, most of them suffer from serious limitations.

For example, many are based on either complex user interactions, unrealistic assumptions or have not been implemented in practice. In addition, all models have been exclusively designed to allow users aggregating attributes only from SAML IdPs. With the proliferation of social networks, many users store different attributes across different non-SAML social IdPs (e.g. Facebook, Google and OpenID Providers) and different non-SAML technologies (such as OpenID and OAuth) are increasingly being used for accessing such IdPs. Allowing users to combine attributes from such providers would enable novel service access scenarios. In this chapter, a Hybrid model of attribute aggregation for federated services is presented. The model is based on two existing models and has been designed to specifically address these limitations. This model does not require complex user interactions and is the first to allow users to aggregate attributes from both SAML and non-SAML IdPs. The model imposes several functional, security, privacy and trust requirements. An in-depth analysis has been provided regarding the design choices to ensure that every requirement is met. The current SAML specification does not have any mechanism that will allow an SP to pass any information regarding requested attributes to an IdP during the authentication process. To rectify this problem, a mechanism is proposed. A proof of concept prototype has been implemented based on the proposal. Then, several use-cases using the prototype have been illustrated to demonstrate the applicability of the model. Finally, the advantages of the model along with its limitations are discussed.

8.2 Attribute Aggregation

Attribute Aggregation is the mechanism of aggregating or combining attributes of a user retrieved from multiple identity providers in a single session. As mentioned earlier, the profile of a user in an IdP contains the subset of attributes of that user retrieved from that IdP. Therefore, this means that attribute aggregation, in reality, is actually an aggregation of profiles of a user from different IdPs and the terms "profile" and "attribute" will be used interchangeably. Formally, the set of aggregated attributes can be defined in the following way:

Definition 28 *For any entity $e \in E$, the set of aggregated attributes of e, denoted by $Att{-}Agg^e$ is given by:*

$$Att{-}Agg^e = \cup\{PROFILE^e_d \,|\, d \in DOMAIN\}\,.$$

Here, $DOMAIN$ represents the set of domains and $PROFILE^e_d$ represents the profile of entity e in domain d.

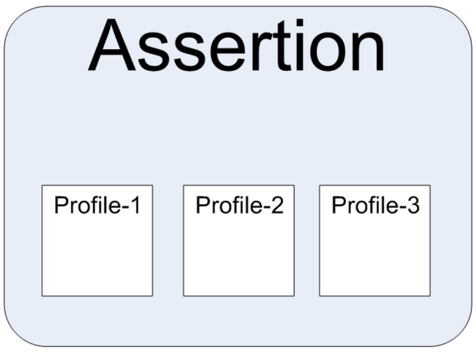


Figure 8.1: A single assertion containing all profiles (attributes).

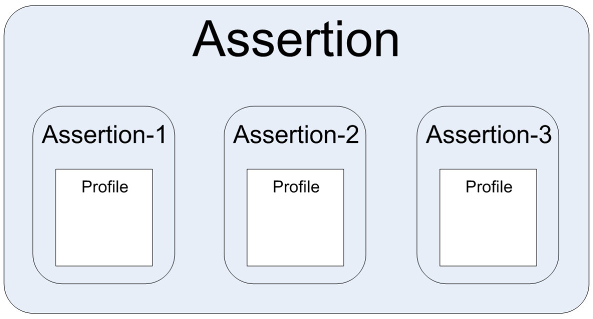


Figure 8.2: A single assertion containing all assertions.

In practice, profiles of a specific user from several IdPs can be aggregated only if the user has authenticated at each IdP and released the profile from each IdP. In SAML, each such profile is embedded inside a separate assertion. Therefore, from the perspective of an implementation, there are two interpretations of attribute aggregation: it either means the aggregation of attributes where all attributes are embedded inside a single assertion (Figure 8.1) or it means the aggregation of assertions where other assertions are embedded inside a single assertion (Figure 8.2). These two possible interpretations will be discussed later in the chapter.

8.2.1 Taxonomy of Attribute Aggregation Models

While aggregating, it is necessary to determine the entity which will aggregate the attributes and the entity that mediates the process. Here, the term *mediation* means initiating the aggregation mechanism. During attribute aggregation a user may need to access several IdPs from where she wants to aggregate her attributes and it is crucial that the user has a starting point and an end point during the process. The two entities, that initiate the process and that combine the released attributes, act as the starting and finishing points for the process. These points can be utilised as the criteria to create a taxonomy of existing attribute aggregation models. Based on where the attribute aggregation takes place, the existing attribute aggregation mechanisms can be classified in three categories: Aggregation at an SP, Aggregation at an IdP and Aggregation at a Client (e.g. a User-Agent/Browser). Each category can then further be classified based on who mediates the aggregation process: an IdP or an SP. Based on these categories, a taxonomy has been created which is illustrated in Figure 8.3. Next, a brief description of each mechanism is provided.

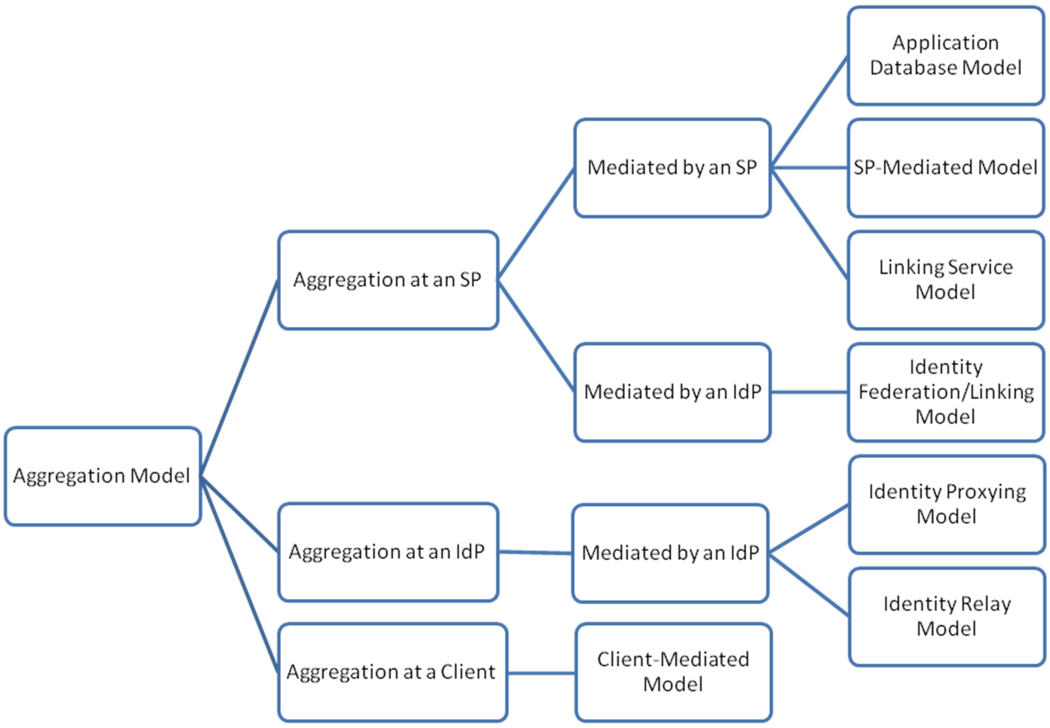


Figure 8.3: Taxonomy of Attribute Aggregation Models.

8.2.2 Aggregation at an SP

The models discussed in this section enable a user to aggregate attributes at an SP. As mentioned earlier, the models can be further classified based on the entity that initiates the process: an SP or an IdP.

Mediated by an SP

The models which allow an SP to initiate the aggregation mechanism are placed into this category.

Application Database (AD) Model. This is the simplest form of attribute aggregation model (Figure 8.4) [58, 59]. In this model, an SP can store additional attributes of a user such as a local identifier, user-preferences for that particular service or group membership, in addition to the attributes supplied by an IdP. The SP creates a mapping of the SP-created identifier to the IdP-supplied identifier to store these additional attributes at a local repository. Such local attributes can be retrieved later on using this mapping to determine if the user is authorised to access its service(s).

SP-Mediated (SPM) Model. In this model, an SP allows a user to aggregate attributes from multiple IdPs in a single session (Figure 8.5) [58, 59]. A user is forwarded to different IdPs, one after another, where she is authenticated separately and finally returns to the SP with the

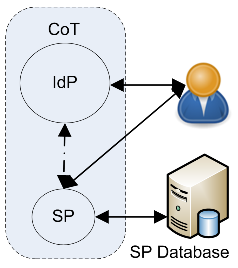


Figure 8.4: Application Database Model.

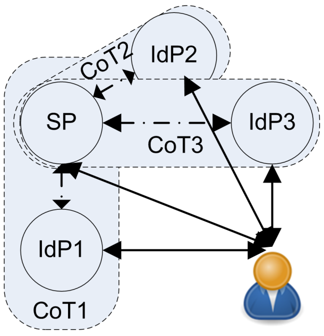


Figure 8.5: SP-Mediated Model.

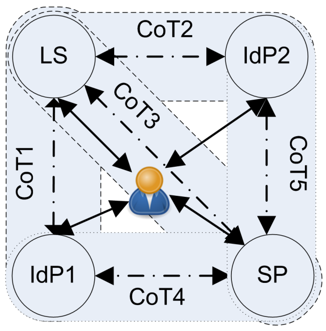


Figure 8.6: Linking Service Model.

IdP-supplied attributes. The SP combines the sets of attributes at its end to determine if the user can access its service(s).

Linking Service (LS) Model. Linking Service model is a combination of the linking and identity relay model (see below). It consists of a special type of SP called the Linking Service (LS, Figure 8.6) [61, 59] which is used by a user using a LS-supplied identifier. This identifier is used to link different IdPs using an IdP-supplied LS-specific persistent identifier in a table called the Linking Table. To access any service of an SP, the user visits the SP and is forwarded at the first IdP (IdP1 in Figure 8.6). The user authenticates at IdP1 and then an assertion containing user-attributes, the persistent identifier for the LS and a reference to the LS is returned to the SP. The SP forwards the persistent identifier to the LS to aggregate attributes. At this point, two options are available: either the LS can retrieve the list of linked IdPs for this persistent identifier using the Linking Table and retrieve attributes from each of them which are then combined at the LS and returned to the SP. Alternatively, the LS can send back the list of linked IdPs to the SP. The SP then retrieves attributes from each IdP. Based on the aggregated attributes, the SP determines if the user can access its service(s).

Mediated by an IdP

This category consists of the model which allows an IdP to initiate the aggregation mechanism.

Identity Federation/Linking (IFL) Model. This model, introduced by the Liberty Alliance framework, is one of the first models to address the problem of attribute aggregation (Figure 8.7) [58, 59]. In this model, an IdP allows a user to create a pair-wise link between two IdPs. To create the link, the user needs to visit and authenticate to the first IdP. The first IdP will ask the user if she wants to federate this IdP with another IdP. If chosen, the user will be asked to federate the second IdP with the first one. At this point, both IdPs will interact with each other to create a random alias. During accessing services from an SP, one IdP will provide that random alias to the SP along with the assertion containing attributes. The SP can use that alias to retrieve another assertion containing attributes from the other IdP. Combining attributes from both IdPs, the SP can determine if the user can access its service(s).

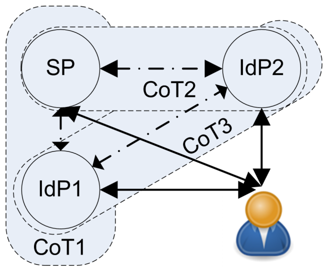


Figure 8.7: Identity Federation/Linking Model.

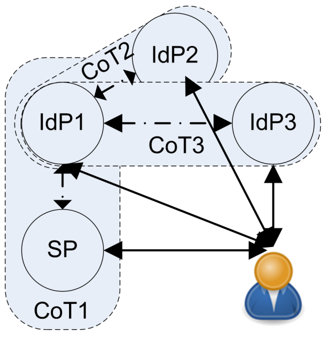


Figure 8.8: Identity Proxying Model.

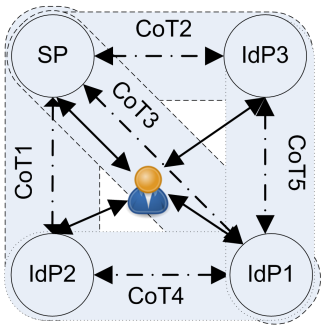


Figure 8.9: Identity Relay Model.

8.2.3 Aggregation at an IdP

This category includes the model where an IdP aggregates attributes.

Mediated by an IdP

The models in which an IdP initiates the aggregation mechanism are placed in this category.

Identity Proxying (IP) Model. In this model, an SP allows a user to aggregate attributes from multiple IdPs using a highly trusted IdP (Figure 8.8) [58, 59]. A user is forwarded to the highly trusted IdP (IdP1 in Figure 8.8) and then the trusted IdP forwards the user to other multiple IdPs. Such an IdP is known as the *Proxy IdP*. After the user is authenticated separately at each IdP, the user returns back to the trusted IdP with an assertion including attributes. At this point, the trusted IdP validates each assertion, retrieves attributes from them and combines the attributes. The trusted IdP might supplement the combined set with the attributes of the user stored at its end and then reasserts all attributes to the SP as its own attributes using a single assertion. The SP validates the assertion, retrieves all attributes from the assertion and, not being aware of other IdPs, assumes that all attributes have been released by the trusted IdP. Based on the combined attributes, the SP determines if the user can access its service(s).

Identity Relay (IR) Model. The Identity Relay model is a generalised case of the Identity Proxying model (Figure 8.9) [58, 59]. Since the IP model requires an SP to have a strong trust in the trusted IdP, it cannot function properly in situations when the Proxy IdP cannot be fully trusted. The Identity Relay model fits in such scenarios where an intermediary IdP (or Relay IdP), (IdP1 in Figure 8.9) is used instead of a trusted IdP. A user is forwarded to the relay IdP and then the relay IdP forwards the user to other multiple IdPs. The user is authenticated separately at each IdP and is returned back to the Relay IdP with assertions including user-attributes. The relay IdP combines all assertions into a single assertion and forwards it to the SP. The SP extracts embedded assertions from this assertion and validates each assertion to retrieve attributes from other IdPs. Based on the combined attributes, the SP determines if the user can access its service(s).

8.2.4 Aggregation at a Client

In this model, a client (e.g. a browser) aggregates attributes on behalf of a user.

Client-Mediated (CM) Model. This model is similar to the Identity Relay Model. Here, the functionality of a relay IdP has been replaced by an intelligent user-agent or application (known as the client) that has the capability to aggregate attributes from different IdPs [58, 59]. An SP informs the client about the IdPs that it trusts. The client forwards the user to each of these IdPs. After respective authentication at each IdP, the client receives assertions from all IdPs and present the combined set of assertions to the SP. The SP validates each assertion, retrieves all attributes and then determines if the user can access its service(s).

8.2.5 Comparative Analysis

Each model discussed above has their own strengths and weaknesses. A side-by-side comparison of strengths and weaknesses of each model is presented in Table 8.1. Unfortunately, it is evident from the table that no one model is superior to other models. Extensive research is needed to come up with a model that offers a good number of advantages. According to the analysis, it has been found that the Identity Proxying and the Identity Relay models are good research candidates since both are easy to deploy for an SP, offer a good number of advantages and have relatively fewer disadvantages. Therefore, the implementation is based on these two models. Before the implementation is discussed, it is investigated how attributes are released in the existing online (federated and non-federated) services and whether any of them has any support for attribute aggregation.

8.3 The State of the Art

It has been mentioned earlier that SAML is one of the most widely-used technologies for authenticating a user, releasing user attributes to an SP and for accessing federated services. In addition, many popular social networks utilise OpenID and OAuth protocols to authenticate users and release their attributes to third party service providers. Frameworks are available

Table 8.1: Aggregation Models: Strengths and Weaknesses.

Models	Advantages	Disadvantages
AD	Easy to implement and maintain. Small number of requirements.	Aggregation from only one IdP in a session. Changing identifiers causes the system to fail.
SPM	Aggregation from a number of IdPs in a session.	Hard to maintain. Potentially privacy invasive. Extensive changes are required at the SP. No implementation exists.
LS	Secure and Privacy-preserving. Proof of concept implementation available. Attribute aggregation from multiple IdPs in a single session. SSO capability during service access.	Hard to deploy and difficult to maintain. The LS represents a single point of failure.
IFL	Secure and Privacy-preserving. Proof of concept implementation available. SSO capability during service access.	Difficult to maintain and scale. Attribute aggregation from only two IdPs in a single session.
IP	Attribute aggregation from multiple IdPs in a single session. An SP utilising this model is easy to maintain. Relatively small number of requirements.	Strong trust is required on the Proxy IdP. The Proxy IdP is the single point of failure. No implementation based on Shibboleth and SimpleSAMLphp.
IR	Attribute aggregation from multiple IdPs in a single session. An SP utilising this model is easy to maintain. Relatively small number of requirements. Less trust on the relay IdP.	The relay IdP is the single point of failure. Hard to maintain. No implementation based on Shibboleth and SimpleSAMLphp.
CM	No need to rely on external IdP. Attribute aggregation from multiple IdPs in a single session. Relatively small number of requirements.	No implementation exists currently. Extensive changes are required to ensure that IdPs, SPs and clients can interact.

that allow the integration of such social networks within SAML federations to offer federated services [106]. However, how attributes are released using these protocols or if such protocols can be used to aggregate attributes in a federated setting have not been considered. These issues are now examined next.

8.3.1 SAML

How and what attributes are requested and released in SAML depend on a specific implementation. There are many different implementations of SAML including Shibboleth [109], SimpleSAMLphp [11] and ZXID [110]. One approach to request attributes as adopted by

Shibboleth is using an *AttributeQuery* request which is initiated once the user is authenticated by an IdP and the IdP has returned an assertion to an SP. Then, the SP requests a few attributes using the *AttributeQuery* request and the IdP replies back with an assertion containing the requested attributes. On the other hand, SimpleSAMLphp has adopted a different approach in which, without using a separate *AttributeQuery* request, the attributes are embedded inside the same assertion which is returned in response to the SAML authentication request. This is a reasonable approach as it reduces protocol flows. However, there are drawbacks: an IdP has no way of knowing what attributes an SP is requesting and hence the user does not know which attributes are required by an SP to access a particular service. One adhoc approach has been used in Chapter 7 where an SP enlists, at its homepage, the attributes required for accessing any of its services. This is not very efficient as users need to remember the list of attributes needed to release to an SP.

Also, the way in which attributes are selected is not standardised in SAML and depends entirely on the specific implementation. Shibboleth uses the Attribute Release Policy (ARP) [140], known as the Attribute Filter Policy (AFP) in Shibboleth V 2.0 [141]. An ARP/AFP allows an administrator to setup a policy that determines which attributes can be released to an SP. However, there are third-party extensions such as ShARPE [142] that allows a user to set up an ARP. An alternative approach has been used by SimpleSAMLphp using a module, called the *Consent* module, that allows a user to select attributes in real time once the user is authenticated at an IdP with no need to setup the policy beforehand. The SAML specification has no mechanism to aggregate attributes, nor does any SAML implementation provides this facility. However, the implementations of existing models of attribute aggregation are mainly based on SAML.

8.3.2 OpenID

The original OpenID protocol was intended for authentication only and hence, the aspect of exchanging attributes was not initially considered. However, two extensions, the OpenID Simple Registration Extension (SREG) [143] and the OpenID Attribute Exchange (AX) [124] allow an SP to request and fetch attributes from a provider. The provider might ask a user to release the requested attributes which are then sent back to the SP embedded inside the response.

At this point, it is investigated how a few popular OpenID providers act when they receive a request to return attributes. Especially, the focus is to investigate how each provider allows a user to selectively choose their attributes individually (the selective disclosure of attributes). With these in mind, five popular OpenID providers - *Yahoo* [144], *Google* [145], *LiveJournal* [146], *myopenid* [147] and *VeriSign* [148] - have been selected. Then, a SAML SP and IdP using the SimpleSAMLphp implementation has been deployed. Even though the IdP and SP are based on SAML, the SimpleSAMLphp allows different types of (e.g. OpenID or OAuth enabled) IdPs to be configured as authentication sources at an IdP. These authentication sources can be used by a user to be authenticated at another IdP. Once authenticated, the user returns back to the SimpleSAMLphp IdP with attributes. This approach has been used for two reasons:

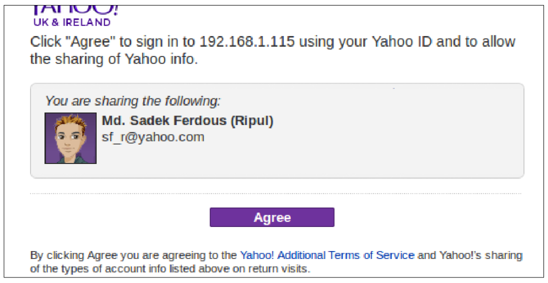
1. the support of such IdPs are built in SimpleSAMLphp and hence, it is not needed to deploy a separate OpenID or OAuth SP, and
2. a very similar setup will be used for implementing the proof of concept.

The SimpleSAMLphp OpenID module had been configured to request *openid, email, fullname* and *age* attributes among which only the *age* attribute was optional. A typical flow using this approach is as follows: a user visits the deployed SP and then chooses the SAML IdP. When the user is forwarded to the SAML IdP, she chooses OpenID as the authentication source and then the user provides her OpenID and the usual OpenID protocol flows take place.

Dummy user accounts have been created in the chosen providers to use during the experiments. It has been noted how the provider allows a user to authorise it to release her attributes, after the user is authenticated or during the user authentication process. The screenshots while accessing different providers are captured and presented in Figures 8.10 to 8.14. Two problems are noticed: the login screen does not inform the user which individual attributes have been requested; therefore, the user has no way of knowing what information will be released. Also, none of the providers except VeriSign has any support of selective disclosure. This practice not only violates the principle of data minimisation but also forces the user to release attributes based on obscure information. This may have serious effect on the privacy of the user. Also, note that the OpenID Specification does not have any mechanism to aggregate attributes from other providers and hence, none of the implementations has any such facility.

8.3.3 OAuth

Unlike OpenID, OAuth does not require any separate mechanism to exchange attributes since all user attributes are considered as resources which can be accessed using OAuth upon approval from the user. With the same intention as OpenID, it it observed how a user can grant authorisation to a client - an entity that wants to access some attributes of the user. Three popular IdPs based on OAuth - *Facebook* [149], *Linkedin* [150] and *Twitter* [151] - have been chosen. The same SimpleSAMLphp setup to initiate the OAuth protocol flow has been used. A typical flow takes the following form: a user visits an SP and then chooses


YAHOO! UK & IRELAND

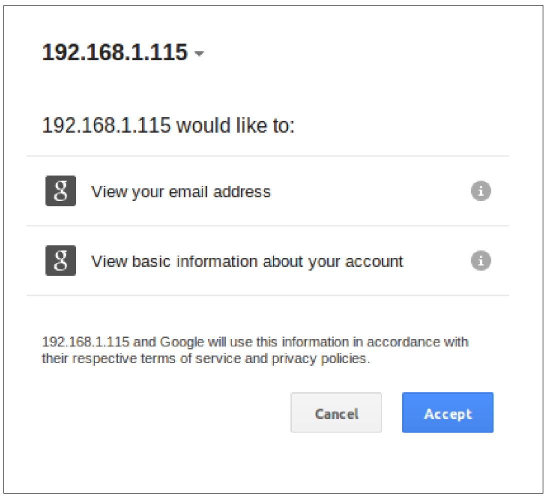
Click "Agree" to sign in to 192.168.1.115 using your Yahoo ID and to allow the sharing of Yahoo info.

You are sharing the following:
Md. Sadek Ferdous (Ripul)
sf_r@yahoo.com

Agree

By clicking Agree you are agreeing to the Yahoo! Additional Terms of Service and Yahoo!'s sharing of the types of account info listed above on return visits.

Figure 8.10: Yahoo Login screen.


192.168.1.115

192.168.1.115 would like to:

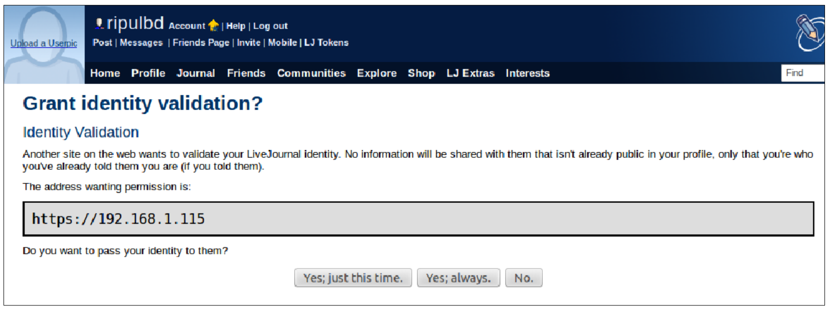
View your email address

View basic information about your account

192.168.1.115 and Google will use this information in accordance with their respective terms of service and privacy policies.

Cancel Accept

Figure 8.11: Google Login screen.


ripulbd Account | Help | Log out
Post | Messages | Friends Page | Invite | Mobile | LJ Tokens
Home Profile Journal Friends Communities Explore Shop LJ Extras Interests
Upload a Userpic

Grant identity validation?

Identity Validation

Another site on the web wants to validate your LiveJournal identity. No information will be shared with them that isn't already public in your profile, only that you're who you've already told them you are (if you told them).

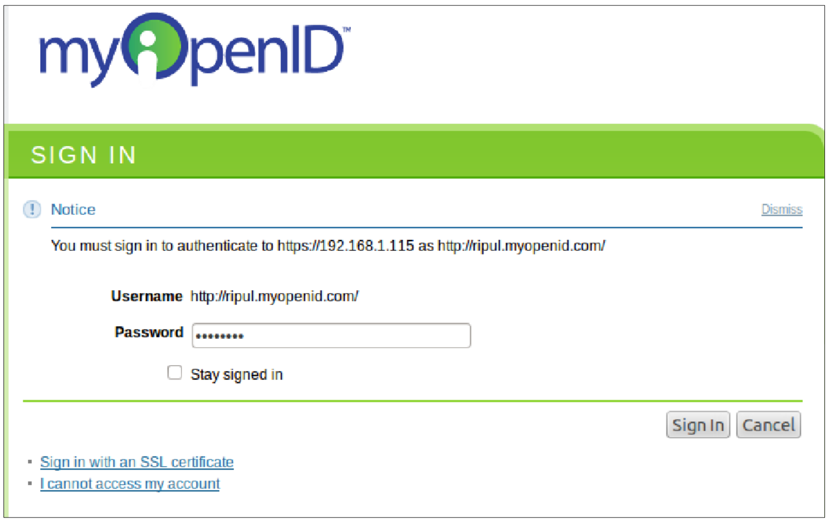
The address wanting permission is:

https://192.168.1.115

Do you want to pass your identity to them?

Yes; just this time. Yes; always. No.

Figure 8.12: LiveJournal Login screen.


myOpenID

SIGN IN

Notice Dismiss

You must sign in to authenticate to https://192.168.1.115 as http://ripul.myopenid.com/

Username http://ripul.myopenid.com/

Password ••••••••

Stay signed in

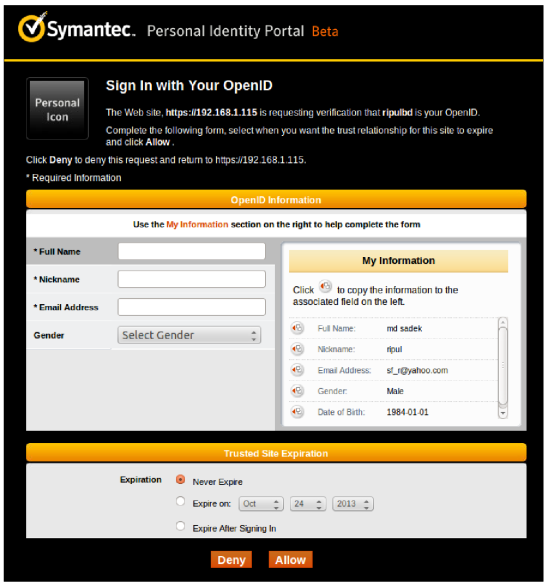
Sign In Cancel

- Sign in with an SSL certificate
- I cannot access my account

Figure 8.13: myopenid Login screen.

the SAML IdP. When the user is forwarded to the SAML IdP, she chooses one of the OAuth providers (Facebook, Twitter and Linkedin) as the authentication source and then the usual OAuth protocol flow takes place.

To initiate this protocol flow, a respective application needs to be registered at each provider. Once the application is registered, an application key and a corresponding *secret* are gen-



Sign In with Your OpenID

The Web site, **https://192.168.1.115** is requesting verification that **ripulbd** is your OpenID.

Complete the following form, select when you want the trust relationship for this site to expire and click **Allow** .

Click **Deny** to deny this request and return to https://192.168.1.115.

* Required Information

OpenID Information

Use the My Information section on the right to help complete the form

* Full Name	
* Nickname	
* Email Address	
Gender	Select Gender

My Information

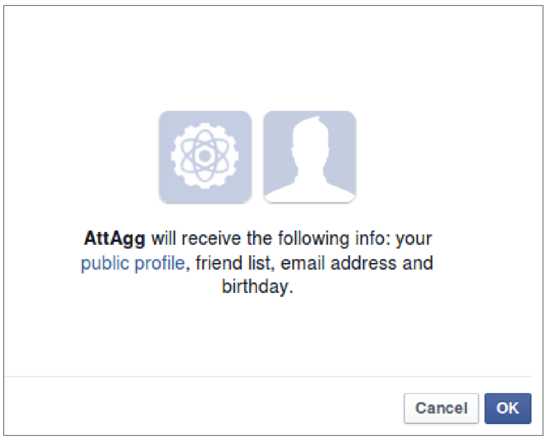
Click to copy the information to the associated field on the left.

Full Name:	md sadek
Nickname:	ripul
Email Address:	sf_r@yahoo.com
Gender:	Male
Date of Birth:	1984-01-01

Trusted Site Expiration

Expiration [✓ Never Expire] Expire on: Oct 24 2013 Expire After Signing In

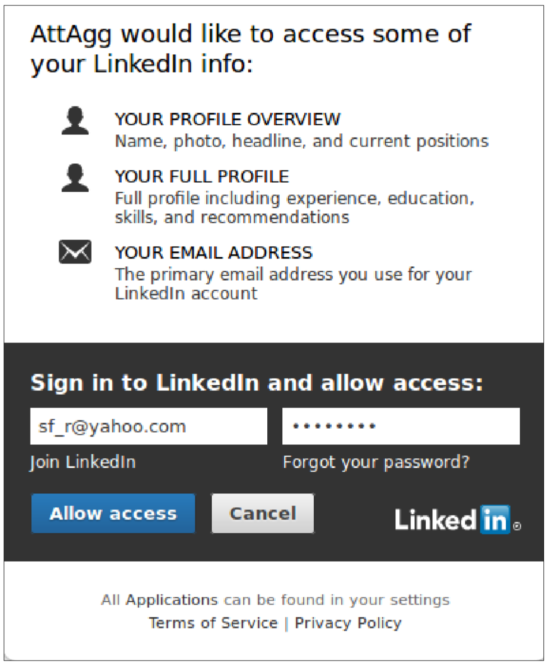
Figure 8.14: VeriSign Login screen.



AttAgg will receive the following info: your public profile, friend list, email address and birthday.

Figure 8.15: Facebook Authorisation screen.

erated. These two pieces of information are then used at the SimpleSAMLphp to initiate the protocol flow at the respective IdP. In addition, the way in which user attributes can be requested using a registered application depends entirely on the respective provider. For Facebook, the application can be configured to request different user attributes or information regarding requested attributes can be passed as configuration parameters during initiating the protocol. The second approach has been adopted for the investigation. For Linkedin, the application needs to be configured to request attributes. The application has been configured to request the full public profile and the email address attributes. For Twitter, the public user attributes have been requested from the SimpleSAMLphp. As before, the process by which the user allows the application to access the requested attributes have been noted and screenshots have been captured. The screenshots are presented in Figures 8.15, 8.16 and 8.17. It has been found that, like OpenID providers, the same two problems exist: the login screen does not inform a user which individual attributes have been requested (for example, in Figure 8.15 it is not clear which information from the public profile will be released) and none of the providers has any support for the selective disclosure of attributes. The user needs to release either all attributes or none. This approach also violates the principle of data minimisation and forces the user to release attributes based on obscure information and thus can have negative effect on the privacy of the user. OAuth also does not have any support of attribute aggregation.



AttAgg would like to access some of your LinkedIn info:

YOUR PROFILE OVERVIEW
Name, photo, headline, and current positions

YOUR FULL PROFILE
Full profile including experience, education, skills, and recommendations

YOUR EMAIL ADDRESS
The primary email address you use for your LinkedIn account

Sign in to LinkedIn and allow access:

sf_r@yahoo.com ••••••••

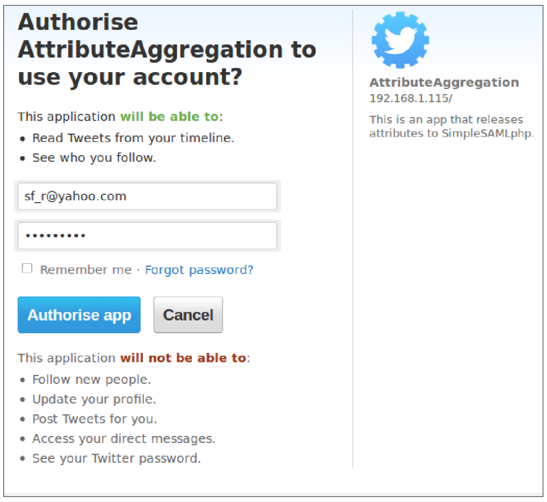
Join LinkedIn Forgot your password?

Allow access Cancel

LinkedIn

All Applications can be found in your settings
Terms of Service | Privacy Policy

Figure 8.16: Linkedin Authorisation screen.



Authorise AttributeAggregation to use your account?

This application **will be able to**:
- Read Tweets from your timeline.
- See who you follow.

sf_r@yahoo.com
•••••••••

Remember me · Forgot password?

Authorise app Cancel

This application **will not be able to**:
- Follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.
- See your Twitter password.

AttributeAggregation
192.168.1.115/
This is an app that releases attributes to SimpleSAMLphp.

Figure 8.17: Twitter Authorisation screen.

8.4 The Hybrid Model

Now, a model for attribute aggregation is introduced for overcoming the issues discussed above. The model is called the *Hybrid Model* and is based on the combination the IP and the IR models. In this model, the trusted IdP will be called the ***Hybrid IdP***.

8.4.1 Requirement Analysis

Before designing such a model, a list of requirements that the model should fulfil are identified using the taxonomy of requirements of an ideal IMS as presented in Chapter 4. The requirements have been formulated assuming that there is an SP offering different services. To access these services, users need to aggregate attributes from multiple SAML and/or non-SAML IdPs using the Hybrid IdP.

Functional Requirements (FR). The functional requirements ensure that a system behaves as desired. The requirements are:

F1 **Federation Requirement.** The Hybrid IdP and the SP must be a part of the same federation. All SAML IdPs as well as the Hybrid IdP must be a part of a SAML federation. For enabling the capability of the IR model so that IdPs can return encrypted assertions to the SP, such SAML IdPs also need to be a part of a SAML federation with the same SP.

F2 **Session Retention.** A session is maintained at the Hybrid IdP so that it can correlate the attributes from an IdP with attributes retrieved previously from other IdPs.

F3 **Dual Capabilities.** The Hybrid IdP must have the dual capabilities of an IdP and an SP. The Hybrid IdP has to act as an IdP to the SP and as an SP to other IdPs.

F4 **Assertion Target.** For the IP model, the assertions returned by other IdPs should be targeted for the Hybrid IdP so that it can validate each assertion, extracts attributes from them and then aggregates all of them, possibly also with its own attributes. For the IR model, the encrypted assertions returned by other SAML IdPs should be targeted for the SP. The Hybrid IdP will just aggregate all assertions and embed them inside another assertion and send it back to the SP. The SP will validate the outer assertion and retrieve all embedded assertions. Then, it must validate each assertion in turn to extract attributes from them.

F5 **Heterogeneous IdP.** The Hybrid IdP should allow a user to aggregate attributes from SAML as well as non-SAML IdPs.

F6 **Source of Aggregated Attributes.** The existing proxy model assumes that the aggregated attributes, from SAML IdPs, are combined to build a single assertion to pass over to the SP. The effect of this assumption is that the original sources of the attributes are lost, leaving the SP with the notion that the attributes have originated from the trusted Proxy IdP (Hybrid IdP in this model). This is a risky assumption to make which might escalate privileges since some (or even all) attributes might originate from the SAML IdP which the SP might not trust at all. To prevent the SP from making such risky assumptions, the Proxy IdP must indicate the source of all attributes. The SP should have the capabilities to interpret such groupings to ensure that it can easily identify the source of each attribute and the associated LoA value.

F7 **IP/IR Model Switching.** A user should have the ability to choose between the IP and IR models. If the user chooses the IR model, it means that she wants the other IdPs to release encrypted assertions, instead of regular unencrypted assertions, to the SP via the Hybrid IdP. In such cases, the Hybrid IdP should forward the information regarding the SP to the other IdPs to enable them to generate the encrypted assertions properly.

Security Requirements (SR).

S1 **Secure User Authentication.** Every user should be securely authenticated at all IdPs from where attributes are aggregated.

S2 **Confidentiality, Integrity and Authenticity.** Aggregated attributes should be transmitted between different entities, maintaining their confidentiality, integrity and authenticity.

S3 **Transient Storage.** To ensure the security of attributes, the Hybrid IdP must not store the aggregated attributes (assertions for the IR model) once the session with the Hybrid IdP is terminated.

S4 **LoA Calculation.** The LoA values for each group of attributes from a specific IdP are properly determined.

Privacy Requirements (TR). There is no additional privacy requirements other than the ones outlined in the taxonomy of Chapter 4. Hence, it must be ensured that the existing requirements - Support of Pseudonym (denoted as *P1*), Selective Disclosure (denoted as *P2*), Explicit Consent (denoted as *P3*) and Data Minimisation (denoted as *P4*) - are satisfied.

Trust Requirements (TR). Like the implementation discussed in Chapter 6 and 7, the trust requirement while using the system is expressed using a trust metric: the NIST LoA value from 1 to 4. The responsibility lies on the Hybrid IdP to calculate the respective trust metric for each group of attributes from a specific IdP and embed that metric in the assertion. The SP then can use the trust metric to determine the level of trust over a set of attributes from each IdP.

8.4.2 Architecture

The architecture of the Hybrid Model is illustrated in Figure 8.18. The dotted area in the figure represents the Circle of Trust (i.e. the federation) between entities. The Hybrid IdP

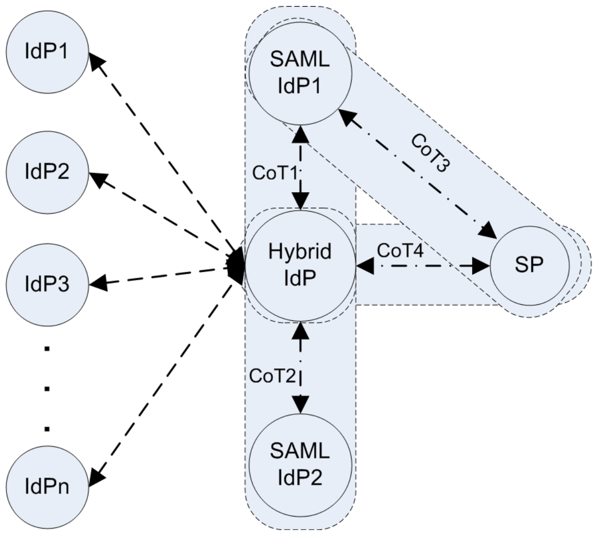


Figure 8.18: Architecture of Hybrid Model.

takes the central stage in the model. Any number of SAML and non-SAML IdPs can be added with the Hybrid IdP. These IdPs will act as third party IdPs and a user can aggregate attributes from these IdPs using the Hybrid IdP. Any such third-party SAML IdP (*SAML*

IdP1 and *SAML IdP2* in Figure 8.18) must be federated with the Hybrid IdP, while there is no need for any non-SAML IdP (*IdP1, IdP2, ... , IdPn* in Figure 8.18) to be federated with the Hybrid IdP since they can be accessed using their respective protocols. In addition, the SP needs to be federated with the Hybrid IdP (as illustrated in Figure 8.18) and any other third-party SAML IdP to enable users to release encrypted assertions for the SP. The absence of any federation between the SP and any third-party SAML IdP will prohibit the user from releasing any encrypted assertions from those IdPs. For example, in Figure 8.18, the SAML IdP1 can release encrypted assertions to the SP whereas the SAML IdP2 cannot release any encrypted assertions to the SP. The whole architecture has been designed in such a way that it satisfies all the requirements listed above. In the subsequent section, it will be explored how the design choices have achieved this.

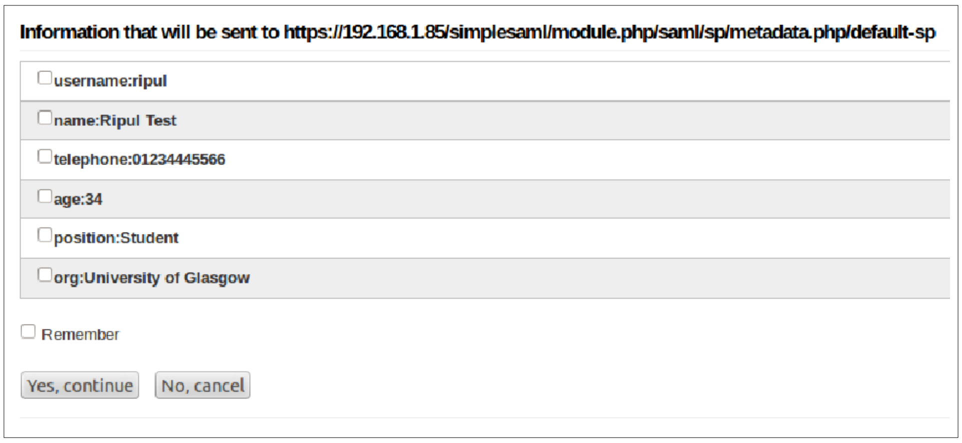
8.4.3 Implementation

SimpleSAMLphp has been used for the implementation. This is for two reasons:

1. SimpleSAMLphp has built-in support for the selective disclosure of attributes based on their *Consent* module, and
2. SimpleSAMLphp also has another built-in module called the *Multiauth* module that allows a SAML IdP to delegate the authentication task to an external SAML or non-SAML IdP (e.g. OpenID or Facebook). Once the user has authenticated at the external IdP and returned back to the SAML IdP with attributes, it releases the attributes to the SP via a SAML assertion.

The second module allows an IdP to act as a Hybrid IdP with the Identity Proxying capability. However, it does not allow a user to choose more than one IdP for a particular session and hence, attribute aggregation from multiple IdPs is not possible. In addition, it cannot return an encrypted assertion to an SP; this means that it cannot act as a Relay IdP. Even with these limitations, these two modules would give a solid platform on which to base the implementation, with the main focus to confine most of the changes to the Consent and the Multiauth modules. The idea is to modify the core code-base of SimpleSAMLphp only when this is absolute necessary.

In SimpleSAMLphp, the Consent module is loaded to show a consent form once a user is authenticated at an IdP. The consent form displays the list of attributes of the user stored at the IdP and allows the user to choose the attributes that she wishes to release to an SP. A sample consent form is shown in Figure 8.19. The Consent module has been modified by adding two new inputs - a check box and a button - so that the consent form acts as the entry point for allowing users to aggregate attributes. The modified consent form is illustrated



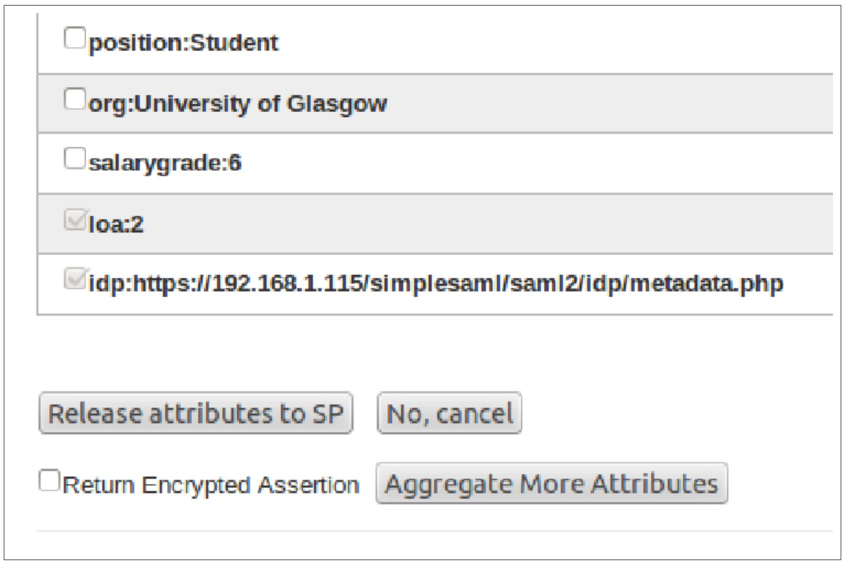
Information that will be sent to https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

☐ username:ripul
☐ name:Ripul Test
☐ telephone:01234445566
☐ age:34
☐ position:Student
☐ org:University of Glasgow

☐ Remember

Figure 8.19: The usual consent form.

in Figure 8.20. The check box with the text *Return Encrypted Assertion* is used to toggle between the IP and IR mode and the *Aggregate More Attributes* button is used to initiate the process of attribute aggregation.



☐ position:Student
☐ org:University of Glasgow
☐ salarygrade:6
☑ loa:2
☑ idp:https://192.168.1.115/simplesaml/saml2/idp/metadata.php

☐ Return Encrypted Assertion

Figure 8.20: The modified consent form.

The original Multiauth module enables an IdP to act as a Proxy IdP to delegate the authentication to another IdP. A list of different IdPs is shown to a user at an IdP. Once the user selects an IdP, she is forwarded to the chosen IdP where the user authentication takes place. The user is then redirected back to the Proxy IdP where a session is created. Once the session is created, the user cannot choose another IdP without logging out from the Proxy IdP. This behaviour needs to be changed to allow users to choose other IdPs even if there is already a session at the Proxy IdP. For this, two amendments have been made:

1. the Multiauth module has been modified so that it shows the list of external IdPs, excluding those IdPs where a user has already been authenticated at, when the user clicks the *Aggregate More Attributes* button at the consent page, and

2. The SimpleSAMLphp IdP code-base has been modified to allow a user to initiate an authentication process even if the user has a session at the IdP and to retain that session when the user is redirected back from the chosen external IdP. This allows the IdP to collate attributes from the recently authenticated IdP with the attributes aggregated previously.

Note that the IP model is based on the first interpretation of attribute aggregation where different attributes are combined in a single assertion. As such, when attributes are aggregated from multiple sources at the Hybrid IdP and passed to an SP inside a single assertion, it will be impossible for the SP to identify the source of each single attribute. Without identifying the source of each attribute, it might be difficult for the SP to take access control decisions. None of the previous implementations considered this crucial issue. One way to retain the source of attributes is to group them according to their sources and the only place this can be done properly is at the Hybrid IdP. This is because only the Hybrid IdP has access to all aggregated attributes. In addition, the Hybrid IdP can also determine the LoA value for each IdP and add this information to each group. A data structure is needed to hold all this information. One option for such a data structure is a list of lists where each entry of the list is a list holding the attributes from a particular IdP along with its LoA value. The structure is illustrated in Figure 8.21.

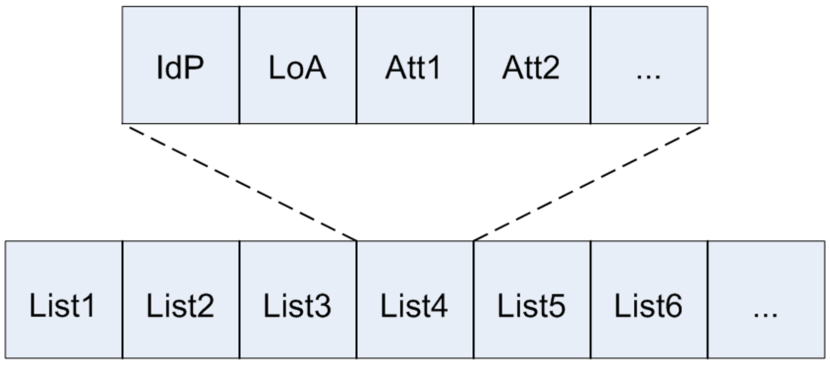


Figure 8.21: The data structure for the attribute list.

In the current SAML specification, each attribute with its value, is inserted inside an *AttributeStatement* element [5]. The schema for the AttributeStatement is given in Listing 8.1. If the grouped attributes following the proposed data structure are inserted in the AttributeStatement element in this manner, there are chances that each group may get mixed up with each other. To mitigate this problem, the inclusion of a new SAML element called ***AttributeStatements*** is proposed. The schema of the proposed element is given in Listing 8.2. The *AttributeStatements* element will essentially accommodate one or more AttributeStatement element(s). In this way, the Hybrid IdP can include the IdP and LoA information for each group of attributes inside each AttributeStatement element. The IdP and the SP code-bases of the SimpleSAMLphp have been modified to reflect this mechanism.

The next amendment made is to handle the IR mode which is based on the second interpretation of the attribute aggregation where multiple assertions containing different attributes are combined into a single assertion. As such, the relay mode will enable external SAML IdPs to release encrypted assertions to the Hybrid IdP. To do so, two things are required:

1. a mechanism to signal to the external IdP that the Hybrid IdP is requesting an encrypted assertion, and

2. the Entity ID of the SP so that the external IdP can encrypt an assertion that is only decryptable by the SP.

Since the SP is only communicating with the Hybrid IdP, not with the external IdP, the Hybrid IdP has the responsibility of passing on these two pieces of information. As stated earlier, a check box in the consent form is used to toggle between the IP and the IR mode to indicate if a user wants to receive an encrypted assertion. Once the checkbox is ticked and the user clicks the *Aggregate More Attributes* button, an attribute called ***EmbedAssertion*** is added to the SAML authentication request which contains the Entity ID of the SP.

Listing 8.1: Schema for AttributeStatement Element

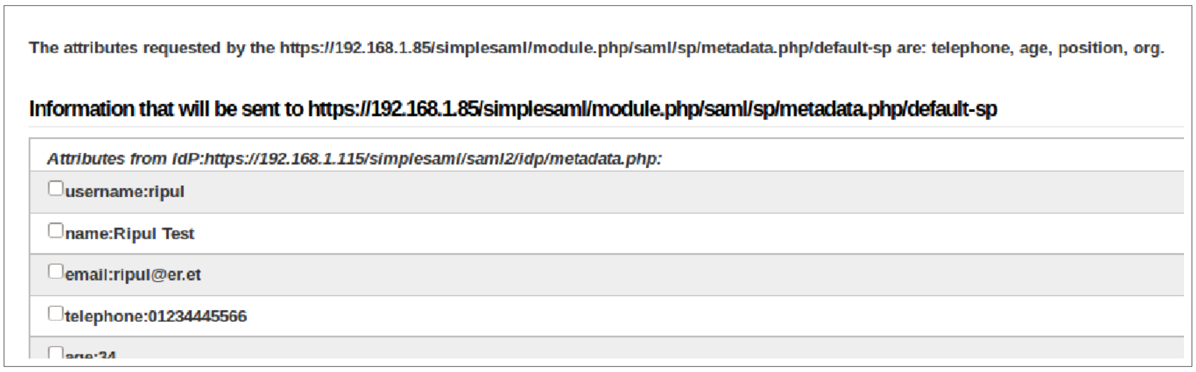
```
<element name="AttributeStatement" type="saml:AttributeStatementType"/>
<complexType name="AttributeStatementType">
  <complexContent>
  <extension base="saml:StatementAbstractType">
    <choice maxOccurs="unbounded">
      <element ref="saml:Attribute"/>
      <element ref="saml:EncryptedAttribute"/>
    </choice>
 </extension>
 </complexContent>
 </complexType>
```

Listing 8.2: Proposed Schema for AttributeStatements Element

```
<element name="AttributeStatements" type="saml:AttributeStatementsType"/>
<complexType name="AttributeStatementsType">
  <complexContent>
  <extension base="saml:StatementAbstractType">
    <choice minOccurs="1" maxOccurs="unbounded">
      <element ref="saml:AttributeStatement"/>
    </choice>
 </extension>
 </complexContent>
 </complexType>
```

When an external IdP receives a SAML authentication request, it checks for an attribute called ***EmbedAssertion*** in the request. If this attribute is not found, the IdP behaves as usual. However, if the attribute is found, the external IdP knows that it has to release an encrypted assertion. Then, the Entity ID of the SP that sent the request is retrieved from that attribute which is then used to create an encrypted assertion. To transmit the encrypted assertion, it is Base64 encoded and added as the value of a special type of attribute called ***encryptedAssertion*** into an *AttributeStatement* element which is then embedded inside a regular unencrypted SAML assertion. This regular assertion is then passed to the Hybrid IdP. After receiving an assertion from an external IdP, the Hybrid IdP validates it and if it finds that there is an attribute called ***encryptedAssertion***, it is treated differently (see below). Otherwise, the attributes retrieved from the external IdP are aggregated with the previously aggregated attributes and are presented in the consent form. Necessary amendments to the SimpleSAMLphp code-base have been added to reflect this behaviour.

To achieve data minimisation and selective disclosure of attributes and to provide an optimal way of aggregating attributes, a user must know beforehand the attributes required by an SP to access its services. As mentioned earlier, one way to achieve this is to show the list of attributes for each specific service at the home page of an SP as explained in Chapter 7. However, it is unrealistic to assume that a user will remember such a list for each service. A draft to extend the SAML AuthnRequest element has been proposed in [152] to include a SAML AttributeQuery statement within the authentication request. This approach has been adopted in the implementation. The SimpleSAMLphp SP code-base has been modified to include the AttributeQuery statement, containing the list of required attributes, with the SAML AuthnRequest statement at the SP side. Furthermore, the IdP code-base has been modified as to parse the list of attributes from the AttributeQuery statement of the authentication request at the IdP side and then to show them at the top of the consent form (Figure 8.22). It is believed


The attributes requested by the https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp are: telephone, age, position, org.

Information that will be sent to https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

Attributes from IdP:https://192.168.1.115/simplesaml/saml2/idp/metadata.php:

☐username:ripul

☐name:Ripul Test

☐email:ripul@er.et

☐telephone:01234445566

☐age:34

Figure 8.22: The requested attributes at the consent form.

that the consent form is the best place to show the required attributes as it will allow a user to choose the IdP from where she can aggregate attributes as well as to decide the minimum number of attributes she needs to release to an SP. In addition, when a user chooses another external SAML IdP for attribute aggregation, the requested attributes will also be included within the AttributeQuery element of the SAML AuthnRequest that will be sent to an external IdP. This will enable any external SAML IdP to parse and show the requested attributes at any appropriate place. Moreover, the capability has been added to the SP code-base to allow an administrator to set the required attributes for each SP in the configuration file (called *config.php* in SimpleSAMLphp). Then this file can be used to configure the AttributeQuery element with the requested attributes to be embedded inside the AuthnRequest statement as described before. In this manner, the administrator can set different attribute requirements for different services for each SP.

The current implementation equipped with all these features can aggregate attributes from SAML IdPs, IdPs based on OpenID including Google and IdPs based on OAuth such as Facebook, Twitter and Linkedin. The implementation supports two types of SAML IdPs: trusted and semi-trusted. A trusted SAML IdP is the one which has been federated in the traditional way by exchanging metadata at the administrative level whereas a semi-trusted SAML IdP is the one that has been federated using the concept of dynamic federation as discussed in Chapter 5.

8.5 Use-cases

To demonstrate the applicability of the proposed model in different usage scenarios, three use-cases are described below.

8.5.1 Use-case 1

The first use-case illustrates the simplest form of attribute aggregation using the Hybrid model. The initial setup is that a user wants to access a service from an SP. The SP requires a set of user attributes from different IdPs. The SP is deployed using the modified SimpleSAMLphp and is federated with the Hybrid IdP. Another SAML IdP is also implemented using SimpleSAMLphp. Moreover, the administrator of the SP has set the required attributes for each service in the configuration file as discussed above. With this setup, the protocol flow for the first use-case is illustrated in Figure 8.23 and is described below:

1. A user visits the SP to access one of its services. The user clicks the respective service and is forwarded to the WAYF Page of the SP to choose an IdP. The user chooses the Hybrid IdP.

2. The required attribute(s) for the requested service are read from the configuration file and are used to create a SAML authentication request. The user is redirected to the Hybrid IdP with a SAML authentication request.

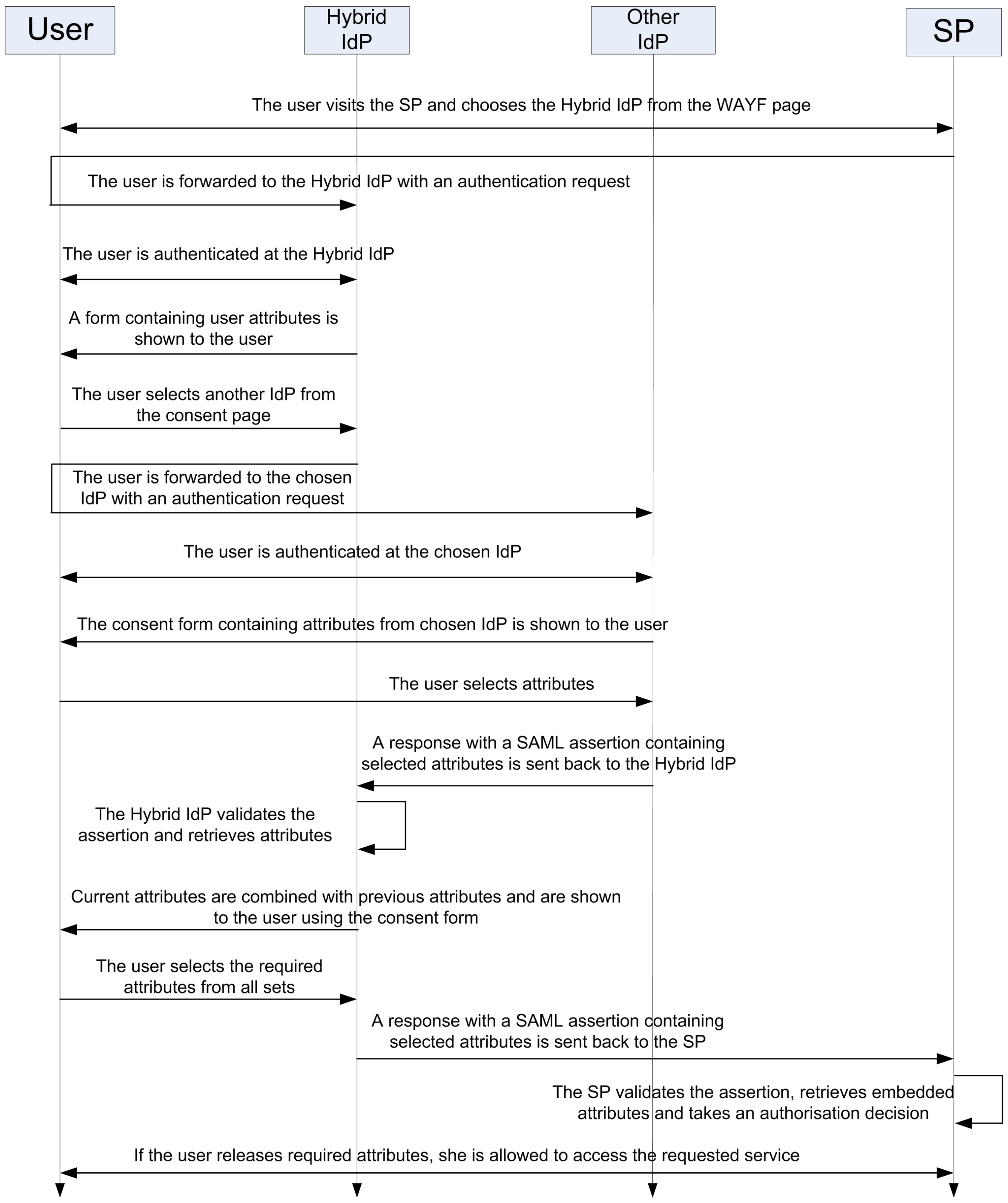

User
Hybrid IdP
Other IdP
SP
The user visits the SP and chooses the Hybrid IdP from the WAYF page
The user is forwarded to the Hybrid IdP with an authentication request
The user is authenticated at the Hybrid IdP
A form containing user attributes is shown to the user
The user selects another IdP from the consent page
The user is forwarded to the chosen IdP with an authentication request
The user is authenticated at the chosen IdP
The consent form containing attributes from chosen IdP is shown to the user
The user selects attributes
A response with a SAML assertion containing selected attributes is sent back to the Hybrid IdP
The Hybrid IdP validates the assertion and retrieves attributes
Current attributes are combined with previous attributes and are shown to the user using the consent form
The user selects the required attributes from all sets
A response with a SAML assertion containing selected attributes is sent back to the SP
The SP validates the assertion, retrieves embedded attributes and takes an authorisation decision
If the user releases required attributes, she is allowed to access the requested service

Figure 8.23: Protocol flow for Use-case 1.

3. The user is authenticated at the Hybrid IdP and the consent form is shown. The consent form displays the requested attributes from the SP (Figure 8.22). In addition, the consent form allows the user to aggregate more attributes from other IdPs (Figure 8.20). The user compares the requested attributes and the attributes shown in the consent form to decide if she needs to aggregate attributes.

4. Assuming that the user decides to aggregate attributes, she clicks the *Aggregate More Attributes* button. A session is created to keep track of the previous attributes and then

the user is forwarded to the IdP selection page of the Hybrid IdP. This page contains the list of those IdPs from where the user has not aggregated attributes in the current session.

5. The user chooses one IdP and based on the user's selection, the respective protocol is initiated. For example, if the user chooses another SAML IdP, a SAML authentication request is created and the user is forwarded to the respective IdP. If the user chooses OpenID, a HTML form is shown to the user to input the user's OpenID. Once the OpenID is provided and the Login button is clicked, the usual OpenID protocol starts and the user is forwarded to the OpenID provider. If the user chooses an OAuth-based IdP she is forwarded to the respective IdP following the OAuth protocol.

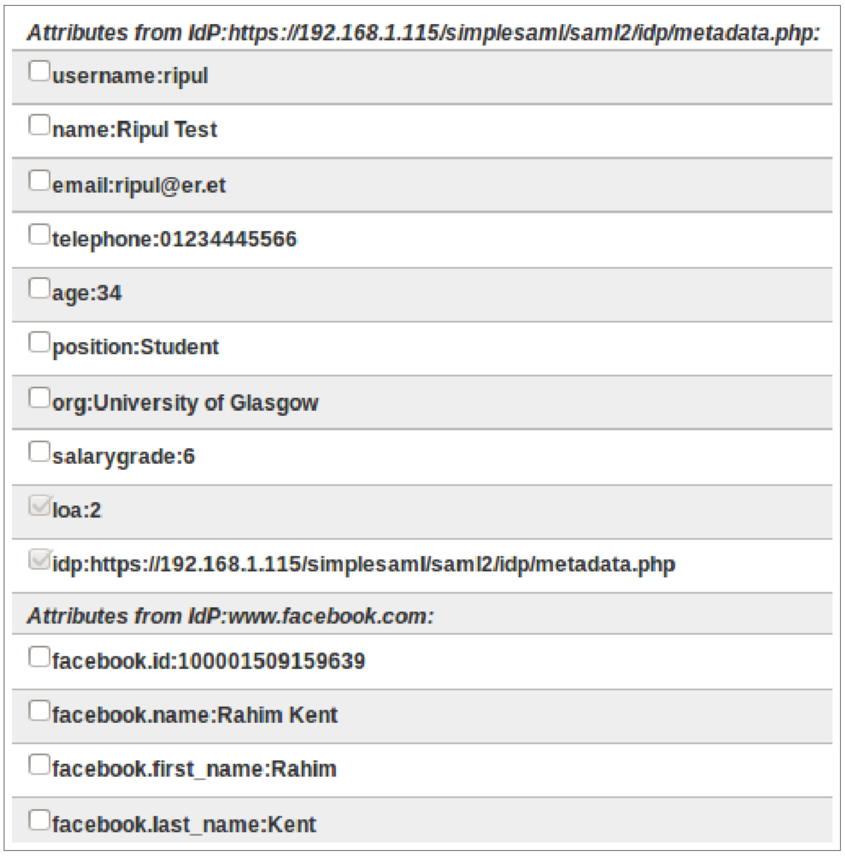
6. The user authenticates at the respective IdP and releases the attributes to the Hybrid IdP using their respective protocols and implementations. For example, if the user chooses the SAML IdP, the user is shown a basic consent form (cf. Figure 8.19) where she chooses the attributes. Based on her selection, a SAML response is returned to the Hybrid IdP. For other OpenID or OAuth IdPs the attributes are released using the methods described previously.

7. Once the user returns to the Hybrid IdP, it validates the assertion (in case of SAML) or the security token/information (for other protocols) using their corresponding mechanisms. Then, the attributes are retrieved from the assertion or the security token and the previously aggregated attributes are retrieved. The two sets of attributes are then merged and are shown to the user by grouping them based on the IdP in the consent form (Figure 8.24). The Hybrid IdP determines the LoA for each IdP. Note that the IdP attribute and the LoA attribute for each attribute group are disabled so that the user cannot deselect them. If the user chooses to release at least one attribute from an attribute group, the respective IdP and the LoA value will be attached along with the chosen attribute(s).

8. The requested attributes from the SP are listed on the consent form. The user can always consult that list to determine if she needs to aggregate more attributes. If so, the steps 4-7 can be repeated.

9. Once the user has aggregated the required attributes, she selects those attributes. Based on her selection, the attributes from each IdP are wrapped inside a SAML *AttributeStatement* element. These are then are wrapped inside a SAML *AttributeStatements* element. A SAML assertion is then created with the AttributeStatements element which is embedded inside a SAML response and is sent back to the SP.

10. Upon receiving the response, the SP validates the assertion and retrieves the SAML AttributeStatements element. From there each attribute group is retrieved and from



Attributes from IdP:https://192.168.1.115/simplesaml/saml2/idp/metadata.php:

- [ ] username:ripul
- [ ] name:Ripul Test
- [ ] email:ripul@er.et
- [ ] telephone:01234445566
- [ ] age:34
- [ ] position:Student
- [ ] org:University of Glasgow
- [ ] salarygrade:6
- [x] loa:2
- [x] idp:https://192.168.1.115/simplesaml/saml2/idp/metadata.php

Attributes from IdP:www.facebook.com:

- [ ] facebook.id:100001509159639
- [ ] facebook.name:Rahim Kent
- [ ] facebook.first_name:Rahim
- [ ] facebook.last_name:Kent

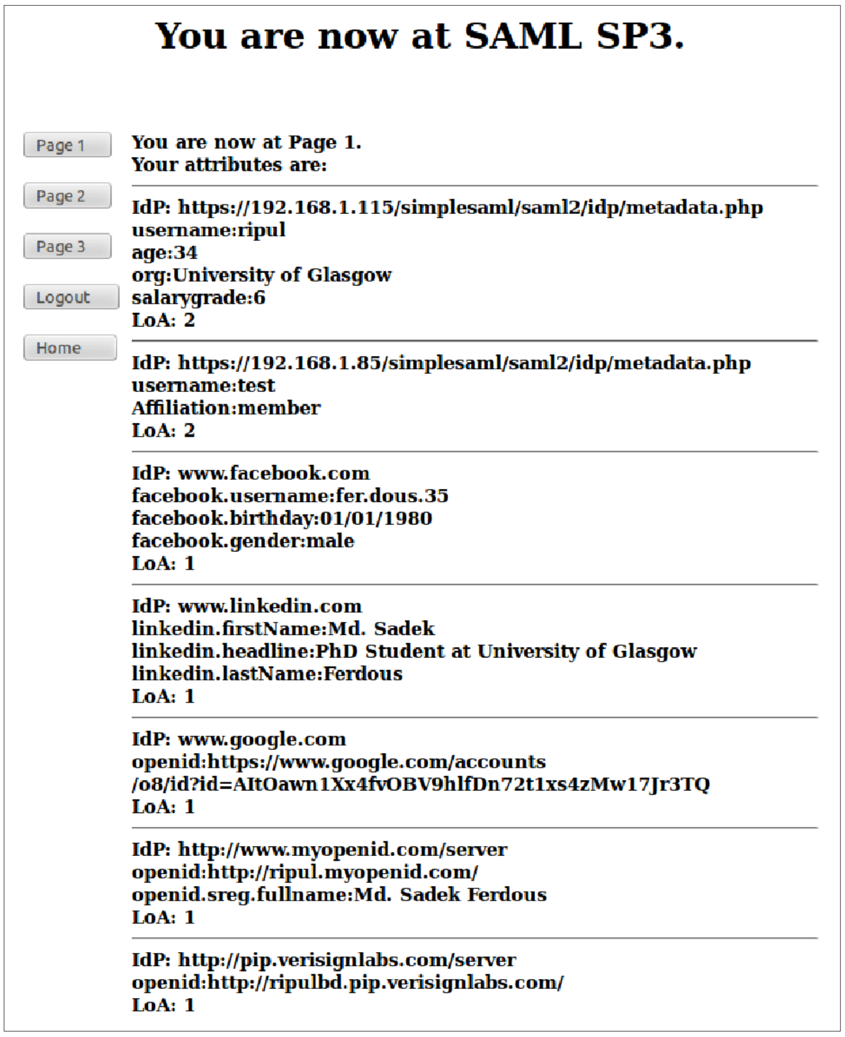
Figure 8.24: Aggregated attributes from two IdP at the consent form.

each group, all attributes are retrieved. A sample of such attributes is illustrated in Figure 8.25. The SP then can take authorisation decisions based on those attributes.

8.5.2 Use-case 2

The first use-case allows a user to aggregate attributes from multiple IdPs in a single session. However, the main drawback is the extreme level of trust on the Hybrid IdP as it obtains all the released attributes. Even if it is highly trusted, there is no guarantee that it will not abuse the attributes. One way to mask attributes released by a SAML IdP from the Hybrid IdP is to wrap them inside an encrypted assertion and present the encrypted assertion as an attribute. The second use-case illustrates this scenario. The setup is very similar to the first use-case except that the SAML IdP has to be federated with the SP to allow it to create an encrypted assertion that the SP can decrypt. It is also assumed that the user has already gone through the steps from 1 - 7 from the previous use-case. With this setup, the protocol flow for the use-case is illustrated in Figure 8.26 and is described below:

1. The user needs to aggregate attributes from a SAML IdP and she does not want to reveal the attributes to the Hybrid IdP. Therefore, the user selects the *Return Encrypted Assertion* checkbox and clicks the *Aggregate More Attributes* button (Figure 8.20).



You are now at SAML SP3.

You are now at Page 1.
Your attributes are:

IdP: https://192.168.1.115/simplesaml/saml2/idp/metadata.php
username:ripul
age:34
org:University of Glasgow
salarygrade:6
LoA: 2

IdP: https://192.168.1.85/simplesaml/saml2/idp/metadata.php
username:test
Affiliation:member
LoA: 2

IdP: www.facebook.com
facebook.username:fer.dous.35
facebook.birthday:01/01/1980
facebook.gender:male
LoA: 1

IdP: www.linkedin.com
linkedin.firstName:Md. Sadek
linkedin.headline:PhD Student at University of Glasgow
linkedin.lastName:Ferdous
LoA: 1

IdP: www.google.com
openid:https://www.google.com/accounts
/o8/id?id=AItOawn1Xx4fvOBV9hlfDn72t1xs4zMw17Jr3TQ
LoA: 1

IdP: http://www.myopenid.com/server
openid:http://ripul.myopenid.com/
openid.sreg.fullname:Md. Sadek Ferdous
LoA: 1

IdP: http://pip.verisignlabs.com/server
openid:http://ripulbd.pip.verisignlabs.com/
LoA: 1

Figure 8.25: Released attributes from multiple IdPs in a single SP session.

2. A session is created to keep track of aggregated attributes and then, the user is forwarded to the IdP selection page of the Hybrid IdP as discussed in the previous use-case. However, this time the page enlists only SAML IdPs. Since non-SAML IdPs cannot provide an encrypted assertion, they are filtered out. When the user chooses a SAML IdP, a SAML authentication request is created. An attribute called *EmbedAssertion* is inserted within the request containing the Entity ID of the SP. Then, the user is forwarded to the SAML IdP with the request.

3. The IdP notes the *EmbedAssertion* attribute in the request and the entity ID of the SP is stored in a session variable.

4. The user is authenticated at the IdP and the consent form containing attributes stored at this IdP is shown. The user chooses the attributes and clicks the *Release attributes to SP* button.

5. The IdP uses the entity ID from the session variable to create an encrypted assertion targeted for the SP using the chosen attributes. The encrypted assertion is then Base64

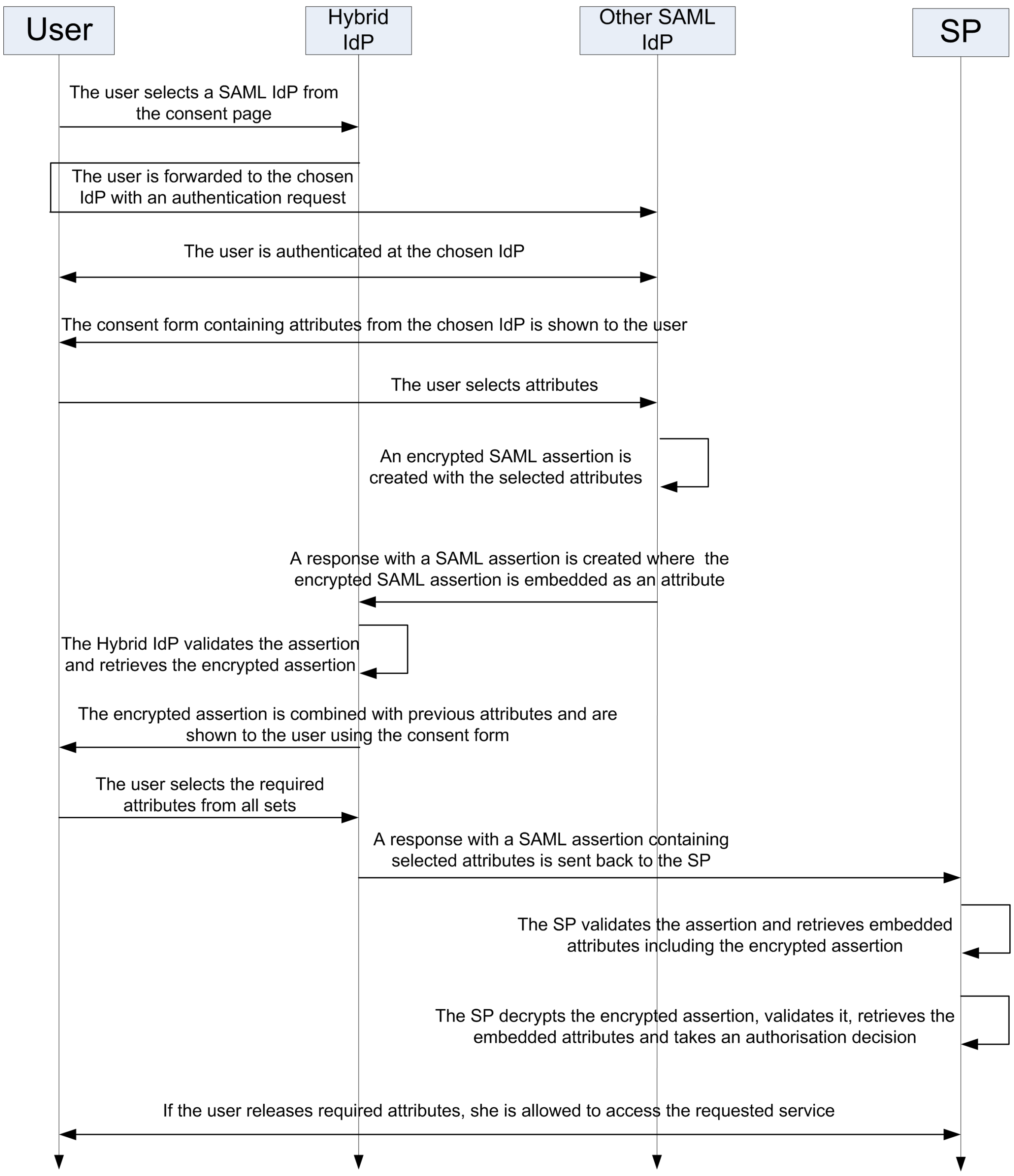

User
Hybrid IdP
Other SAML IdP
SP
The user selects a SAML IdP from the consent page
The user is forwarded to the chosen IdP with an authentication request
The user is authenticated at the chosen IdP
The consent form containing attributes from the chosen IdP is shown to the user
The user selects attributes
An encrypted SAML assertion is created with the selected attributes
A response with a SAML assertion is created where the encrypted SAML assertion is embedded as an attribute
The Hybrid IdP validates the assertion and retrieves the encrypted assertion
The encrypted assertion is combined with previous attributes and are shown to the user using the consent form
The user selects the required attributes from all sets
A response with a SAML assertion containing selected attributes is sent back to the SP
The SP validates the assertion and retrieves embedded attributes including the encrypted assertion
The SP decrypts the encrypted assertion, validates it, retrieves the embedded attributes and takes an authorisation decision
If the user releases required attributes, she is allowed to access the requested service

Figure 8.26: Protocol flow for Use-case 2.

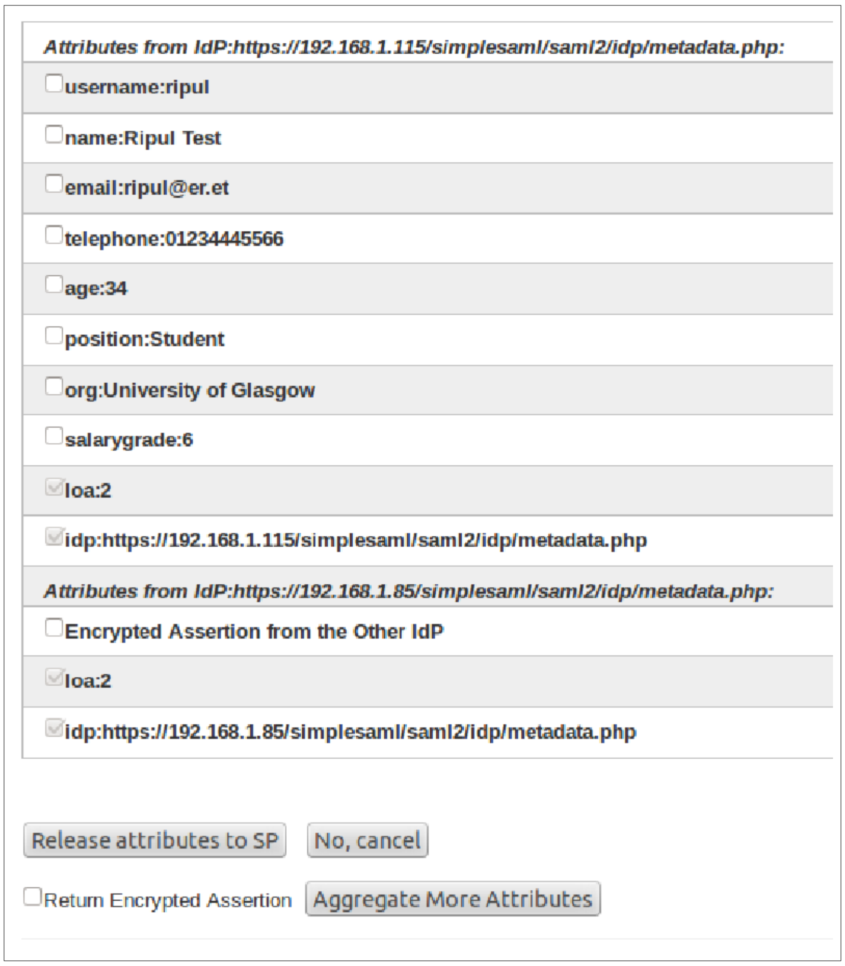
encoded and added as the value of the *encryptedAssertion* attribute and is used to create a regular unencrypted SAML assertion.

6. The regular assertion is then sent back to the Hybrid IdP.

7. Upon receiving this assertion, the Hybrid IdP validates the assertion and retrieves the *encryptedAssertion* attribute. Since this encrypted assertion is not decryptable by the Hybrid IdP, the IdP treats it as it is.

8. The Hybrid IdP merges the encrypted assertion with the previously aggregated at-

tributes using the approach described before and displays all of them in different attribute groups in the consent page. Since, the Base64-encoded encrypted assertion can be quite a large value, it is not shown in the consent page. Instead, the text *Encrypted Assertion from the Other IdP* is shown (Figure 8.27). Note that the Hybrid IdP assigns a LoA value of 2 for the encrypted assertion as the other SAML IdP is assumed to be trusted in this instance.


Attributes from IdP:https://192.168.1.115/simplesaml/saml2/idp/metadata.php:

☐ username:ripul
☐ name:Ripul Test
☐ email:ripul@er.et
☐ telephone:01234445566
☐ age:34
☐ position:Student
☐ org:University of Glasgow
☐ salarygrade:6
✓ loa:2
✓ idp:https://192.168.1.115/simplesaml/saml2/idp/metadata.php

Attributes from IdP:https://192.168.1.85/simplesaml/saml2/idp/metadata.php:

☐ Encrypted Assertion from the Other IdP
✓ loa:2
✓ idp:https://192.168.1.85/simplesaml/saml2/idp/metadata.php

Figure 8.27: Encrypted assertion as the attribute in the consent page.

9. When the user chooses the encrypted assertion (along with other attributes from the other IdP) and clicks the *Release attributes to SP* button, an assertion containing these attributes is created (as described previously) and is sent back to the SP.

10. The SP validates the assertion and retrieves attributes as discussed previously. However, when the SP finds the *encryptedAssertion* attribute, it is treated in a special way. The attribute is, at first, Base64-decoded. Then, the SP decrypts the assertion, validates it and retrieves the attributes. Finally, the SP can take an authorisation decision as described above.

8.5.3 Use-case 3

Even though the second use-case can tackle the problem of revealing attributes to the Hybrid IdP from a SAML IdP, it fails when attributes are released from non-SAML IdPs. However, the issue can be tackled effectively by combining the IR model with a PPIdP. By using a PPIdP, the user can aggregate attributes from non-SAML IdPs at the PPIdP, rather than at the Hybrid IdP. The aggregated attributes then can be released using an encrypted assertion targeted for the SP, just like the second use-case. This will allow a user to release attributes to the SP, however, such attributes will not be known to the Hybrid IdP and thus the mentioned problem can be tackled effectively. Before the use-case is initiated, it is assumed that a PPIdP has been federated with the SP as an IdP and has been added as an authentication source (an IdP) at the Hybrid IdP following the mechanism of dynamic federations. It is also assumed that the user has already gone through steps 1 - 7 from the first use-case. Moreover, as the current implementation of the PPIdP requires the user to access the SP in a mobile browser, the following interaction is assumed to take place from a browser of the mobile phone where the PPIdP is installed. With this setup, the protocol flow for the use-case is illustrated in Figure 8.28 and is described below:

1. The user needs to aggregate attributes from non-SAML IdPs and she does not want to reveal the attributes to the Hybrid IdP. Therefore, the user selects the *Return Encrypted Assertion* checkbox and clicks the *Aggregate More Attributes* button.

2. A session is created to keep track of the previous attributes and then the user is forwarded to the IdP selection page of the Hybrid IdP, as before. However, this time the page enlists only SAML IdPs. Since other non-SAML IdPs cannot fulfil the requirement, they are filtered out (Figure 8.29).

3. When the user chooses the PPIdP (tagged as *Mobile-IdP* in Figure 8.29), a SAML authentication request with the *EmbedAssertion* attribute is forwarded to the PPIdP.

4. The PPIdP notes the *EmbedAssertion* attribute in the request and the entity ID of the SP is retrieved and stored in a session variable.

5. The user is authenticated at the PPIdP and then the consent form containing attributes stored at this IdP are shown (Figure 8.30). The form has a *Aggregate Attributes* button which the user can click to initiate the attribute aggregation process.

6. Assuming the user clicks the *Aggregate Attributes* button, a new page with a list of non-SAML IdPs is shown to the user (Figure 8.31). The current implementation of the PPIdP allows the user to aggregate attributes from Twitter, Google, OpenID, Linkedin and Facebook.

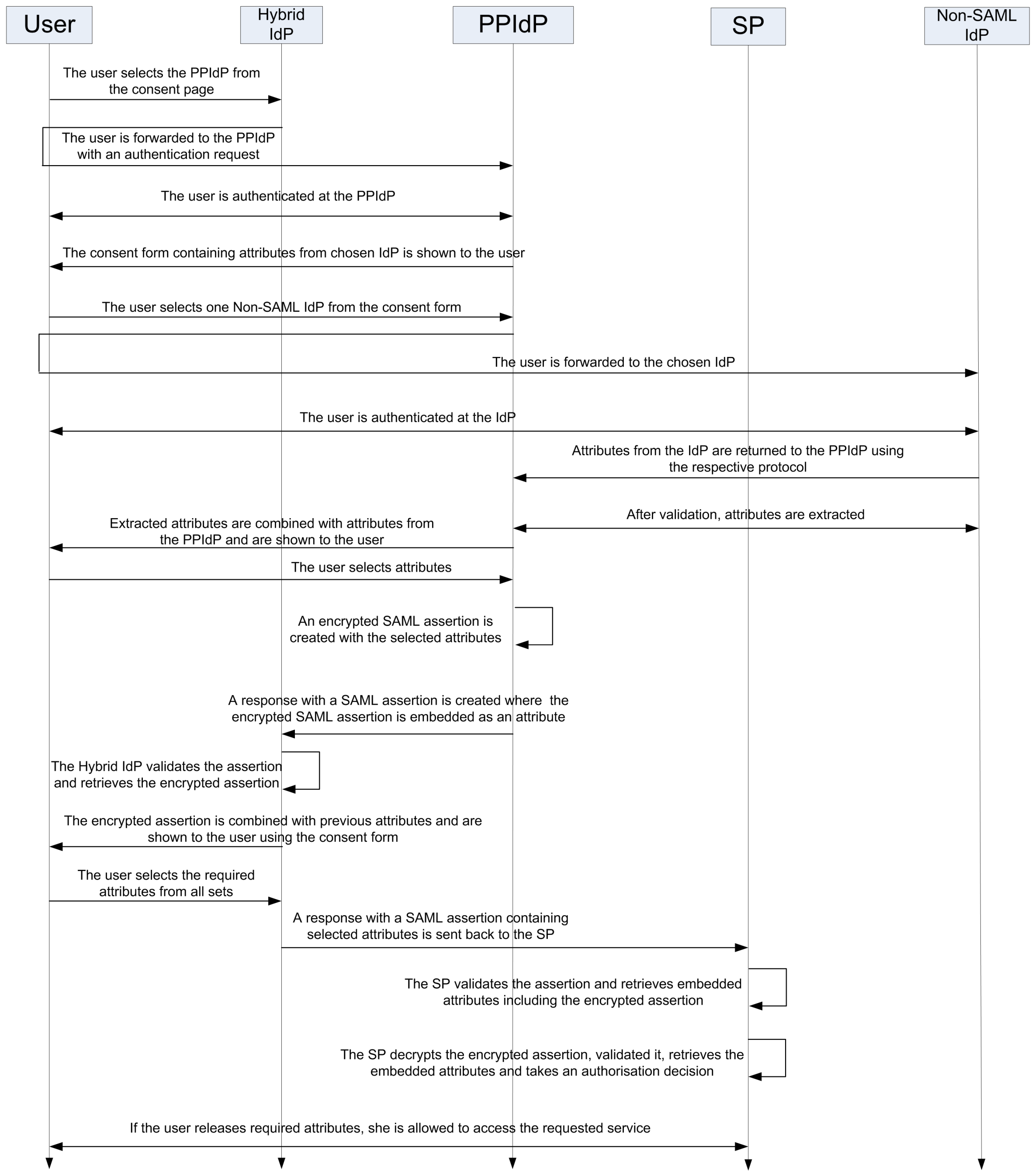
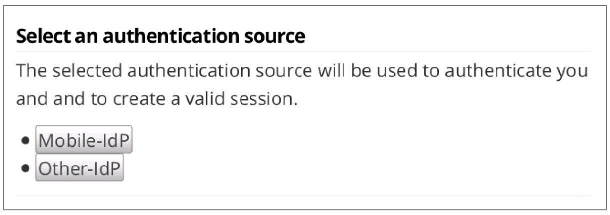

User
Hybrid IdP
PPIdP
SP
Non-SAML IdP
The user selects the PPIdP from the consent page
The user is forwarded to the PPIdP with an authentication request
The user is authenticated at the PPIdP
The consent form containing attributes from chosen IdP is shown to the user
The user selects one Non-SAML IdP from the consent form
The user is forwarded to the chosen IdP
The user is authenticated at the IdP
Attributes from the IdP are returned to the PPIdP using the respective protocol
After validation, attributes are extracted
Extracted attributes are combined with attributes from the PPIdP and are shown to the user
The user selects attributes
An encrypted SAML assertion is created with the selected attributes
A response with a SAML assertion is created where the encrypted SAML assertion is embedded as an attribute
The Hybrid IdP validates the assertion and retrieves the encrypted assertion
The encrypted assertion is combined with previous attributes and are shown to the user using the consent form
The user selects the required attributes from all sets
A response with a SAML assertion containing selected attributes is sent back to the SP
The SP validates the assertion and retrieves embedded attributes including the encrypted assertion
The SP decrypts the encrypted assertion, validated it, retrieves the embedded attributes and takes an authorisation decision
If the user releases required attributes, she is allowed to access the requested service

Figure 8.28: Protocol flow for Use-case 3.

7. The user chooses one of the IdPs and she is forwarded to the corresponding IdP where she is authenticated. Then, she comes back to the PPIdP with attributes as discussed in the first use-case.

8. The user can repeat steps 3-7 to aggregate more attributes from other IdPs. After aggregating attributes from multiple IdPs, the screenshot of the consent form at the PPIdP is given in Figure 8.32.

9. The user chooses the attributes that she wants to release and clicks the *Submit* button. The PPIdP uses the SP entity ID from the session variable to create an encrypted

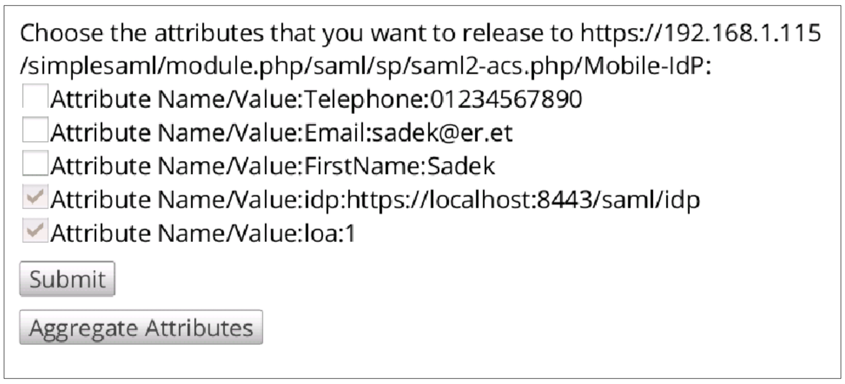


Select an authentication source

The selected authentication source will be used to authenticate you and and to create a valid session.

- Mobile-IdP
- Other-IdP

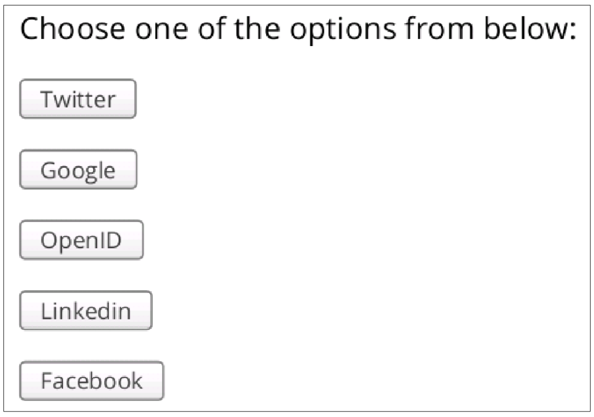
Figure 8.29: Filtered out non-SAML IdPs.



Choose the attributes that you want to release to https://192.168.1.115 /simplesaml/module.php/saml/sp/saml2-acs.php/Mobile-IdP:

☐ Attribute Name/Value:Telephone:01234567890
☐ Attribute Name/Value:Email:sadek@er.et
☐ Attribute Name/Value:FirstName:Sadek
✓ Attribute Name/Value:idp:https://localhost:8443/saml/idp
✓ Attribute Name/Value:loa:1

Figure 8.30: Consent form at the PPIdP.

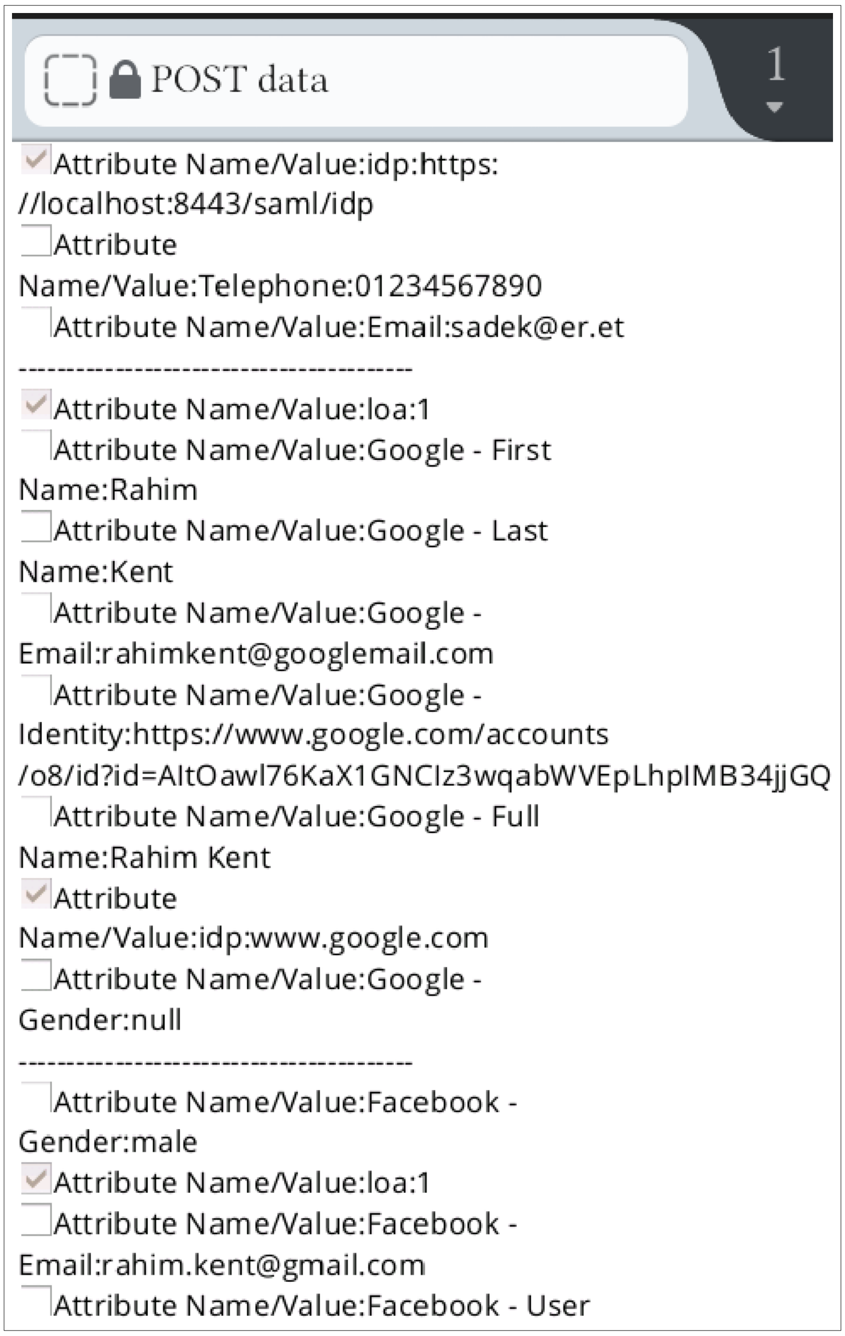


Choose one of the options from below:

Figure 8.31: Available IdP at the PPIdP.

assertion targeted for the SP using the chosen attributes. Then steps 8-10 of the second use-case are repeated and ultimately the SP receives all these attributes and takes an authorisation decision based on these attributes.

While preparing the encrypted assertion, the PPIdP groups the attributes based on their sources and inserts them in different AttributeStatement elements which are then wrapped inside an AttributeStatements element and inserted into the encrypted assertion as previously discussed. The PPIdP also inserts the name of the IdP in each group, however, it does not provide a LoA value. This is because the SP trusts a LoA value only provided by the Hybrid IdP and thus there is no advantage to provide a LoA value by the PPIdP. Instead, the Hybrid IdP assigns a LoA value of 1 for the encrypted assertion received from the PPIdP. This will enable the SP to assume all attributes contained in this particular encrypted assertion have a LoA value of 1. Note that this does not downgrade the level of assurance whatsoever for any


POST data

✓ Attribute Name/Value:idp:https://localhost:8443/saml/idp
☐ Attribute Name/Value:Telephone:01234567890
☐ Attribute Name/Value:Email:sadek@er.et

✓ Attribute Name/Value:loa:1
☐ Attribute Name/Value:Google - First Name:Rahim
☐ Attribute Name/Value:Google - Last Name:Kent
☐ Attribute Name/Value:Google - Email:rahimkent@googlemail.com
☐ Attribute Name/Value:Google - Identity:https://www.google.com/accounts/o8/id?id=AItOawl76KaX1GNCIz3wqabWVEpLhpIMB34jjGQ
☐ Attribute Name/Value:Google - Full Name:Rahim Kent
✓ Attribute Name/Value:idp:www.google.com
☐ Attribute Name/Value:Google - Gender:null

☐ Attribute Name/Value:Facebook - Gender:male
✓ Attribute Name/Value:loa:1
☐ Attribute Name/Value:Facebook - Email:rahim.kent@gmail.com
☐ Attribute Name/Value:Facebook - User

Figure 8.32: Aggregated attributes at the PPIdP.

IdP involved in this manner since the Hybrid IdP considers that attributes from the PPIdP and other non-SAML IdPs have a LoA value of 1. The advantage is that the Hybrid IdP has no way of knowing what attributes from other non-SAML attributes the user has consented to release to the SP.

8.6 Discussion

In this section, the proposed model is analysed against the required functional, security and privacy requirements (see Section 8.4.1). The advantages and limitations of the proposed model are also discussed.

8.6.1 Analysis

The three use-cases require that the Hybrid IdP and SP are part of the same federation as well as any other SAML IdPs and the PPIdP, thereby satisfying *F1*. The Hybrid IdP and the PPIdP also retain the sessions while the user is forwarded to aggregate attributes from other IdPs and performs the dual role of an IdP to the SP and an SP to other IdPs, thereby satisfying *F2* and *F3*. Based on the IP/IR mode, assertions are targeted for the appropriate party and thus enabling only the targeted entity to validate the assertion and retrieve embedded attributes. This satisfies *F4*. From the three use-cases, it is evident that the Hybrid model also satisfies *F5, F6* and *F7*.

The security requirement *S1* is mostly dependant on the respective IdP. The current implementation uses encrypted communication channels (HTTPS) during different protocol flows as per the specification of SAML, OpenID and OAuth. The encrypted channel guarantees the confidentiality, integrity and authenticity of aggregated attributes passed from different IdPs to a Hybrid IdP and from a Hybrid IdP to an SP. This satisfies *S2*. The Hybrid IdP itself does not store any released attributes and it determines and assigns the LoA in a correct manner and thus satisfies *S3* and *S4*.

The implementation of the Hybrid model is based on SAML and utilises the SimpleSAMLphp implementation and the PPIdP. Since both the SimpleSAMLphp implementation and the PPIdP satisfy privacy requirements *P1-P3*, these requirements are also met by the system. The Data Minimisation requirement is satisfied by allowing an SP to pass on the list of required attributes to the IdP and then by showing that list of user attributes at the *Consent* page.

As the Hybrid IdP and SP have been federated in the traditional way, they trust each other whereas all non-SAML IdPs and the PPIdP are considered untrusted to the Hybrid IdP and SP. The other SAML IdPs to which the authentication is delegated by the Hybrid IdP will be considered as trusted by the Hybrid IdP and following the rule of transitive trust property, those IdPs will also be considered trusted by the SP. The LoA value is used as a trust metric. Any attributes released by an untrusted entity, the Hybrid IdP will implicitly assign a LoA value of 1. On the other hand, for any trusted entity, a LoA value of 2-4 can be used depending on the authentication mechanism. In such cases, the Hybrid IdP may depend on the LoA value received by the respective IdP since the respective IdP knows which authentication mechanism has been used. For example, the other SAML IdP may release attributes with a LoA value of 2 when the respective user is authenticated using a username/password and a LoA value of 3 or 4 when the user is authenticated using a more secure mechanism such as OTP (One Time Password). The Hybrid IdP will just retrieve the embedded LoA value and assign it into the respective group of attributes. If no LoA value is present in the assertion by the other trusted IdP, the Hybrid IdP will provide a LoA value of 2 for that group of attributes.

8.6.2 Advantages

The Hybrid model offers a number of advantages which are described below:

1. By combining the features of the IP and IR model, it has been possible to leverage the advantages of both models allowing a user to aggregate attributes almost seamlessly.

2. The model allows a user to switch between the IP and IR mode in the most convenient way - just by selecting a checkbox. Then, all the mechanisms are handled internally without burdening the user.

3. This is the first model to allow a user to aggregate attributes from non-SAML IdPs in a SAML setting. With the proliferation of social networks, lots of user attributes are stored in such IdPs. Allowing users to aggregate attributes from such IdPs in the FIM setting is novel yet timely.

4. Another added advantage that comes from the SSO feature of SAML is that once the Hybrid IdP releases the attributes to an SP, a user does not need to log in and choose attributes for the same SP until the session is lost. However, when the user accesses services from another SP and chooses the Hybrid IdP, the SSO feature directly takes the user to the consent form containing the aggregated attributes from the first instance with the first SP. At this point, the user can aggregate more attributes if she wishes or chooses a separate set of attributes. In this way, a user can release completely different sets of attributes to different SPs using either the same or extended sets of aggregated attributes.

5. When a user logs in to a non-SAML IdP for the first time, she can take advantage of the cookie based session maintenance feature of non-SAML IdPs. She will just need to select the appropriate OAuth IdP or select the OpenID option and write down the name of the OpenID provider; the user will retrieve attributes automatically without any further login. This allows the user to achieve a form of SSO at these IdPs.

6. As discussed above, OAuth and most OpenID providers do not support the selective disclosure of attributes. A user needs to release either the full basic or extended set of attributes to an SP. In this regard, the model provides another layer for the user to choose individual attribute from any OAuth and OpenID provider either using the Hybrid IdP or PPIdP. As there is no way for the Hybrid or a PPIdP to know which attributes it should ask the OpenID provider to release, the best option is to ask for an extended set of attributes from the OpenID/OAuth provider and to allow the user to choose from that extended list.

7. None of the current SAML implementations allows an SP to pass on information regarding requested attributes along with the authentication request. The adopted approach to pass such request via the SAML authentication request is the first of its kind and could be widely adopted not only for the attribute aggregation mechanism but also for any general SAML implementation.

8.6.3 Limitations

There are still a few limitations to the approach, many of which are due to the way the Hybrid IdP and the other non-SAML IdPs interact with each other. The limitations are discussed below:

1. One major problem that still exists in almost every SAML system is the lack of control over the attributes once they are released to the SP. Being mostly based on SAML, the architecture also suffers from this limitation.

2. Instead of requesting an extended set of attributes from an OpenID provider, the implementation could provide an option, preferably at a page where the user provides her OpenID, which would allow the user to choose the attributes that she wants to request from an OpenID provider. However, it should be noted that it does not guarantee that only the requested attributes will be returned, as evident from the previous discussions or even the OpenID Provider has the requested attributes.

3. The LoAs of different IdPs are currently hard-coded. The way LoAs have been assigned might not suit every scenario. The best approach would be to allow an administrator to assign the LoA value in the configuration file and then read them during run-time.

4. There is a problem when a PPIdP is used for aggregating attributes: a PPIdP is only accessible from a browser of the mobile phone where it is installed. This is not a realistic assumption for wide-scale adaptation of the approach.

8.7 Conclusion

With the proliferation of online services and social networks, more and more user attributes will be stored online across multiple IdPs. In the future, innovative service scenarios would definitely need a mechanism to allow users to aggregate attributes from multiple sources in the simplest way. The existing models are complex and require preliminary steps which are not intuitive. In this chapter, a hybrid model is presented to allow a user to aggregate attributes from SAML and non-SAML IdPs in a single service session. The approach, based on the Identity Proxying and Relay model, is simple and does not require a preliminary step or complex user interactions like previous systems. Depending on the requirement, the user can switch back and forth between the proxy and relay model by selecting or deselecting a single checkbox. Unlike previous models, it allows the proxy and relay IdP to retain the source of the aggregated attributes which would be beneficial for an SP to take any authorisation decision.

A few additional modifications of SAML have been proposed. In addition, the lacking in the current SAML specification prohibits an SP to inform an IdP regarding its attribute requirements while authenticating a user. The implementation illustrates how it can be rectified. It has also been shown how the privacy of the user can hugely be improved by leveraging the IR model and using a PPIdP. This essentially gives a user the ultimate control over her data as no intermediate IdP knows the data released to an SP. The proposed mechanism not only can be used to introduce novel service access scenarios but can also be used to improve the privacy of users which has been not possible in any existing technologies. Hence, it is strongly believed that the proposed approach has true potential to advance federated service scenarios to the next level and to begin a new era of next generation federated services.

Chapter 9

Conclusions

This thesis has investigated the need for a novel Identity Management System since existing systems do not allow users to have sufficient control over their data and have limitations which can drastically affect the privacy of users. To counteract this problem, this thesis has proposed a novel Identity Management System, called *User-controlled Identity Management System*, utilising a novel type of user-controlled IdP called *Portable Personal Identity Provider* (PPIdP). Moreover, this thesis has showcased the additional usefulness of a PPIdP by illustrating how its capabilities to retrieve data from different sensors can be harnessed to introduce innovative service access scenarios and how it can be used for privacy-preserving attribute aggregation from heterogeneous IdPs. The applicability of the work has been demonstrated through a number of several use-cases. In addition, different security, privacy and trust issues have been analysed throughout the thesis.

9.1 Contributions

The key contributions of this thesis are the following.

Mathematical Model of Identity and Identity Management. The thesis has introduced a novel mathematical model of Identity and Identity Management covering a wide range of related vocabulary. This is the first model of this kind and is aimed to tackle the issue of inconsistency in different central topics of Identity Management. The definition of each topic in the model has been used as the point of reference throughout the thesis. In addition, the model has been used to analyse three issues related to Identity Management: the effect of multiple partial identities, data ownership and Identity Theft. Furthermore, the applicability of the model has been illustrated by utilising it to characterise the behaviour of an Identity Management System and three popular IdM models mathematically.

Dynamic Identity Federations. The thesis has proposed a simple mechanism for managing a dynamic federation in a fully automatic fashion using SAML. The crucial benefit of the proposed approach is that it can be easily adopted by any SAML implementation and requires no modification of the SAML Protocol. A proof of concept has been implemented to show the applicability of the proposed approach by illustrating several use-cases. The proposed approach has been used extensively throughout the thesis to implement different prototypes.

User-controlled Identity Management Systems. The thesis advocates the need for a more privacy-friendly IMS and puts forward the proposal of a User-controlled Identity Management System to allow users to have better control over their data. To realise such a system, a novel type of user-controlled IdP called Portable Personal Identity Provider has been proposed and implemented using the mechanism of dynamic federations. The applicability of such a system has been demonstrated through few use-cases. How a PPIdP can be used to rectify several problems of existing Identity Management Systems have been analysed. Furthermore, different functional, security, privacy and trust requirements have been analysed.

Context-aware Federated Services using PPIdP. The thesis demonstrates how a PPIdP can be used to provide context-aware federated services. This is the first approach to combine the two popular trends: context-aware services and federated services. The proposal illustrates how the additional capabilities of a PPIdP can be leveraged to provide innovative context-aware federated services. As above, the applicability of the system has been outlined using several use-cases and different functional, privacy, security and trust issues have been analysed in details.

A Hybrid Architecture for Attribute Aggregation. Finally, the thesis presents a hybrid architecture for aggregating attributes from heterogeneous identity providers. This is the first architecture for aggregating attributes from SAML as well as non-SAML IdPs in a SAML federation. The architecture combines the advantages of the IP and IR models. This thesis has also examined how the privacy of a user can be improved during attribute aggregation by federating a PPIdP with the proposed architecture. A number of use-cases are considered and different functional, security, privacy and trust issues have been examined. In addition, it has been analysed how different design choices have addressed these issues.

9.2 Conclusions

This section outlines the major conclusions drawn from the research conducted in this thesis. In particular, this section revisits the research objectives outlined in Chapter 1 and analyses whether these objectives have been fulfilled.

The number of partial identities that any user needs to manage is growing with the increasing number of online services introduced by different service providers. These identities are scattered across multiple providers making them increasingly difficult to manage. In addition, providers do not give users enough control over their stored data. The concepts of Identity Management and different practical Identity Management Systems have been introduced to tackle these issues. Unfortunately, the investigation in this thesis demonstrates the following shortcomings.

- The very concepts of Identity and Identity Management are defined in different ways and there lacks a consistency across these different notions and definitions of many central aspects of Identity and Identity Management.
- The existing IMSs do little to help reduce the number of partial identities that users have in different IdPs. The main reason for this is that users may not trust a single third party IdP to hold all of their attributes. The after effect of this is that users end up with different partial identities with different types of attributes in different IdPs.
- Since partial identities are scattered across different IdPs, there is not a central place from where a user can manage her partial identities.
- IdPs do not empower users with enough control so that they do not know how their attributes are being handled in the respective IdP.
- The existing IMSs do not satisfy certain privacy requirements which can drastically affect the privacy of a user.
- The existing IMSs do not take advantage of many novel capabilities of smart mobile devices to offer innovative services.
- Finally, the existing IMSs do not offer any effective mechanism to aggregate attributes from different types of IdPs considering different privacy issues.

In this thesis, the aim is to address all these issues. Next, it is analysed how these issues have been tackled and in doing so how the research objectives have been fulfilled.

In Chapter 3, a consistent vocabulary on Identity and Identity Management has been developed. The vocabulary has been used as a point of reference throughout the thesis. Then, a mathematical framework has been developed to formally model the central aspects of IdM. Furthermore, it has been shown how this framework can be used to model several aspects of an IMS. This is the first model of its kind and it has the potential for rigorous reasoning and formal analysis of IdM. In addition, four application scenarios are discussed illustrating how the model can be used for analysis and to mathematically characterise popular IdM Models. This fulfills research objectives RO1 and RO2.

In Chapter 4, a taxonomy of requirements has been built up and these requirements have been used to compare three popular IMSs side by side. The analysis has identified the strengths and weaknesses of each system. Based on the analysis, SAML has been chosen for subsequent research. Thus, this chapter fulfils research objective RO3.

In Chapter 5 and 6, a proposal of a User-controlled Identity Management System has been presented. It has been shown how such a system based on this concept can be developed. The PPIdP takes the central stage in realising such a system. It has been discussed in detail how a PPIdP has been developed. It has also been analysed how this approach can be used to tackle the stated shortcomings of existing IMSs. In doing so, this chapter addresses research objective RO4.

In Chapter 7, a framework for Context-aware Federated Services and in Chapter 8 a hybrid architecture for attribute aggregation from multiple providers have been presented. Both these chapters have showcased the additional advantages of a PPIdP. In such, these two chapters fulfil research objectives RO5 and RO6 respectively.

Also, the PPIdP has been developed in such a way so that it satisfies almost all requirements presented in the taxonomy of Chapter 4. For functional requirements, the identifier is represented using a username in a PPIdP. The context in a PPIdP is not only represented using application domains, but also using locations and attributes of a user. It provides both logging and history management functionalities and supports an attribute aggregation mechanism from multiple providers.

SAML satisfies all security requirements of the taxonomy (Authentication, Confidentiality, Integrity and Non-repudiation) as discussed in Chapter 4. Being based on SAML, the PPIdP also satisfies all these security requirements. As for the privacy requirements, the PPIdP releases pseudonymous identifiers for each SP, like SAML. One of the ways data minimisation can be achieved is to inform users what attributes are required for accessing any service. The implemented mechanism illustrated in Chapter 8 enables any SAML implementation to acknowledge a user regarding the requirements of attributes for accessing a service in the consent module. This feature has been integrated with the implemented PPIdP as well. The PPIdP and the SAML implementation used in this research have been equipped with the functionality that allows a user to disclose attributes selectively and with explicit consent. The only requirement that is not fulfilled in a PPIdP, like in SAML, is the support for anonymous identifiers. This is just to be consistent with SAML. Next, Tables 4.1 and 4.2 are revisited for comparing the proposed system with SAML, OpenID and OAuth. The result of this comparison is outlined in Tables 9.1 and 9.2.

By fulfilling all research objectives, this research also validates the statement of the thesis. It has been shown that smart mobile devices can be used to realise a novel User-controlled Identity Management System which can tackle the privacy problems occurring in the existing IMSs. More precisely, it offers users the ultimate control over their data and provides additional functionalities that can be utilised to offer novel online services.

Table 9.1: Functional and Security Requirements.

	Functional					Security				
	I-R	CD	LOG	HM	AA	AUTH	STO	CON	INT	NR
SAML	✓ (IdP Specific)	✓	✗	✗	✗	✓	✓	✓	✓	✓
OpenID	✓ (URI/XRI)	✓	✗	✗	✗	✓	✓	✓	✓	✓
OAuth	✓ (IdP Specific)	✓	✗	✗	✗	✓	✓	✓	✓	✓
PPIdP	✓ (IdP Specific)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 9.2: Privacy and Trust Requirements.

	Privacy					TRUST
	SAP		DM	UE		
	ANON	PSD		SD	EC	
SAML	✗	✓	✓	IdP Specific	IdP Specific	Federated
OpenID	✗	✗	✓	Provider Specific	Provider Specific	Open
OAuth	✓	✗	✗	✗	Provider Specific	Open
PPIdP	✗	✓	✓	✓	✓	Federated

A User-controlled Identity Management System using a PPIdP is all about empowering users to have ultimate control over their data. Since all attributes are stored in a PPIdP which is under the full control of a user, she can trust it to store all their attributes. Storing all attributes in a central place also makes them easier to manage. In addition, a PPIdP can be used to replace all IdPs except any highly trusted IdPs. This will reduce the number of partial identities a user needs to manage in the replaced IdPs. A PPIdP also enables a user to release only those attributes that she wants to release to an SP and has many additional advantages as demonstrated in this thesis. With all these advantages that the PPIdP offers, it can be regarded as the next evolution of User-centric Identity Management where users are empowered with the sufficient control to introduce a new era of privacy-friendly user-controlled service provisioning.

9.3 Future Work

In this thesis, a novel approach for managing identities has been investigated and implemented. Being a novel approach, the possible new directions for research inspired by or stemming from this thesis are broad. In this section, a few of such possible directions for future research are highlighted.

The mathematical model of Identity and Identity Management presented in Chapter 3 does not deal with privacy properties. This is intentional as there are other works, e.g. [67, 68], that have formalised the privacy properties of an IMS. It will be interesting to investigate how the proposed model can be integrated with these existing works. Currently, the model is static in nature and is not able to capture the dynamics of a practical IMS. Such dynamics of practical systems can be captured by extending the model using the state-based Z language [153] or the event-based Process Algebra CSP [154]. This will not only help in providing a rigorous understanding of these practical systems, but also to formally compare the behaviour of systems. For example, if the dynamics are captured using CSP, then the tool FDR3 [155] can be used for automated verification (e.g. using refinement checking to compare specifications of IMSs with their implementations). Furthermore, the model lays down a solid foundation that can be used as the framework for an *Identity Calculus* which ultimately can be used to study the change of identities or partial identities that users go through from systems to systems and its effect in those systems. The topic of Internet of Things (IoT) is one of the hotly anticipated technologies that has been envisioned to change the way users and objects inter-connect and interact with each other. One of the major challenges in IoT is to manage the identities of different users and objects (things) in a dynamic fashion. The proposed Identity Calculus can be an effective tool to study the notion of identity management in IoT.

One of the weaknesses of the current implementation is that a PPIdP allows a user access only from a browser of the mobile device on which it is installed. This imposes a restriction for any user who wants to access the service of the PPIdP using a browser in a PC or a laptop. The implementation of PPIdP needs to be extended with additional capabilities so that it is visible to any device within a local network. All smart mobile devices can be connected to a PC or a laptop using a cable or with the equipped bluetooth functionality. Even if a device and a PPIdP are not visible to each other inside a network, this facility can be used to connect a mobile device, where the PPIdP is installed, with another device. In this manner, the service of the PPIdP can be availed from a browser of any device. With this functionality, the Personal Attribute Store (PAS), can be used to store additional information (e.g. contact book and identifier/credential for different online services along with other user attributes and their values) and make it available to any browser in any device.

In this thesis, it has been mainly focused on the investigation of existing systems to identify their limitations as well as on the design and development of a new system to overcome these limitations. Hence, the usability study of the proposed system has not been considered. The wide-scale adoption of any new system will, in part, depend on its usability. To reach a wider audience, it is therefore necessary to conduct a thorough usability study of the proposed system. The challenge here is the lack of any existing work in this field within the scope of Identity Management Systems. In addition, there is no guideline on how such a study can be conducted and no evaluation criteria on how such a system can be evaluated. This opens up the possibility of a new research area on how to conduct a usability study within the scope of Identity Management Systems and on what criteria should be used for evaluation.

Based on the guidelines and the criteria, the proposed system can be evaluated. In addition, an empirical comparative analysis between the proposed system and existing systems can be carried out to determine its usability against theirs.

The term *Personal Identity Federation (PIF)* has been used to denote a federation with a PPIdP. The concept of PIF can be generalised with the concept of a (private) federation, with or without a PPIdP, which is only valid within a private network. For example, such a private federation will be useful to create a network of IoT within a home and the federation will be the baseline to establish trust among different entities in the network. The very dynamic nature of how different entities will be added and removed from such a federation underlines that the federation must be managed in a dynamic fashion. The approach for dynamic federation with some required modification will be an ideal choice for managing such a federation.

As mentioned in Chapter 8, one problem that exists in almost every SAML system is the lack of control over the attributes once they are released to the SP. One way to achieve control in this regard could be the use of remote administration of policies between two entities. None of the SAML implementation currently supports this mechanism. It will be interesting to investigate how such mechanisms can be integrated within SAML.

The concept of trust plays a central role in Federated Identity Management as evident from this thesis. The trust requirements of traditional FIM evolve and novel trust requirements emerge as new types of federations (e.g. dynamic federations) are proposed or novel systems based on FIM (e.g. CAFS, Hybrid Attribute Aggregation architecture) are introduced. Such novel trust requirements have been introduced in textual formats throughout the thesis. The limitation of describing such trust requirements in textual formats is that they are difficult to analyse and compare against each other. A mathematical model will not only tackle this problem, but also will enable formalism within this scope. There have been numerous works on the mathematical representation, modelling and analysis of trust issues in online services. Surprisingly, the mathematical representation, modelling and analysis of different trust requirements in FIM have received very little attention so far. It is essential to address this issue by developing a mathematical model of trust issues in FIM. Such formal modelling can help to represent complex trust issues in a convenient way and can be used to analyse and calculate trust among different entities qualitatively as well as quantitatively.

References

[1] K. Cameron, “The Laws of Identity,” 14 May, 2005. [Online]. Available: http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf.

[2] mobiThinking, “Global mobile statistics 2014,” May, 2014. [Online]. Available: http://goo.gl/dgh5m1.

[3] Cisco, “Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014-2019,” 3 February, 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[4] A. Jøsang and S. Pope, “User Centric Identity Management,” in *Asia Pacific Information Technology Security Conference, Australia*, p. 77-89, 2005.

[5] OASIS Standard, “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0,” 15 March, 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[6] OpenID, “OpenID Authentication 2.0 - Final,” 5 December, 2007. [Online]. Available: http://openid.net/specs/openid-authentication-2_0.html.

[7] D. H. (Editors), “The OAuth 2.0 Authorization Framework: RFC 6749,” October, 2012. [Online]. Available: http://tools.ietf.org/html/rfc6749.

[8] Wikipedia, “Security Assertion Markup Language,” Accessed on 12 September, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Security_Assertion_Markup_Language.

[9] A. Jøsang, J. Fabre, B. Hay, J. Dalziel and S. Pope, “Trust requirements in identity management,” in *Proceedings of the 2005 Australasian workshop on Grid computing and e-research - Volume 44*, ser. ACSW Frontiers ’05.Darlinghurst, Australia, p. 99-108, 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1082290.1082305

[10] Shibboleth Project, “Shibboleth Architecture: Protocols and Profiles,” 10 September, 2005. [Online]. Available: http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-arch-protocols-latest.pdf.

[11] UNINETT, “SimpleSAMLphp”. [Online]. Available: http://simplesamlphp.org/.

[12] Wikipedia, “OpenID,” Accessed on 2 September, 2011. [Online]. Available: http://en.wikipedia.org/wiki/OpenID.

[13] B. Kissel, “OpenID 2009 Year in Review,” 16 December, 2009. [Online]. Available: http://openid.net/2009/12/16/openid-2009-year-in-review/.

[14] OpenID Wiki, “List of OpenID Libraries,” Accessed on 29 August, 2011. [Online]. Available: http://wiki.openid.net/w/page/12995176/Libraries.

[15] OAuth, “OAuth Community Site”. [Online]. Available: http://oauth.net/.

[16] E. Hammer-Lahav, “OAuth 1.0 guide,” Accessed on 2 April, 2012. [Online]. Available: http://hueniverse.com/oauth/.

[17] E. Hammer-Lahav (Editor), “The OAuth 1.0 Protocol - RFC,” April, 2010. [Online]. Available: http://tools.ietf.org/html/rfc5849.

[18] OAuth, “Introducing OAuth 2.0,” 15 May, 2010. [Online]. Available: http://hueniverse.com/2010/05/introducing-oauth-2-0/.

[19] M. S. Bargh, B. Hulsebosch and H. Zandbelt, “Scalability of trust and metadata exchange across federations,” December, 2010. [Online]. Available: https://tnc2011.terena.org/getfile/693.

[20] P. Harding, L. Johansson and N. Klingenstein, “Dynamic Security Assertion Markup Language: Simplifying Single Sign-On,” *Security Privacy, IEEE*, vol. 6, no. 2, p. 83-85, March, 2008.

[21] OASIS. Standard, “A profile for distributed SAML metadata management,” 22 October, 2007. [Online]. Available: https://spaces.internet2.edu/display/dsaml/A+profile+for+distributed+SAML+metadata+management.

[22] OASIS Security Services TC, “SAML V2.0 Metadata Interoperability Profile, Working Draft 01,” 1 August, 2008. [Online]. Available: https://spaces.internet2.edu/download/attachments/11275/draft-sstc-metadata-iop-01.pdf?version=2&modificationDate=1217876016355.

[23] “UNINETT”. [Online]. Available: https://www.uninett.no/.

[24] A. Solberg, “Dynamic SAML,” 18 February, 2010. [Online]. Available: https://rnd.feide.no/2010/02/18/dynamic_saml/.

[25] P. A. Cabarcos, F. A. Mendoza, A. Marín-López and D. Díaz-Sánchez, “Enabling SAML for Dynamic Identity Federation Management,” in *Wireless and Mobile Networking*, Springer Boston, vol. 308, p. 173-184, 2009.

[26] Y. Zuo, X. Luo and F. Zeng, “Towards a Dynamic Federation Framework Based on SAML and Automated Trust Negotiation,” in *Web Information Systems and Mining*, ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 6318, p. 254-262, 2010.

[27] Y. Xiang, J. A. Kennedy, M. Egger, H. Richter, “Network and Trust Model for Dynamic Federation,” in *Proceedings of the Fourth International Conference on Advanced Engineering Computing and Applications in Sciences*, p. 1-6, 2010.

[28] M. C. Mont, B. Balacheff, J. Rouault, D. Drozdzewski,“On Identity-Aware Devices: Putting Users in Control across Federated Services,” HP Laboratories Bristol, Tech. Rep. HPL-2008-26, March, 2008. [Online]. Available: http://www.hpl.hp.com/techreports/2008/HPL-2008-26.pdf

[29] T. Abe, H. Itoh and K. Takahashi, “Implementing identity provider on mobile phone,” in *Proceedings of the 2007 ACM workshop on Digital identity management*, ser. DIM ’07, p. 46-52, 2007.

[30] Liberty Alliance Project, “Liberty ID-FF Architecture Overview Version: 1.2-errata-v1.0,” Accessed on 30 June, 2011. [Online]. Available: http://projectliberty.org/liberty/content/download/318/2366/file/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf.

[31] C. Staite, “Portable secure identity management for software engineering,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ser. ICSE ’10. New York, USA, ACM, p. 325-326, 2010.

[32] “Android”. [Online]. Available: https://www.android.com/.

[33] “iOS”. [Online]. Available: https://www.apple.com/uk/ios.

[34] “Windows”. [Online]. Available: http://windows.microsoft.com/.

[35] “Google MyAccount”. [Online]. Available: https://myaccount.google.com/.

[36] “AppleID”. [Online]. Available: https://appleid.apple.com/.

[37] “Windows Live ID”. [Online]. Available: https://www.live.com/.

[38] "Federated Login for Google Account Users". [Online]. Available: https://developers.google.com/accounts/docs/OpenID.

[39] J. E. Bardram, R. E. Kjær and M. Ø. Pedersen, "Context-aware user authentication–supporting proximity-based login in pervasive computing," in *UbiComp 2003: Ubiquitous Computing*, p. 107–123, 2003.

[40] W. A. Jansen, S. I. Gavrila and V. Korolev, "Proximity-Based Authentication for Mobile Devices," in *Security and Management*, p. 398-404, 2005.

[41] B. Malek, A. Miri and A. Karmouch, "A framework for context-aware authentication," in *IET 4th International Conference on Intelligent Environments*, p. 1-8, 2008.

[42] W. Hsieh and J. Leu, "Design of a time and location based One-Time Password authentication scheme," in *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference*, ser. IWCMC '11, p. 201-206, 2011.

[43] E. Hayashi, S. Das, S. Amini, J. Hong and I. Oakley, "Casa: context-aware scalable authentication," in *Proceedings of the Ninth Symposium on Usable Privacy and Security*, ser. SOUPS '13, p. 3:1–3:10, 2013.

[44] M. Moyer and M. Ahamad, "Generalized role-based access control," in *Proceedings of the 21st International Conference on Distributed Computing Systems*, p. 391-398, 2001.

[45] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, p. 38-47, 1996.

[46] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev and M. Ahamad and G. Abowd, "Securing context-aware applications using environment roles," in *Proceedings of the sixth ACM symposium on Access control models and technologies*, p. 10–20, 2001.

[47] G. Zhang and M. Parashar, "Context-aware dynamic access control for pervasive applications," in *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, p. 21-30, 2004.

[48] R. J. Hulsebosch, A. H. Salden, M. S. Bargh, P. W. G. Ebben and J. Reitsma, "Context sensitive access control," in *Proceedings of the 10th ACM symposium on Access control models and technologies*, ser. SACMAT '05. New York, USA: ACM, p. 111-119, 2005.

[49] J.Y. Tigli, S. Lavirotte, G. Rey, V. Hourdin and M. Riveill, "Context-aware Authorization in Highly Dynamic Environments," *CoRR*, vol. abs/1102.5194, 2011.

[50] OASIS, “A Brief Introduction to XACML,” 14 March, 2003. [Online]. Available: http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html.

[51] M. Covington, P. Fogla, Z. Zhan and M. Ahamad, “A context-aware security architecture for emerging applications,” in *Proceedings of the 18th Annual Computer Security Applications Conference*, p. 249-258, 2002.

[52] J. Hu and A.C. Weaver, “A dynamic, context-aware security infrastructure for distributed healthcare applications,” in *the Proceedings of the first workshop on pervasive privacy security, privacy, and trust*, 2004.

[53] W3C, “Web Services Policy 1.2 - Framework (WS-Policy),” 25 April, 2006. [Online]. Available: http://www.w3.org/Submission/WS-Policy/.

[54] K. Nishiki and E. Tanaka, “Authentication and Access Control Agent Framework for Context-Aware Services,” in *the 2005 Symposium on Applications and the Internet Workshops*, p. 200-203, 2005.

[55] M. J. Covington, M. R. Sastry and D. J. Manohar, “Attribute-based authentication model for dynamic mobile environments,” in *Security in Pervasive Computing*, p. 227-242, 2006.

[56] D. Goel, E. Kher, S. Joag, V. Mujumdar, M. Griss and A. Dey, “Context-Aware Authentication Framework,” in *Mobile Computing, Applications, and Services*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer Berlin Heidelberg, vol. 35, p. 26-41, 2010.

[57] A. S. Q. Arabo and M. Merabti, “A Framework for User-Centred and Context-Aware Identity Management in Mobile Ad Hoc Networks (UCIM) ,” *Ubiquitous Computing and Communication Journal*, 9 June, 2009.

[58] N. Klingenstein, “Attribute Aggregation and Federated Identity,” in *International Symposium on Applications and the Internet Workshops*, p. 26-29, 2007.

[59] B. Hulsebosch, M. Wegdam, B. Zoetekouw, N. V. Dijk and R. P. Wijnen, “Virtual collaboration attribute management,” 2011. [Online]. Available: https://www.surf.nl/binaries/content/assets/surf/en/knowledgebase/2012/EDS+11-06+Attribute+Management+v1.0.pdf.

[60] G. Inman, D. W. Chadwick and N. Klingenstein, “Authorisation using attribute from multiple authorities-a study of requirements,” in *Proceedings of HCSIT Summit-ePortfolio International Conference*, p. 15-19, 2007.

[61] D. Chadwick and G. Inman, "Attribute aggregation in federated identity management," *Computer*, vol. 42, no. 5, p. 33-40, 2009.

[62] U. Glässer and M. Vajihollahi, "Identity management architecture," in *IEEE International Conference on Intelligence and Security Informatics*, p. 137-144, 2008.

[63] Modinis, "Common Terminological Framework for Interoperable Electronic Identity Management," Accessed on 28 June, 2011. [Online]. Available: https://www.cosic.esat.kuleuven.be/modinis-idm/twiki/bin/view.cgi/Main/GlossaryDoc.

[64] J. Camp, "Digital identity," *Technology and Society Magazine, IEEE*, vol. 23, no. 3, p. 34-41, 2004.

[65] T. E. Maliki and J.M. Seigneur, "User-centric Mobile Identity Management Services," *Management*, p. 33-76, 2008. [Online]. Available: http://asg.unige.ch/publications/TR08/ASG2008-3.pdf

[66] E. I. Tatli and S. Lucks, "Mobile Identity Management Revisited," *Electron. Notes Theor. Comput. Sci.*, vol. 244, p. 125-137, August, 2009.

[67] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimization:Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management," V0.34, 10 August, 2010. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.

[68] M. Veeningen, B. D. Weger and N. Zannone, "Modeling identity-related properties and their privacy strength," in *Proceedings of the 7th International conference on Formal aspects of security and trust*, ser. FAST'10. Berlin, Heidelberg: Springer-Verlag, p. 126-140, 2011.

[69] G. Alpár, J. H. Hoepman and J. Siljee, "The Identity Crisis. Security, Privacy and Usability Issues in Identity Management," *CoRR*, vol. abs/1101.0427, 2011.

[70] M. S. Ferdous, A. Jøsang, K. Singh and R. Borgaonkar, "Security Usability of Petname Systems," in *Identity and Privacy in the Internet Age*, ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 5838, p. 44-59, 2009.

[71] D. O. Jaquet-Chiffelle, E. Benoist, R. Haenni, F. Wenger and H. Zwingelberg, "Virtual Persons and Identities," in *The Future of Identity in the Information Society*, Springer Berlin Heidelberg, p. 75-122, 2009. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01820-6_3

[72] B. Priem, R. Leenes, E. Kosta and A. Kuczerawy, “The Identity Landscape,” in *Digital Privacy*, ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 6545, p. 33-51, 2011.

[73] R. Joosten, D. Whitehouse and P. Duquenoy, “Towards a Meta Model for Identity Terminology,” IFIP/FIDIS Summer School, 2008. [Online]. Available: http://www.buslab.org/SummerSchool2008/slides/Rieks_Joosten.pdf.

[74] Wikipedia, “Identifier,” Accessed on 28 June, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Identifier.

[75] H. Tajfel and J. C. Turner, “An integrative theory of intergroup conflict,” in *Monterey Workshop*, 1979.

[76] A. Jøsang, M. A. Zomai and S. Suriadi, “Usability and privacy in identity management architectures,” in *ACSW ’07: Proceedings of the fifth Australasian symposium on ACSW frontiers’*, p. 143-152, 2007.

[77] J. Lewis, “Enterprise Identity Management: It’s About the Business,” The Burton Group Directory and Security Strategies Report, v1, 2 July, 2003.

[78] ITU-T, “Baseline capabilities for enhanced global identity management and interoperability,” September, 2009. [Online]. Available: http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.1250.

[79] Wikipedia, “Service provider,” Accessed on 29 June, 2011. [Online]. Available: http://en.wikipedia.org/wiki/Service_provider.

[80] “Using Access Control Lists (ACLs)”. [Online]. Available: http://www.hp.com/rnd/support/manuals/pdf/release_06628_07110/Bk2_Ch3_ACL.pdf.

[81] D. W. Chadwick, “Federated Identity Management,” in *FOSAD 2008/2009*, ser. LNCS, Berlin: Springer-Verlag, no. 5705, p. 96-120, January, 2009.

[82] A. M. Al-Khouri, “Data ownership: who owns ‘my data’?” *International Journal of Management & Information Technology*, vol. 2, no. 1, p. 1-8, 2012.

[83] Facebook, “Statement of Rights and Responsibilities,” Accessed on 14 February, 2014. [Online]. Available: https://www.facebook.com/legal/terms/.

[84] I. Brown, “Could even Facebook become a convert to privacy?” 24 February, 2014. [Online]. Available: http://www.theguardian.com/commentisfree/2014/feb/24/facebook-privacy-convert-personal-data-mining.

[85] Guardian, "Edward Snowden Revelations," Accessed on 21 February, 2014. [Online]. Available: http://www.theguardian.com/world/edward-snowden.

[86] Javelin, "2013 IDENTITY FRAUD REPORT: Data Breaches Becoming a Treasure Trove for Fraudsters," February, 2013. [Online]. Available: https://www.javelinstrategy.com/brochure/276.

[87] National Check Fraud Center, "Identity Theft and Assumption Deterrence Act of 1998:Title 18 USC 1028," 1998. [Online]. Available: http://www.ckfraud.org/title_18.html.

[88] N. Mitchison, M. Wilikens, L. Breitbach, R. Urry and S. Portesi, "Identity Theft - A Discussion Paper," Tech. Rep., 2004.

[89] B.J. Koops and R. Leenes, "Identity theft, identity fraud and/or identity-related crime," *Datenschutz und Datensicherheit-DuD*, vol. 30, no. 9, p. 553-556, 2006.

[90] B. AlFayyadh, P. Thorsheim, A. Jøsang and H. Klevjer, "Improving usability of password management with standardized password policies," *The Seventh Conference on Network and Information Systems Security (Sécurité des Architectures Réseaux et des Systèmes d'Information) (SARSSI 2012)*. Cabourg, May 2012. ISBN 978-2-9542630-0-7. Available: http://folk.uio.no/josang/papers/ATJK2012-SARSSI.pdf.

[91] M. S. Ferdous, M. J. M. Chowdhury, M. Moniruzzaman and F. Chowdhury, "Identity federations: A new perspective for Bangladesh," in *International Conference on Informatics, Electronics Vision (ICIEV), 2012*, p. 219-224, May, 2012.

[92] Liberty Alliance Whitepaper, "Benefits of Federated Identity to Government," March, 2004. [Online]. Available: http://projectliberty.org/liberty/content/download/388/2723/file/Liberty_Government_Business_Benefits.pdf.

[93] "Identity Management Systems (IMS): Identification and Comparison Study," Accessed on 7 September, 2013. [Online]. Available: https://www.datenschutzzentrum.de/idmanage/study/ICPP_SNG_IMS-Study.pdf.

[94] R. Leenes, S. Poetzsch, M. Meints, B. Priem and R Husseiki, "D3.12: Federated Identity Management - what's in it for the citizen/customer?" 10 June, 2009. [Online]. Available: http://www.fidis.net/fileadmin/fidis/deliverables/new_deliverables/fidis-wp3-del3.12.Federated_Identity_Management.pdf.

[95] S. Pötzsch, K. Borcea-Pfitzmann, M. Hansen, K. Liesebach, A. Pfitzmann and S. Steinbrecher, "Requirements for Identity Management from the Perspective of Multilateral Interactions," in *Digital Privacy*, ser. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 6545, p. 609-626, 2011.

[96] T. Kölsch, J. Zibuschka and K. Rannenberg, “Digital privacy,” ch. Privacy and Identity Management Requirements: An Application Prototype Perspective, Berlin, Heidelberg: Springer-Verlag, p. 735-749, 2011.

[97] WP3, Future of Identity in the Information Society, “Study on Mobile Identity Management,” May, 2005. [Online]. Available: http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp3-del3.3.study_on_mobile_identity_management.pdf.

[98] B. Schilit and M. Theimer, “Disseminating active map information to mobile hosts,” *Network, IEEE*, vol. 8, no. 5, p. 22-32, 1994.

[99] Sky News, “Second Online Security Breach Hits Sony,” 4 May, 2011. [Online]. Available: http://news.sky.com/story/853559/second-online-security-breach-hits-sony.

[100] BBC, “LinkedIn passwords leaked by hackers,” 7 June, 2012. [Online]. Available: http://www.bbc.co.uk/news/technology-18338956.

[101] L. Constantin, “Hackers leak 1 million Apple UDIDs,” 7 September, 2012. [Online]. Available: http://www.pcworld.com/article/261869/hackers_leak_1_million_apple_udids_allegedly_stolen_from_fbi_laptop.html.

[102] J. J. Borking and C. D. Raab, “Laws, PETs and Other Technologies for Privacy Protection,” *Journal of Information, Law and Technology*, p. 1, 2001.

[103] M. Hansen, “Privacy-Enhancing Technologies,” *Alexander RoSSsnagel (Ed.): Handbuch Datenschutzrecht; Verlag C.H. Beck*, München, p. 291-324, 2003.

[104] NISTWP, “Electronic Authentication Guideline: INFORMATION SECURITY,” April, 2006. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.

[105] “Authentication Processing Filters in SimpleSAMLphp”. [Online]. Available: http://simplesamlphp.org/docs/stable/simplesamlphp-authproc.

[106] D. W. Chadwick, G. L. Inman, K. W. Siu and M. S. Ferdous, “Leveraging social networks to gain access to organisational resources,” in *Proceedings of the 7th ACM workshop on Digital identity management*, ser. DIM ’11. New York, USA: ACM, p. 43-52, 2011.

[107] D. Recordon and D. Reed, “OpenID 2.0: A Platform for User-centric Identity Management,” in *Proceedings of the Second ACM Workshop on Digital Identity Management*, ser. DIM ’06. New York, USA: ACM, p. 11-16, 2006.

[108] “Microsoft Windows CardSpace”. [Online]. Available: http://www.microsoft.com/windows/products/winfamily/cardspace/default.mspx.

[109] “Shibboleth”. [Online]. Available: http://shibboleth.internet2.edu/.

[110] “ZXID”. [Online]. Available: http://www.zxid.org/.

[111] “SQLite Database”. [Online]. Available: http://www.sqlite.org/.

[112] “SQLCipher”. [Online]. Available: http://sqlcipher.net/.

[113] “OpenSAML 2.x”. [Online]. Available: https://wiki.shibboleth.net/confluence/display/OpenSAML/Home.

[114] “OpenID4Java Library”. [Online]. Available: http://code.google.com/p/openid4java/.

[115] “PHP OpenID Library by JanRain, Inc.”. [Online]. Available: http://www.janrain.com/openid-enabled.

[116] Jetty Eclipse, “Jetty WebServer”. [Online]. Available: http://www.eclipse.org/jetty/.

[117] Wikipedia, “Contactless smart card,” Accessed on 4 November, 2014. [Online]. Available: http://en.wikipedia.org/wiki/Contactless_smart_card.

[118] Smart Card Alliance, “NFC Resources,” Accessed on 4 November, 2014. [Online]. Available: http://www.smartcardalliance.org/smart-cards-applications-nfc/.

[119] ISO/IEC International Standard 14443-1:2008, “Identification cards - Contactless integrated circuit cards - Proximity cards - Part 1: Physical characteristics,” 2008. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=39693.

[120] ISO/IEC International Standard 14443-2:2010, “Identification cards - Contactless integrated circuit cards - Proximity cards - Part 2: Radio frequency power and signal interface,” 2010. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50941.

[121] ISO/IEC International Standard 14443-3:2011, “Identification cards - Contactless integrated circuit cards - Proximity cards - Part 3: Initialization and anticollision,” 2011. [Online]. Available: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50942.

[122] ISO/IEC International Standard 14443-4:2008, “Identification cards - Contactless integrated circuit cards - Proximity cards - Part 4: Transmission protocol,” 2008. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50648.

[123] "OpenID Provider Authentication Policy Extension 1.0," 30 December, 2008. [Online]. Available: http://openid.net/specs/openid-provider-authentication-policy-extension-1_0.html.

[124] OpenID, "OpenID Attribute Exchange 1.0 - Final," 5 December, 2007. [Online]. Available: http://openid.net/specs/openid-attribute-exchange-1_0.html.

[125] Marketsandmarkets.com, "Context Aware Computing Market - Global Advancements, Emerging Applications, Worldwide Forecasts & Analysis (2013 - 2018)," March, 2013. Accessed on 4 July, 2015. [Online]. Available: http://www.marketsandmarkets.com/Market-Reports/context-aware-computing-market-763.html.

[126] K. Wrona and L. Gomez, "Context-aware security and secure context-awareness in ubiquitous computing environments," in *Proceedings of the XXI Autumn Meeting of Polish Information Processing Society*, p. 255-265, 2005. [Online]. Available: http://proceedings2005.imcsit.org/docs/75.pdf

[127] R. Nick, J. Pascoe and D. Morse, "Enhanced reality fieldwork: the context-aware archaeologist assistant," in *Computer Applications & Quantitative Methods in Archaeology*, Exon, Ed., vol. 0. Archaeopress, 1997.

[128] R. Hull, P. Neaves and J. Bedford-Roberts, "Towards situated computing," in *First International Symposium on Wearable Computers*, p. 146-153, 1997.

[129] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, ser. HUC '99. London, UK, Springer-Verlag, p. 304-307, 1999.

[130] Open Geospatial Consortium, "Geospatial eXtensible Access Control Markup Language (GeoXACML), Version 1 Corrigendum," 12 May, 2011. [Online]. Available: http://www.opengeospatial.org/standards/geoxacml.

[131] OGC, "OpenGIS Geography Markup Language (GML) Encoding Standard, Version: 3.2.1," 27 August, 2007. [Online]. Available: http://www.opengeospatial.org/standards/gml.

[132] Sun Microsystems, "Sun's XACML Implementation," Accessed on 1 March, 2013. [Online]. Available: http://sunxacml.sourceforge.net/.

[133] HERAS-AF, "HERASAF XACML Core," Accessed on 1 March, 2013. [Online]. Available: http://www.herasaf.org/downloads/binaries.html.

[134] "enterprise-java-xacml," Accessed on 1 March, 2013. [Online]. Available: https://code.google.com/p/enterprise-java-xacml/.

[135] "Commercial Implementation of GeoXACML". [Online]. Available: http://geoxacml.secure-dimensions.net/.

[136] "GeoServer SVN Repository," Accessed on 2 May, 2013. [Online]. Available: https://github.com/opengeo/geoserver-2.1.x/tree/master/community/geoxacml.

[137] Wikipedia, "QRCode," Accessed on 1 April, 2013. [Online]. Available: http://en.wikipedia.org/wiki/QR_code.

[138] "Fake GPS Location". [Online]. Available: https://play.google.com/store/apps/details?id=com.lexa.fakegps&hl=en.

[139] "Location Spoofer". [Online]. Available: https://play.google.com/store/apps/details?id=org.ajeje.fakelocation&hl=en.

[140] S. Cantor, "Shibboleth Attribute Release Policies," 7 January, 2008. [Online]. Available: https://wiki.shibboleth.net/confluence/display/SHIB/IdPARPConfig.

[141] "Shibboleth Attribute Filter Policy," 5 April, 2013. [Online]. Available: https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAddAttributeFilter.

[142] "Shibboleth Attribute Release Policy Editor - ShARPE". [Online]. Available: http://www.federation.org.au/twiki/bin/view/Federation/ShARPE.

[143] OpenID, "OpenID Simple Registration Extension 1.1 - Draft 1," 6 December, 2006. [Online]. Available: http://openid.net/specs/openid-simple-registration-extension-1_1-01.html.

[144] "Yahoo OpenID". [Online]. Available: http://openid.yahoo.com.

[145] "Google". [Online]. Available: https://www.google.com.

[146] "LiveJournal". [Online]. Available: http://www.livejournal.com/.

[147] "myopenid". [Online]. Available: https://www.myopenid.com/.

[148] "Symantec Personal Identity Portal". [Online]. Available: https://pip.verisignlabs.com.

[149] "Facebook". [Online]. Available: https://www.facebook.com.

[150] "LinkedIn". [Online]. Available: http://www.linkedin.com.

[151] "Twitter". [Online]. Available: https://twitter.com.

[152] S. Kellomäki, "Query Extension for SAML AuthnRequest (Draft)," 22 April, 2008. [Online]. Available: http://zxid.org/tas3/anrq-index.html.

[153] J. M. Spivey and J. Abrial, "The Z notation." Prentice Hall Hemel Hempstead, 1992.

[154] A. W. Roscoe, "The theory and practice of concurrency." Prentice Hall, 1998. [Online]. Available: http://www.cs.ox.ac.uk/people/bill.roscoe/publications/68b.pdf

[155] T. Gibson-Robinson, P. Armstrong, A. Boulgakov and A. Roscoe, "FDR3 - A Modern Refinement Checker for CSP," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, Springer Berlin Heidelberg, vol. 8413, p. 187-201, 2014.