IDENTITY MANAGEMENT ARCHITECTURE AND IMPLEMENTATION: EVALUATION AND IMPROVEMENT

by

CHRISTOPHER STAITE

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
March 2012

Abstract

The definition of identity varies, and on the Internet it can be difficult to keep track of. Rather than trying to discuss the philosophical question of “who am I?”, I like to define your digital identity as the information you place on the Internet (actively or passively). Managing this identity comes down to what information you give out and how to protect and modify that information. This thesis focuses on the latter half, the protection and modification of online identities and only skims the realms of protecting the information given to third parties.

A distinct lack of drive in the development of technologies for managing authentication has dogged the Internet for some time. Numerous efforts have been made to simplify administration, but open protocols meant for simplifying the user experience have had little promotion and ended up forgotten or used to simplify administration. The question that needs to be answered, as usual in research, is why? Studies have shown that password fatigue is a very real issue and identity theft is increasing. Companies will always optimise their time and resources, but academics need to focus their work on optimising the user experience.

In this thesis, a study of existing work produces a methodology to evaluate previous developments. This aids in determining where progress has been made in previous iterations and how, leading to a new development in identity management focussed on the needs of the end user. Finally, two implementations are created to realise this new form of identity management.

For my wonderful Gran,

without whom I would not be who I am today.

Given the chance she would have read every

last word in this thesis, even though she

would understand few.

Acknowledgements

First, I would like to thank my supervisor, Dr Rami Bahsoon. As co-examiner on my final year project it is almost exactly four years to the day between us meeting and my thesis submission. Without Rami I would not have had, nor taken, the opportunity to perform the work in this document or enjoy the three and a half years it took to produce. His guidance and support has provided me with an excellent foothold due to my lack of a Masters.

My Dad has provided me with wholehearted support throughout my education, for this I am very greatful. In particular, his command over the English language not only improved my writing through his many, many proof readings, but also evolved the quality of my writing in the first place.

During my work I have been accepted into the Computer Security Research Group at The University of Birmingham. They have willingly hosted a number of talks on my work and provided useful feedback and discussion. One member, Dr Tom Chothia, in particular aided greatly in the development of my ideas about trust and methods for identity management. His "hacking" group also provided a rather nice way to spend a Friday afternoon.

I would like to thank the numerous members of Room 117 who have been my surrogate office throughout my time as a docutoral resarcher. Their company has made my time as a research student far more enjoyable. Particularly, I would like to thank Dr Olaf Klinke for his tireless efforts in attempting to explain the mathematics behind complex cryptography. Dr Matt Smart for begrudgingly giving assistance in my use of ProVerif and cryptography in general. Benjamin Woolford-Lim was kind enough to offer feedback on my thesis and its numerous issues with understandability.

Finally, my PhD twin, Vivek Nallur, has been a constant aid in all of my research and made my time as a doctoral researcher an absolute pleasure. He is an extra- ordinary person, with time for everyone and valid opinions about many research fields. His feedback on this thesis has brought the contributions to the fore and undoubtedly improved the understandability no end.

CONTENTS

1 Introduction **1**

1.1 Setting the scene for identity management research 7

1.2 Research questions 11

1.3 Overview 13

2 Motivation and research objectives **17**

2.1 The purpose of identity management 18

2.1.1 Individuals 19

2.1.2 Identity Providers 21

2.1.3 Services 22

2.2 Motivating example 23

2.3 Identity management and software engineering 25

2.4 Contributions 26

2.4.1 Framework for evaluation 27

2.4.2 Evaluation of existing architectures and implementations 29

2.4.3 Design of a new identity management architecture 29

2.4.4 Implementation of new identity management architecture 30

2.5 Conclusion 31

3 The evolution of identity management **33**

3.1 Roles, research and implementations of traditional identity management . . . 34

3.2 Evolution of identity management to single sign-on 37

3.3 User centric identity management 40
3.4 Conclusion 43

4 An ATAM evaluation 45

4.1 Performing an ATAM evaluation 46
4.2 ATAM Evaluation Results 51
4.3 Conclusion 53

5 Systematically evaluating identity management architectures 55

5.1 Systematic review of existing identity management architectures 57
5.1.1 Research method 57
5.1.2 Execution of search 60
5.1.3 Results 61
5.2 High level modelling of identity architectures 62
5.2.1 Evaluating by example 67
5.3 Metrics for identity architectures 68
5.3.1 Authentication 70
5.3.2 Profile 74
5.3.3 Evaluating by example 78
5.4 Quantitative analysis 87
5.4.1 Evaluating by example 89
5.5 Trade-offs 90
5.6 Comparison with ATAM evaluation 91

6 Preliminaries 93

6.1 Group Signature Scheme 94
6.2 Monotonic Counter 95
6.3 Dynamic accumulators 96
6.4 Diffie-Hellman key exchange 97

7 Device based identity **99**

7.1 Evolving identity management 100

7.2 Device based identity management with an external trusted service 106

7.2.1 Requirements 108

7.2.2 Implementation 109

7.2.3 Evaluation 115

7.3 Device based identity management through three-factor security 118

7.3.1 Overview 118

7.3.2 Evaluation 125

8 Evaluation revisited and discussion **133**

8.1 Identity architecture evaluation 135

8.2 Device based identity 137

8.2.1 Type one 137

8.2.2 Type two 139

8.2.3 Comparison 141

8.2.4 Comparison with prediction 143

8.3 Summary of evaluation results 144

9 Conclusions **147**

9.1 What have we learnt? 148

9.2 The future of identity management 150

LIST OF FIGURES

4.1 Brief for classroom experiment 50

5.1 Key for high-level architecture evaluation 67

5.2 Framework model for service based identity management 68

5.3 Framework model for OpenID identity management 68

5.4 Qualitative analysis of service based authentication 82

5.5 Qualitative analysis of service based authentication profile data 82

5.6 Qualitative analysis of OpenID authentication 86

5.7 Qualitative analysis of OpenID profile data 86

5.8 Quantitative analysis of service authentication 90

5.9 Quantitative analysis of service registration 90

7.1 Framework model for device based identity management 101

7.2 Qualitative analysis of device based authentication 105

7.3 Qualitative analysis of device based profile data 105

7.4 Device based authentication framework 108

7.5 Setup protocol for group signature device based identity management 111

7.6 Protocol to add a device to the identity in group signature device based identity management 112

7.7 Protocol for revoking a device from the identity in group signature device based identity management 113

7.8 Profile data in device based identity management 119

7.9 Protocol for creating a new identity. 120

7.10 Protocol for adding a device to the identity. 121
7.11 Protocol for revoking a device from the identity. 123
7.12 Protocol for registering an identity with a service. 124
7.13 Protocol for authenticating with a service. 125
7.14 HTTP requests for device addition 126
7.15 UI device addition in device based identity 128
7.16 UI device removal in device based identity 129
7.17 UI registration of a new device based identity 129
7.18 The add protocol implemented in ProVerif 132

8.1 Qualitative analysis of type one device based authentication 142
8.2 Qualitative analysis of type one device based profile data 142
8.3 Qualitative analysis of type two device based authentication 142
8.4 Qualitative analysis of type two device based profile data 142

LIST OF TABLES

5.1 Summary of systematic literature review results 61

5.2 Authentication metrics from literature review 63

5.3 Profile metrics from literature review 64

8.1 Comparison of device based identity implementations against initial architecture evaluation 143

8.2 Evaluation of service based identity management 144

CHAPTER 1

INTRODUCTION

Our true reality is in our identity and unity with all life.

— Joseph Campbell

The ubiquity of the Internet has been strengthening due to a number of factors. The first of these is simply the ability to access the Internet faster, and from numerous and more portable devices. Developments made in wireless technology not only permit cheaper data connections, but faster and further range than ever before. This means the general population has access to the Internet almost constantly, at a reasonable speed from many devices. The devices themselves have also been getting more advanced, and not simply in terms of their radio abilities. The dawn of the "smartphone" has opened most of the facilities of the Internet to mobile users. When the Wireless Application Protocol (WAP) originated in 1997, phone browsers were able to handle very little of what the Internet had to offer. Small images, text and links were the limits which mobile browsers handled. Due to the screen resolutions at the time, any more functionality would have perhaps been futile. Email was also a facility provided, popularised by Research In Motion (RIM) in 1999 with their BlackBerry line of handsets, aimed at businesses users for constant email access. Development was relatively stagnated in the area with minor improvements, the most functional being those running on Windows Mobile (derived from Windows CE). Apple launched the iPhone in 2007 which started a huge leap forward in the "smartphone" technology. RIM had to work hard to keep their phones relevant and a relatively unknown manufacturer, High Tech Computer Corporation (HTC) was able to break into the market with the Google Android OS rival to iPhone OS (now known as iOS). Finally, the ubiquitous mobile computing that had been predicted almost two decades prior by Weiser [65] of Xerox was becoming reality.

With this ubiquity, ever more daily transactions are taking place using it as a medium. Presenting one's digital self is often a difficult task. The nature of the Internet creates an anonymity, users interacting behind the safety of their own devices. Original Internet applications did not consider anonymity, or privacy. These were simply providing information as it was requested. However, the advent of e-commerce and the more recent development of social networking have revolutionised what information is made available about users on a free and open Internet. Suddenly, it matters if the data being transmitted from a personal device to the masses of interconnected machines is read by someone with malicious intentions. In fact, it matters where the information ends its journey and is processed or stored. These are simply matters of personal security. Privacy is a whole separate issue. From a security perspective, no one could gain access to your house or bank account with the knowledge of what party you attended last week. However, if a friend that you failed to invite found out, that could cause, at very least, tension between you. This is one simple example of a privacy issue, of which there are many more. With the inclusion of GPS and cellular mast triangulation on mobile phones, location based privacy is quickly becoming the most pressing. The most common invasion of privacy is by advertisers. In order to create targeted advertisement campaigns, many different aspects of a user's digital profile are collected through conscious and surreptitious methods. Tracking users to build these profiles can be done in a number of ways. The most commonly used method is cookies. These are tokens, stored in browsers by remote servers in order to uniquely identify users as they browse websites. Another method is using Internet Protocol (IP) addresses to link multiple connections to remote hosts. However, the use of technologies such as Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP) can render this technique less than ideal, if not useless, for tracking individuals. These technologies exist due to the lack of IP addresses on the Internet. NAT permits a group of users to share a single IP, making all traffic appear to be from the same node. DHCP allows the IP address to be changed by the Internet Service Provider (ISP) and hence allocate their range of IP addresses to clients as they connect to the Internet.

Of course, there are many other situations where users give away their personal details, and do so freely. In the case of e-commerce, shopping preferences, delivery and payment information are all required for a transaction to take place. This may be modelled in a manner similar to that of a transaction in a high street store. However, in a store no delivery address need be given, as the goods can be taken immediately. No payment information is required as anonymous cash may be used. A user may not wish to entrust a store with such information, but without doing so, the utility of online purchasing will elude them. In the modern world, many people take it for granted that their privacy will need to be traded for utility, but they should not need to. Problems such as these are classified as *Identity management*, forming a research area which is becoming more important as more people use more online services to perform daily task. Identity management covers a large range of topics over and above authentication and profile distribution, these include access control and privacy. This thesis focusses on simply the authentication of an identity in the scenario of a web service and the storage and distribution of profile data. The issues of what data to distribute and where is out of the scope of this thesis.

The issues discussed have simply considered digital profiles. However, this is not the most pressing issue to users [47]. Websites often utilise authentication in order to link users to their accounts. This provides security to the user, and allows services to keep track of users between devices and network connections. There are many situations where users desire a persistent online identity, such as Internet banking, social networking or email. Authentication takes many forms, the most widespread being that of a username[1] and password pair. The use of different usernames and passwords on websites is promoted in order to ensure that if one site is malicious, or compromised in some way, a user's whole digital identity is not also compromised. Websites will also inflict arbitrary rules and restrictions on usernames and passwords, such as characters it may, must, or must not contain, as well as minimum and maximum permitted lengths. All of these factors lead to password fatigue. The prevalence of "Forgot your password?" and even "Forgot your username?" links on websites demonstrates this problem.

As stated, authentication is the most pressing issue. This is due to authentication being the most visible aspect to the end-user. Users tend to ignore their digital profile, most of which is collected from browsing and purchasing history rather than explicit user input. This area of research has received much less attention than that of authentication. The preference towards authentication over profile management in past work, derived from the tendency to view the issue from a user management perspective. It is only logical that those who maintain and control user accounts, and therefore have the ability to modify how they are implemented, would ease their own workload. To this end, access control for users on

[1]Very often the username utilised is in fact the user's email address.

distributed networks, followed by grouping of users in order to permit easier access control, became a feature of operating systems as soon as multiple users existed on a system. Prior to the use of Internet services, system access control dictated the information that could be accessed, and the programs that could be executed. When the Internet started to be used for accessing services, multiple proprietary authentication protocols existed, and hence were incompatible for access control across domains. In the beginning this was acceptable as Internet services were generally free information resources. As the Internet matured, services began to become chargeable. In academia, societies began to make journals and conference papers available online for those with memberships. Buying an institution-wide license for each society would require some or all of: centralising access; distributing a single set of credentials between all users; limiting access to a certain IP range; or maintaining all users in the institution with the society. All of these options have undesirable consequences for either the institution or the society, or both. To provide access in these circumstances, a method of authenticating with the institution and then gaining an single-use access key in order to access the digital library mitigates the administration problems for both parties. Hence, single sign-on authentication was born. In the United Kingdom, a system called Athens was developed in the late 1990s, centralising national educational authentication. This system was mainly used for library services. At the start of the millennium, Shibboleth [24] was developed to perform a similar function, but this time as an open API.

Problems with authentication go further than simply academia. Similar systems were developed within private companies to perform similar functions. Companies who provided multiple services to users, and hence had implemented internal centralised authentication protocols, made them publicly available. One example of this is the Microsoft Passport, presented as a single identity for all web commerce. Many criticisms where made of this approach, such as being closed and numerous security vulnerabilities, which are still relevant today for other providers attempting the same model (Facebook Connect, for example). An open model permits users to decide whom they wish to trust with their identity, and removes the chance of a monopoly. OpenID [54] was developed precisely to generate an open API for online authentication. Users may use any identity provider they wish, including one of their own creation, providing it complies with the specified protocol. The disadvantage to this approach is that services are unable to impart trust to the identity provider as security depends on the Domain Name System, which is insecure. Verification of the identity provider is futile, since a service has no way of propagating information about all possible identity providers.

The most recent implementation available in the field is that of Mozilla BrowserID [1]. Through the use of an email address, users are able to authenticate with services. Many services verify a user's email address upon registration. Therefore it is a logical proposal to utilise email for authentication. A single authentication for each browser is required, which persists until the user removes the browser from the identity (logs out). The decentralised nature of the development means that all the trust is managed by email which, despite its flaws, is common practice. A bigger issue exists with the decentralised nature of the system: if a user were to forget to log out (remove their security credentials) from a browser when they left it, they would not be able to rectify this remotely. Changing their email address is not a desirable option for account revocation.

A paper by Herley & van Oorschot demonstrates the difficulties of changing authentication schemes away from those which are password based [34]. Their work motivates the need for determining the best fit identity management system for a given scenario. Utilising objective and systematic evaluation of identity management architectures and implementations, this thesis intends to enable just that. A number of existing frameworks have been developed for both generic architecture evaluation, and identity management specific evaluation. As will become evident later, each of these is currently inadequate for this purpose. One paper from an Australian government department goes so far as to state that existing literature does not well understand identity management architecture [66]. Research in this area is very timely and is a major consideration in the CORDIS FP7[2] and is highly likely to remain in FP8. Generic solutions require a large amount of domain specific knowledge and are often unreliable for determining a consistent result. Existing identity management specific

[1]`http://browserid.org/`
[2]`http://cordis.europa.eu/fp7/ict/security/eid-management_en.html`

evaluation has focused on a specific attribute of identity management: authentication, privacy, usability, security, to name a few. Where improvements may be made in one area, it is important to determine what the trade-offs made with regards to other areas are. This allows developers to make informed decisions about the type of identity management to utilise in their services. Furthermore, it assists researchers to develop improved identity management systems based on missing features or abilities of existing techniques. This is the focus of Chapter 5 and one of the major contributions of this thesis. In order to develop a methodology for this purpose, a systematic review is performed to evaluate, and consolidate the existing literature in the area of identity management evaluation. A generic architecture trade-off analysis method is then chosen for a classroom experiment in order to provide a benchmark for trade-off analysis in the field of identity management.

To demonstrate the ability of this evaluation method to drive future research, a new style of identity management is developed in Chapter 7. This style increases the usability over existing styles by using the device as the trusted entity, similarly to BrowserID. This style may be used to develop multiple types of identity management architectures, of which two are created. One technique provides many security improvements over existing identity management implementations, but the method requires complex cryptography to be performed, and large data payloads. The second modifies the trust model in order to decrease the computational load of the system. Finally, these styles are evaluated with the framework from which they were derived, in order to go full circle and confirm that the intended improvements were made.

1.1 Setting the scene for identity management research

Certain aspects of the challenge of identity management have been focussed upon more than others. For obvious business reasons, collection of personal data and identifying users between accesses in order to link the collected data has been extensively covered in industry, especially that of advertising. Similarly, management of large numbers of users from an administration point of view has also been covered. Only recently has research begun to consider the problems faced by users. From an industrial point of view, simplifying identity management for the end-user requires far less customer support. Additionally, a study conducted by Blue Research in 2011 [1] has shown that users are more likely to use a website if registration is either simple or not required. In an increasingly competitive market, services which adopt user friendly identity management have a competitive edge compared to those that don't. Despite these compelling arguments, little research has been done from a software engineering point of view. This perspective is important in order to systematically evaluate the options available to companies, and for the development and evaluation of new identity management architectures and implementations.

Improvements made to existing architectures are sometimes revolutionary, and therefore are capable of either maintaining or improving every aspect of those that they replace. These kinds of improvement are generally rare occurrences. Instead, improvements are often made by sacrificing other aspects which were not the focus of development. Due to this, methods have been developed which are capable of evaluating architectures for these trade-offs. Performing these evaluations requires a large amount of domain specific knowledge, and is often not deterministic with a large amount of subjective opinion thrown in. More deterministic processes should be utilised in order to ensure that all relevant aspects are considered. Combining trade-off analysis methods and objective processes will result in a methodology which is not only more objective, but is capable of evaluating all aspects of an identity management architecture. Such an evaluation is therefore capable of determining which new ideas are revolutionary in the field, rather than mere improvements. The problem of the deployment and utilisation of new identity management architectures by services and users still exists. Other than easing the transition, this problem lies out of the scope of this thesis, and is instead left as a discussion point for the big players, those who have millions of existing users.

Trust is an important aspect of identity, digital or otherwise; users who do not trust the

[1] `http://www.janrain.com/consumer-research-social-signin`

entity which is requesting and collecting personal information are unlikely to give that information freely. Segmentation of digital identity exists currently, users maintaining an account at one service which contains information the user has given to that particular service. Terms and conditions of sale on e-commerce websites and privacy policies are available for users to browse, should they wish to concern themselves with the treatment of their personal information once it has been supplied. Many legal frameworks are also in place, such as the Data Protection Act (1999) in the United Kingdom. These set out rules and regulations with regard to the collection, storage and disposal of data both physically and digitally, including but not restricted to Personally Identifiable Information (PII).

Segmentation of profiles, combined with the ever increasing number of services, causes a usability nightmare and credential overload for users. Administration solutions proposed by industry and academia simply solve access control issues for groups within a single organisation. General users do not have the luxury of a centralised identity provider due to the lack of corporate incentive to operate one. Despite this, a number of identity providers have emerged, some such as MyVidoop [1] utilising advertising during the authentication and others such as Google [2] whose intentions are unclear. As of early 2012, Google has released a new privacy policy in which it expressly permits the sharing of information between its services for more relevant advertisement. A number of commentators have deemed this to contravene European data protection, something that Google refutes. The open API (in this case, OpenID) that both of these identity providers utilise, allows users to become their own identity provider. However, this requires knowledge of web service administration and a financial outlay on the part of the user. Services which permit authentication with this form of identity API also need to trust that a user who seems to be the same in two sessions, is actually the same. The trust in this API is derived from the identity provider which, itself is identified by simply the domain name (a mechanism never designed to maintain trust). This is mitigated through the use of Secure Socket Layer (SSL) certificates. However this is not a requirement of the OpenID standard, and can therefore be easily bypassed.

[1]`https://myvidoop.com/`
[2]`http://code.google.com/apis/accounts/docs/OpenID.html`

Developing a new style of identity management is sorely needed in order to resolve the problems caused by decentralised identity. This new style should not depend on existing protocols, which were never designed for security, and should still permit a free market of identity providers. In all identity systems, there must be a trusted entity in order to ensure security. Shifting trust between entities in the identity ecosystem alters the properties exhibited. Using the decentralised method, trust is placed in each service. However, separate digital identities for each service allows users to monitor which information is made available to each entity. Maintaining disparate profile attributes is also difficult. Consider the example of moving home: a large number of services need to be notified of a change, however there is no centralised place to perform this update, and no record of the services with this attribute. Now consider another alternative; a centralised location where the home address is stored. This situation makes updating the profile attribute simple. However, who controls access to this piece of information, and what about the differentiation between work and home addresses? Many more issues arise when addressing the problem of distributed profile maintenance with a centralised profile. A major problem that exists, is that of a compromised account. If a single account is compromised in the distributed model, only the information given to that provider is compromised [1]. In centralised profile model, a user's entire digital identity may be compromised at a single point.

In order to provide a good level of security to a centralised identity management model, the use of much stronger authentication credentials is required. Additionally, the identity provider should not be permitted access to services, or profile data. This ensures that a compromised identity provider does not lead to compromised identities and obtaining access to individual identities is much harder. In order to supply authentication credentials with more entropy than is comfortable for a human, it makes sense to utilise the device which is being used to access the services. Previous identity management enhancements have utilised separate devices, known as Portable Authentication Devices (PADs) [39]. This requires users to manage another piece of hardware, and either have a connection which is universal between

[1] assuming the authentication credentials are not also compromised or the user utilises a different credential for each service

all service accessing devices, or transferring authentication data from the PAD to the service accessing device, manually. Instead, it should be possible to maintain the functionality of a single PAD distributed between all devices owned by the user. Not only does this provide enhanced security, but usability is also improved.

1.2 Research questions

In order to steer the research in this thesis, a number of key questions are detailed below. Throughout the thesis these questions will be unravelled, culminating in answers to them being consolidated in the conclusions. The contributions listed in Section 2.4 are derived from the answers to these questions.

1. **How can an identity management architecture be evaluated systematically?**

   (a) **What are the requirements of an identity management architecture?**

   As with all software architectures, there are many aspects of identity management architectures by which they may be evaluated. Existing identity management evaluation provides a methodology for specific attributes of identity management architectures. However, these do not provide all the benefits of generic trade-off evaluation methods. Through the combination of these approaches, it should be possible to derive an improved evaluation methodology.

   (b) **Which metrics need to be evaluated?**

   Following from the previous question, requirements specific to identity management need to be determined. A wealth of existing research into the subject indicates the areas which should be evaluated, and in some cases the methodology used to test them previously. Further to this, utilising existing generic frameworks for software engineering evaluation (and a group of software engineers) it should be possible to derive further insight into the evaluation of an identity management architecture.

(c) **What method should be followed to perform a systematic evaluation?**

A large number of existing architecture evaluation methods exist. Most of these methods draw upon the domain expertise of the software engineer evaluating the architecture. Those which don't are tailored to a specific domain. Currently, there are no evaluation methodologies tailored for identity management, and consequently those provided for the generic case must be utilised. However, the lack of domain knowledge possessed by those performing evaluation can cause bias or result in incorrect results. Instead, a better evaluation methodology may be derived from the combination of existing generic methods, and domain specific knowledge drawn from multiple sources.

(d) **What are the benefits of a systematic analysis?**

Through the use of a standard systematic analysis, a standardised result may be derived. Instead of making a decision between architectures, the differences may be evaluated in order to produce conclusions about their make-up. For example, a change in the layout of components in the architecture may produce specific advantages or disadvantages which may then be utilised. The evolution between architectures may also provide a stepping stone for further developments.

2. **What improvements may be made to identity management architectures?**

(a) **Where should the effort of improvements be focused?**

The evaluation provides areas which perform well in certain architectures. By determining the areas which are important to the stakeholders, it is possible to determine which architectures are suitable. If none suit, the results of evaluating existing architectures holds the key, providing subtle pointers as to where modifications may be made to demonstrate certain properties. However, it is only through experimentation that a conclusive answer may be obtained as to whether these modifications have the expected result.

(b) **Which areas of identity management are important to which parties (such as users or services)?**

Stakeholders in architectures are often consulted during the development of new architectures. Modern development paradigms involve architecting systems for unknown users owing to the nature of the Internet. The most pressing issues with identity management exist in the Internet domain, since those with known stakeholders have previously been addressed. It is therefore important to determine which areas affect the stakeholders, allowing future development to focus on areas for users. Identifying the service-related areas is equally important in order to drive usability with existing services and/or promote the integration with new and existing services.

(c) **How can such systematic evaluation inform the design of a new identity management architecture?**

Combining the evaluation of existing architectures and the importance of specific areas, those which are in most need of improvement should become apparent. Improvement in these areas is obviously progress in the field of identity management. However, progress in one area is likely to affect other areas of the architecture. Evaluation of new designs facilitates the definition of improvements and trade-offs.

(d) **How may improvements made in architectural design be implemented?**

Designing a new architecture for evaluation is simply the first stage. These improvements must be exemplified through implementation.

1.3 Overview

This thesis covers two major aspects of research into identity management. Initially, the focus is placed on developing an evaluation methodology for identity management architectures. Through the use of this methodology and existing architectures, improvements are suggested. This shifts the direction of the thesis toward the development of new identity management architectures and implementations. These implementations utilise a change in paradigm for identity management architectures through the use of user devices for credential storage, and hence authentication. With this change, profile management is also considered utilising the features provided by the new technology. Through combination of both software engineering and security research, this thesis is able to not only evaluate and postulate on future improvements, it is able to derive two separate implementations to provide those improvements. With the invention of two new architectures, the identity management evaluation methodology may itself be evaluated.

Chapter 2 is dedicated to an in-depth description of identity management from different perspectives, and presenting a motivating example which will run through the thesis. The three perspectives are those of the individual (user of services), identity providers and services. Identity management has differing priorities depending on the point of view, and therefore it is key to understand each of them. The motivating example is a service where the issues of identity management for both authentication and profile management need to be considered. The development of the service is considered from the service providers' point of view, however the impact of the decisions must be considered from the users' perspective, and to a lesser extent that of the identity provider. The identity provider is an important link in the chain, however their involvement is simply commercial viability. The chapter concludes with a description of the contributions provided by this thesis, in the domains of software engineering and security.

In Chapter 3, the evolution of identity management is reviewed. The major research and implementations in the field are discussed and reviewed. The review begins with a discussion of early identity management, and the development of decentralised systems. Evolution into centralised systems where users may be managed on a central identity provider is then discussed. Finally, the development of so-called "user-centric" identity management is discussed, where open APIs permit users to take control of their digital identity.

Chapter 4 utilises a classroom experiment to perform a generic architectural evaluation.

The results of which motivate the need for a new, specialised, systematic evaluation mechanism.

Developing from this evaluation of identity management technologies, Chapter 5 performs a systematic review of identity management evaluation methods. The methodology includes a set of metrics in order to aid non-domain experts in performing an objective analysis. Additionally, a high level evaluation method is utilised in order to create a “managerial overview” of identity management implementations, assisting the interpretation of the evaluation results. Throughout the chapter, the identity management architectures evaluated in the classroom experiment are re-evaluated utilising the new methodology and metrics. This has the advantage of evaluating the metrics against the existing methods, performing evaluation in-the-small. This methodology may be used to evaluate more implementations in the future, as well as be extended with additional requirements making it extensible and adaptable to future development. Extensions may aid in the development of further advances in identity management architecture

Chapter 6 describes a number of key concepts which will be utilised in the remainder of the thesis to increase flow and readability of the following chapter.

Through the use of the new evaluation methodology, Chapter 7 derives a new style of identity management. This is once again evaluated with the method proposed in Chapter 5 in order to determine if the proposed modifications have the desired effects. This new style makes use of user devices for credential storage and boasts benefits in usability and security for both authentication and profile management. Two different implementations are suggested in this identity management style with differing abilities. Each implementation is evaluated for security, but compared against the reference in the following chapter.

Evaluation of the thesis in-the-large is performed in Chapter 8. Despite evaluation being performed on the methodology and implementations previously, it is important to consider the performance of the evaluation methodology for creating the new identity management style. Additionally, the evaluation methodology is utilised in order to determine the improvements and trade-offs of each of the new identity management implementations.

To conclude, Chapter 9 gives an overview of the contributions of the thesis and provides concise answers to the research questions posed in Section 1.2. Finally, areas which require future research are detailed. These include further development and refinement for the evaluation methodology, and predictions for improvement in identity management implementations. These implementations may be impossible, or have other issues given the current state-of-the-art, but this does not disqualify them from consideration.

CHAPTER 2

MOTIVATION AND RESEARCH OBJECTIVES

First say to yourself what you would be; and then do what you have to do.

— Epictetus

2.1 The purpose of identity management

Identity management is a term used to cover a wide range of problems in many disciplines. The issue with the definition occurs when one considers what identity is and who or what is managing it. The Oxford English Dictionary defines identity as:

> "the fact of being who or what a person or thing is"

While this is a perfectly good definition, understanding what makes up an identity is the challenging part. Digital identity emerged from the need to differentiate between users of individual systems, and subsequently networked systems. Managing authentication details and the distribution of profile information, especially Personally Identifiable Information (PII) is where the need for identity management as a research field arises.

In order to consider identity management further, first let identity be defined in the following manner for the remainder of this thesis. An identity is the sum total of information provided to, and stored on a computer system which may be accessed through a set of credentials. This definition is reasonable because: only the identity of individuals is being considered; accessing a system which contains no personal information cannot be considered a breach of identity. If groups were being considered, then it may be possible that identity is being used as a form of access control, and therefore its security becomes critical. The fact that belonging to the group may be considered a piece of information about the identity owner can make this point moot. To justify the second point, let us consider the scenario that no profile is currently stored for an account at a service, and the credentials for said account may be made public without any further loss of data. This identity is now public, and owned by everyone. However, once the profile begins to be populated with personal information, it is no longer identical to an empty public profile and becomes an identity.

Now that the generic definition of identity management for this thesis has been laid out, it must be considered for the different entities evolved. These entities take differing roles in the management of identity and have different goals depending on their motives, and regulations imposed on them. The types of entity involved in identity management are:

1. Individuals - those that own an identity
2. Identity Providers - those that supply or store identity information
3. Services - those that provide individuals with a service, often based on their identity information.

In 2008, Miller wrote two articles in IEEE Internet Computing regarding identity [47, 46]. These articles summarised identity management to date, and drew conclusions on its uses and problems. The key to identity management is the benefits to both users and companies for storing identity information online. These will be discussed in more depth later. Due to this dual need, it is therefore not surprising that much development has occurred in this area from a commercial perspective. Unfortunately, these developments usually are driven by the needs of the services rather than the needs of the users. One area that is of benefit to both is centralised profile attribute storage. Miller notes that decentralised attributes easily become outdated. However, centralising attribute storage generates privacy concerns. In [47] it is shown that users have a threshold where the benefits are outweighed by the invasion of privacy. Miller goes on to note in [46] that open APIs assist in privacy, and are therefore the way forward. However, the real issue with identity attributes is that users are unsure of the use of attributes. In some cases users freely give information away, and others keep it safe simply owing to their perceived understanding of its use. Of course, this perception may be inherently incorrect.

The remainder of this section will focus on the needs and requirements of each of the identity entities described. Through this description it is hoped that the drive behind developments to date in the area of identity management will become clearer.

2.1.1 Individuals

Individuals, as previously stated, are the owners of identities. They are likely to own multiple identities where services do not share authentication. These individuals provide all information for populating each of their identities either directly or indirectly. Profile information can range from the data given upon registration, to browsing and purchase history. Linked records may also be combined with a profile, for example a user may provide their house number and postcode and a directory is capable of looking up the full address. The longevity of profile attributes can also vary from real-time properties, such as location, to long-term properties, for example a user's forename. The owners of identities should be able to modify and correct the data in their profiles. In fact, this is a legal requirement in many countries[1]. However, there is often little or no right to have collected information revoked.

In some cases, individuals do not create their identity; instead it is thrust upon them through the use of cookies or other forms of tracking. Profiles generated using these processes still have a registration (the first request without a valid tracking identifier), authentication (supplying tracking identifiers) and profile population (generally browsing history).

To maintain access to a particular identity, the individual is required to maintain a record of their authentication credentials. This may be a digital record, a physical record or simply remembering the information given at registration. The most common credentials are those of a username and password pair. The username is considered to be a semi-public field which is most often unmasked and in the case of discussion boards and forums displayed publicly. The password is considered the private credential and can usually be changed. In some cases services can force a password to be changed after a certain period of time or for some other reason.

One of the first uses of digital profiles was to personalise websites. However, the advent of e-commerce provided many new problems. Since then, online banking and social networking have added more identity attributes and authentication requirements for users. Identity management not only encompasses authentication to multiple services and the provision and maintenance of identity attributes, but the security and separation of them. Keeping digital identities separated, such as home and work addresses, and ensuring that compromising a single service does not compromise multiple services is another matter to address

[1]In the United Kingdom, The Data Protection Act (1998) and, more widely in Europe, the Privacy and Electronic Communications (EC Directive) Regulations (2003).

for individuals.

Recent work has shown that this final point on the separation of services is becoming hard [47] due to password re-use and centralised identity management solutions. Furthermore, centralised identity management also poses the risk of merging identity attributes, hence increasing security risk.

2.1.2 Identity Providers

Before digital identity, identity providers existed as mainly authoritative public bodies, such as the DVLA in the United Kingdom who provide a driving license. Modern driving licenses include a photo and are, in most countries, considered a valid form of official identification. Other countries also implement identity cards which are also maintained by a central governmental authority. In these cases the information provided by the issued identity documents are trusted by third parties as it came from and known reputable source.

With the advent of digital identity, identity providers were either not distinguished from the service or were provided by a library on the computer system. Logging into a computer system passes a set of authentication credentials to the local identity provider (administered by the operating system) which allows the user to execute programs with the authority linked to the credentials entered. The first networked systems centralised the identity provider to a single computer which all of the linked machines sent authentication credentials in order to verify them.

As computers became more popular and numerous, the security of networking became more important as individuals could intercept the communications and gain access to an identity which did not belong to them. Therefore cryptographic functions began to be used. Due to the now-remote nature of the identity provider, it is important to ensure that the service is communicating with the identity provider they trust. Through the use of cryptographic keys, services are capable of verifying the identity provider.

All these solutions operate when systems are maintained within a single organisation, or the identity provider is trusted by all of the services. However, with the advent of the World

Wide Web (WWW), these properties no longer hold. Instead, services were once again forced to act as their own identity provider, like single, unconnected computers of the past.

For individuals, public authorities are yet to create digital identities. Instead, users are forced to utilise commercial identity providers if they wish to centralise authentication and profile data. This process often occurs within organisations without end-user knowledge. For example, ISPs which provide email, billing, online storage and other services often maintain a single login for a user. This single login is used for all of the services, despite them being implemented separately.

2.1.3 Services

Services are the purpose of digital identity. Users obtain all functionality from services. All computer programs may be considered as services; the identity they are able to access and perform computations upon is dependent upon the user they are running for[1]. Through the use of access control built into the host operating system the program has access to certain files, and functionality of the hardware.

As mentioned when discussing identity providers, services were required to become their own identity provider on the Internet. There is no host operating system for the Internet in order to request a user's files, and no central authority to send credentials to verify them. Instead, an identity needs to be created in another way. With no central authority, credentials must be checked by the service, and profile data must be collected and stored by the service. As may be apparent already, but will certainly be discussed further, this manner of managing identity is far from optimal.

Identity for services is bi-directional; services often have to prove **their** identity and that a connection is secure before a user provides identity information in order to verify herself. The reason for this is to prevent service *M* attempting to masquerade as service *A* in order to gain access to identity information that it would otherwise not be privy to.

[1]it should be noted that programs may be running under a different user to that which a request is being served

Identity attributes are utilised by services to provide personalisation, security and to deliver goods, among other purposes. Therefore it is important that these attributes are made available to the service. However, a number of regulations exist in order to specify details about how identity attributes are obtained, transmitted, stored, maintained and deleted. Complying with these rules and requirements all form part of identity management.

2.2 Motivating example

In the UK, Cabinet Office members are discussing identity management. One such case is detailed in a cabinet office blog[1] where the following is stated:

"The days of creating different user names and passwords for every new website are numbered, thank goodness. There is a strong desire to work collaboratively across the public and private sectors to develop solutions that meet users differing needs. That desire is international."

Identity management is an timely problem simply due to the number of identity interactions required on a daily basis between individuals and services. Therefore, in order to fully consider the effect of identity management one cannot simply consider a single interaction or even a single service. This will be discussed in greater length when considering the evaluation criteria for identity management software. Instead, let us consider a set of identity actions that are either common (performed frequently and generally involve a single service) or intensive (performed infrequently, but generally involve multiple services).

The most common action is to authenticate with a service. Depending on the type of identity management in place this can take multiple forms, which will be discussed later. Registering with a service can be both common or intensive, depending on the number of identity attributes required. For example, users regularly register with tracking services which produce reports on site usage without any interaction. On the other hand, signing up to a store to purchase goods requires many details about payment and shipping. The most

[1]http://digital.cabinetoffice.gov.uk/2011/11/04/establishing-trust/

intensive action is that of modifying an identity attribute. This may be required, for example, when moving house. Services which currently send products, bills and other items to a previous address need modifying.

With these important identity management issues in mind, let us consider the situation of creating a new online store, The Online Bargain Shop (OBS). In order to create this service for users we need to handle a number of different aspects of users' identities. OBS will ship goods to an address which has been provided by a user. The goods will only be shipped once payment has been cleared, which requires payment details to be given. Users have the facility to view the status of their order at any time through the OBS website by authenticating with the same account that was used to place the order.

This single example does not take into account all aspects of identity management. It is vital during evaluation to consider other services, users and other stakeholders when making design decisions for OBS. Let us consider a generic user, *Alice*, for OBS. She does not purchase their items solely from OBS. Instead, she utilises a number of e-commerce stores. Alice also has an online bank account and multiple social network accounts. Alice accesses these accounts through multiple devices, including mobile devices. From the perspective of Alice, creating an account at OBS involves a number of decisions and compromises. Firstly, she must choose how to authenticate, and whether to reuse the authentication credentials from other services. Secondly, Alice needs to decide which devices she may wish to access the account on and how to provide authentication details for OBS to those devices. Finally, Alice needs to decide which attributes to provide to OBS, and how to ensure that they are maintained. These decisions will be discussed in-depth in Chapter 3.

Trust is an important part of identity management. It is especially important if identity attributes are to be automatically delivered to services. This thesis only considers the situation where a user is in full control of which attributes are delivered. Further discussion about trust and identity attributes are noted in multiple papers [38, 21, 69, 61]. The primary aim in regards to identity attributes is to make them available to be shared, and not to share them automatically. Additionally, providing a means to update identity attributes is also important [51].

Now that the problem space has been defined, the following section will discuss the field from the software engineering perspective.

2.3 Identity management and software engineering

In the year 2000, a number of software engineering researchers developed a set of roadmaps for software engineering. Roman et al discussed the importance of software engineering in the mobile era [55]. Devanbu & Stubblebine concentrated on the security aspects of software engineering [22].

> "Low level protocols, personal communication appliances, and web content delivery have been some of the most visible elements of this new computing arena. The success and popular acceptance of this technology is accompanied by rapid growth, increased demand for novel applications, and high expectations with regard to quality and dependability. The time has come for the software engineering community to embrace mobile computing as the next frontier to be conquered."

- Roman et al [55]

Despite the decade of development, these words still hold true in the area of identity management. The roadmap discusses the early stages of mobile access and synchronisation for business users who, at the time, were the major users. Now with the costs ever decreasing the ubiquity is increasing. As such, the problems with security increase.

> "Customers demand 'single sign-on', whereby a user authenticates herself once using a specific mechanism and then gains access in a uniform manner to different services (perhaps on different platforms). Developing uniform services that span different platforms is a research challenge."

- Devanbu & Stubblebine [22]

Work on the area defined by the last sentence has been mostly ignored in the software engineering field in the subsequent decade with research focused on the problems faced by services and organisations. Despite this, identity management on the whole has become an important research topic in other areas of computer science. Miller outlined in two articles the development of identity management, the problems and current solutions [47, 46]. An interesting point is that both service providers and users benefit from the enhanced availability of user information. The use of user information is incredibly valuable for one of the Internet's greatest income generators - advertising. The privacy issues raised by this are substantial. An important assertion is that there is a limit to user benefit before it is outweighed by privacy loss. Further to this observation, Miller also notes that redundant storage of profile information can cause it to become outdated very quickly. To conclude, Miller states the use of open APIs reduces the profile redundancy. The key to privacy is ensuring the use of attributes is understood by the user and that the systems adhere to the defined usage. In order to verify these assertions, a study was made where users were prompted to release numerous profile attributes, and many did so freely. Enhancing the users' ability to understand what information is revealed, and its purpose is gaining importance with the rise of social networking.

2.4 Contributions

This thesis develops the field of identity management in two areas: *evaluation* and *development*. Evaluation of identity management architectures permits a high level overview of the benefits and design decisions/trade-offs between different identity management schemes to be evaluated before implementation. It also permits software architects to evaluate whether a change of identity architecture on an existing system would be beneficial. Through an objective evaluation technique, development can be focused on improving identity management architectures. Requirements may also be matched against the abilities of different identity management schemes. To this end, four major novel contributions are made within this thesis:

1. Development of a systematic methodology and metrics for evaluation of identity man-

agement architectures, specifically ensuring all stakeholders are considered, including the end-user, service and identity provider

2. Evaluation of existing identity management architectures, specifically a service based implementation and an identity provider based implementation

3. Design of a new identity management architecture, namely device based, making major improvements to usability while maintaining or increasing security and privacy over existing approaches

4. Two reference implementations of the above architecture.

These four contributions are discussed in further detail within the remainder of this section.

2.4.1 Framework for evaluation

Various evaluation frameworks exist within the software engineering field. These contain no particular focus, and instead guide experts in performing evaluation. Further to this, evaluation of identity management has been performed with emphasis on particular attributes. Both these areas of existing research are lacking in their complete view of the architecture. Instead, an approach that is capable of evaluating all aspects of identity management architectures is required. This combines the multi-attribute trade-off analysis provided by the generic approach with the expert knowledge of identity management specific evaluations. Unlike the generic approaches, identity management requirements form the basis of evaluation, meaning the system requirements are not needed before evaluation. Combining the generic evaluation approaches with identity management specific evaluation has not been attempted to date. Through this union, the technique produced will simplify evaluations and provide a more deterministic result, permitting non-specialised engineers to develop reports in order to choose an appropriate identity management architecture. An additional benefit of the process is providing a benchmark for future improvements.

The problem with existing evaluation has become pronounced due to the differing perspectives that exist within identity management, as described in Section 2.1. A generic evaluation identifies the issues which are important to the stakeholders who are performing the evaluation. For example, service providers consider development time and cost when considering identity management issues. On the other hand, users are interested in the speed, privacy and ease of use. Additionally, service providers and users are not necessarily experts in any of these criteria. Consequently, determining the different variables of identity management frameworks is important in order to ensure that oversights are not made. This will also aid those who make the decision to be aware of any trade-offs that they are making.

As with most implementation issues, evaluation of identity management for a service is a set of trade-offs. There are a wide variety of different problems to overcome and compromises that must be made. A simple example is creating an online store where users email their credit card details, and products they wish to purchase, to the store. This solution is very simple to implement; however the user experiences poor usability and security is non-existent. Additionally, users have no way of tracking their order online, and, without further effort, the store is unable to derive any kind of statistics, such as tracking popular items. Understanding the trade-off in this particular situation needs limited expert knowledge. However more subtle instances exist.

In order to address this problem, a framework for systematic analysis of identity management utilised by a service is required. The framework given in this thesis provides the ability to address a majority of key design and implementation issues and evaluate the trade-offs made between them. The framework also provides a number of metrics which provide guidance for non-expert evaluation. This is an important feature due to the diverse nature of identity management systems and stakeholders.

The framework is aimed at software engineers who do not necessarily have a large knowledge of identity management. Its purpose is to provide a simple method such that the research into identity management may be consolidated and utilised. The output of the evaluation provides a simple high level overview for non-engineers to view the differences and trade-offs between different architectures. Also, from the high-level it is capable of providing evaluation long before implementation specifics have been confirmed.

2.4.2 Evaluation of existing architectures and implementations

To generate the evaluation framework, two studies were performed: a classroom experiment utilising an existing generic architecture evaluation method (ATAM [41]), and a systematic literature review of existing identity architecture evaluation methods. Neither study has been performed in existing available literature to date, and hence it has not been combined either. The experiment was performed utilising a number of security software engineers in order to perform a trade-off analysis of two existing identity management architectures, one service based, the other identity provider based. The systematic review utilised a number of the most popular scientific publishers in order to obtain an deep understanding of the work to date in the field specifically targeting identity management involving Internet transactions between services and users. None of these has utilised a software engineering trade-off evaluation method. Work pertaining to organisation-wide identity management was ignored due to this subject having been mostly solved by industry.

Utilising the framework developed through the evaluation, the two existing architectures were re-evaluated. Comparing the results to those of the classroom experiment, it is possible to determine whether the methodology is an improvement over the generic approach. The results also provide an insight into the trade-offs performed and the causes for said trade-offs. Utilising this data, architectures which are capable of improvements in certain areas may be suggested.

2.4.3 Design of a new identity management architecture

As mentioned, the evaluation of existing architectures provides a starting point for developing novel identity management architectures which give credence to a particularly important variable. In the case of this thesis, the chosen variable is usability. However, some variables must remain at a minimum standard such as security and privacy. Increasing usability means utilising a single identity provider for all (or most) services, therefore minimising the credentials required. This approach decreases the security and privacy afforded to the user and is therefore unacceptable using existing identity management frameworks. Through the high-level analysis of identity architectures, the separation of trust from the identity provider is suggested in order to regain the security, and dependent on the specific implementation, the privacy also.

2.4.4 Implementation of new identity management architecture

The architecture for identity management defined in Chapter 7 is novel and, in theory, provides improvements including usability and security. In order to evaluate whether these improvements may be utilised, an implementation is required to validate the results. Two different implementations are designed in order to complement this framework. One adheres to the framework strictly, providing authentication through the use of digital signatures. Devices may be added and removed from the list of authorised signatories, verified by a simple service or hardware hosted at the identity provider. This approach has the advantage of security against malicious or compromised identity providers. It utilises a large amount of cryptography and bandwidth. The second implementation modifies the trust model slightly, trusting the identity provider to manage devices in the identity, but still securing authentication and profile attributes from the identity provider. This provides much better performance. Both implementations utilise existing cryptographic methods, the first making use of group signatures, and the second, a Diffie-Hellman based key exchange. Neither of these have been utilised for user-service identity management in the past, despite being utilised for device authentication - group signatures famously being utilised for attestation on Trusted Platform Modules (TPMs) and Diffie-Hellman for pairing Bluetooth devices. Both of these existing applications are forms of identity management, but are not designed for authenticating an individual with a service.

2.5 Conclusion

This chapter has walked through the definition of identity management and the need for it in the modern world of computing. The different components that make up an identity management architecture have been defined and discussed. Finally, the contributions of this thesis have been outlined and discussed. Currently, only a few concepts regarding existing identity architectures and research have been discussed. The following chapter will take a more methodical and in-depth review of the development of identity management and the research performed.

CHAPTER 3

THE EVOLUTION OF IDENTITY MANAGEMENT

Each man in his time plays many parts.

— William Shakespeare

There are a number of forms of identity provider based (IdPB) authentication:

- Companies and organisations which utilise many different services may choose to administer all their users on a central identity provider. The administrative load is greatly reduced, and role changes are instantly propagated to all services.
- Services may wish to allow other services access to user profiles or functions through an API. In order to secure user data, they require the user to authenticate. This kind of access allows users to authenticate with their credentials for one service and use the facilities of another.
- User centric identity has evolved mainly from the company based central identity provider. In user centric identity the identity provider which users use is not forced upon them; they are able to choose. This has the disadvantage that it is more complicated to implement and user identifiers are generally longer. Allowing the user to choose their identity provider gives much more freedom, and even allows them to maintain their own identity provider, should they so desire.
- InfoCards are cryptographic certificates containing user details which are signed. These cards may be collected by users and used to authenticate. Users may provide verified profile attributes which are signed by trusted entities. This opens the possibility of anonymous authentication, where only attributes such as age are verified and the exact user is left anonymous.

This chapter details the evolution of identity management through these forms of authentication. The matter of profile collection, storage and distribution is also discussed.

3.1 Roles, research and implementations of traditional identity management

To start with, there was paper based identity, there was nothing digital to need securing. Still today this identity is used, mostly by a select few authorities, mainly governments and financial institutions. Other paper based communication is also used as identity, such as letters from well known companies being used as proof of address.

When computers first existed only physical security was needed. There was no way to access a computer remotely, and if you were in the same room then you could program it to do your bidding. Remote access terminals were developed and it started to become more important to segregate work between different users. Authentication was created where a user input a set of credentials to gain access as themselves. This became increasingly important when computers were able to communicate over telephone lines. At around the same time, the storage on computers became more generic and files were invented. Securing read or write access to files to a certain user or group of users had many business benefits.

The divergence from a centralised computer to multiple systems became a major issue for administration of large networks. Authentication within companies became the first real evolution of digital identity management. Rather than having individual users on individual computers, a central authentication authority is used. Protocols such as Kerberos are used to provide authentication for users. This reduces the administration workload for adding and removing users as well as permitting users to use computers on the network. With centralised authentication, management of user roles may also be performed. By grouping users into roles they may be permitted to do certain actions, such as read or write to specific files or folders, execute certain applications and so on.

A common theme of this thesis is to discuss the issues of “traditional” identity management, otherwise referred to as service based (SB). This architectural form consists of a single service interacting directly with the end-user, and follows a similar interaction pattern as users authenticating to a central machine in a business. All authentication and profile management functions are implemented as part of the service. There are numerous disadvantages to this system from the perspective of the service itself, and from that of the user. These include password fatigue, difficulty of profile management and usability, as will be discussed later.

Yi & Kravets approach this problem for ad-hoc networks [68]. Their work utilises a central trusted authority for verification of certificates. However, the use of certificate chaining allows access rights to be distributed without interaction with the central authority. A major consideration in this technique is that revoking the top of a chain revokes the whole chain. In the standard Internet access model of end user accessing a service though a device, this creates an issue for the user if they need to revoke access to a device in the chain, especially the initial one.

The Internet facilitates anonymity, and unless otherwise identified, users are hard to distinguish. Since services need to distinguish users the problem of identity management exists. Camp provides us with a detailed analysis of the transition of identity from paper based systems to those provided digitally [17]. The centuries of evolution creating unforgeable[1] paper identity is yet to be played out in the digital age. Specifically, Camp details how privacy online is being traded for security. This is due to the lack of ability to provide anonymous authentication. For example, one may purchase an item in a shop for cash and the transaction is untraceable[2], however online all transactions are traceable. Camp also notes that identity theft online is much simpler as pure identity attributes are utilised for authentication rather than an unforgeable medium.

Much of the research at the end of the last and the beginning of this millennium into privacy still considered single systems. Myers & Liskov worked on a static checker for determining data flow through a system [49]. This work does not define where information is gathered and distributed to, and is therefore applicable in more advanced identity management sub-components. The idea of considering identity issues in a single application is firmly rooted in the SB identity mechanisms.

In an effort to increase usability, many client side solutions have evolved. From the perspective of identity attributes, many browsers integrate local storage for automatically filling in web forms. A patent filed by Microsoft [29] even describes a method of utilising a remote server to determine what the type of a particular field is. The most difficult part of identity attribute management is determining which attributes to place in which element, as there is

[1]Or at least unforgeable at cost more than it is worth.
[2]Other than perhaps CCTV footage or similar.

no standardisation. Other problems with this technique include the lack of synchronisation between devices, and being unable to automatically update data at different services. For authentication, most browsers support the storage of usernames and passwords for websites. There are also services[1] which synchronise these with a central database so they can be retrieved on many devices. Unfortunately, the use of an external service requires trust in their systems to all services stored on it.

Josang & Pope describe a hardware device for storing usernames and passwords for services [39]. They claim that their approach is better than a federated approach. This is due to increased compatibility with traditional services and the lack of server-side load. However, hardware support is required for the credentials must be transmitted to the user device.

3.2 Evolution of identity management to single sign-on

Moving onwards from the problems of multiple centralised systems, work began, around the millennium, to assist both users and organisations with identity management issues through consolidation. What is referred to in this thesis as IdPB identity management (*IdPB IdM*) was developed. The issue was even considered important enough by 2003 to warrant its own special edition of IEEE Internet Computing [12]. Companies and organisations which provided multiple services to their users, centralised the authentication and access control systems. After a short time open protocols and implementations emerged such as Shibboleth [24]. This particular implementation is widely used to allow users in educational institutions to access articles via an institutional subscription. This eases the extensive problem of allowing thousands of users access to external resources, provides anonymity of service access to institution level and greatly aids the journal provider in managing subscription access.

SAML (Security Assertion Markup Language) [18] is an XML based standard developed by OASIS. SAML provides the ability to make specific assertions about identity attributes which are signed by identity providers, or provide proof of authentication. This makes SAML

[1]such as LastPass `http://www.lastpass.com/`

versatile, permitting both conventional and anonymous authentication techniques. Users are able to make assertions about a particular value in their identity such that the receiver is able to prove that the assertion is correct and the attribute is valid.

Fujiwara et al. improved upon Shibboleth to allow it to make security assertions [27]. This approach is very similar to that of SAML, although they included the ability to assert a query as unanswerable. This modification improves the privacy of their system over SAML by permitting the user to choose against sending certain information.

Another attempt at improving Shibboleth was made by Takagi et al., who utilised the millionaires' problem and oblivious transfer [64]. This solution is capable of providing a more trusted assertion than that of [27]. Another solution by Bangerter et al.[6] permits the use of certified release. Although not specifically proposed for use as an extension to Shibboleth, it is not hard to conceive using certified release as such. This solution is capable of making inequality and equality assertions about attributes within a certificate without revealing any of the certificate attributes. Additionally, the certificates may be checked for authenticity.

The privacy preserving solutions above maintain user privacy by limiting the information given to services. Another approach is to limit the use and/or length of time an attribute may be stored. Squicciarini et al. propose a solution in this manner specifically for federated services [60]. Attributes are given particular abilities, such as how long they may be kept before deleting them, and whether they may be shared with other services. The problem with such a system is ensuring the qualities expressed have been adhered to.

Pfitzmann & Waidner studied the following browser based techniques: Shibboleth, SAML, Microsoft Passport, IBM e-Community Single Sign-On and Liberty Alliance [52]. Their paper looked specifically at the security and privacy issues relating to this technique of identity attribute distribution. Following this evaluation, a set of design decisions are derived. In order to provide security, they conclude that public key cryptography must be used, and attribute destinations must be authenticated. In order to provide privacy, users must be able to utilise flexible privacy policies, and requests should utilise server session data in order to withhold browsing information to attribute providers. The use of multiple identity sources is noted as a key factor in privacy such that identity providers are only privy to information relating to their identity area. This leads into user centric identity management which will be discussed in the next section.

Siddiqi et al. detailed a framework in order to provision services in a global network [59]. Their framework utilises a centralised single sign-on system, such as Shibboleth or Liberty, and includes an intrusion detection system. The intrusion detection system operates as a middleware which validates identity operations. User phones are utilised as trusted devices in the identification process, with their phone number being used as a unique identifier. This system farms out identity to third parties, but still requires a central identity provider to manage the system and operate the intrusion detection system. Outsourcing the identity may be a major flaw since lost phones mean lack of access, and the requirement of being able to transfer an identity to a new number may be a cause of complete digital identity theft.

Other mechanisms have also been developed to preserve anonymity. A mechanism for subscription based access was developed by Ramzan & Ruhl [53]. They describe two methods to prove that a user holds a valid subscription to a service; however the particular subscription cannot be determined. Both of the methods limit users to a certain number of service accesses, and are not capable of providing a time based subscription. The idea was developed by Backes et al. who utilised certified data release [5]. Their method maintains unlinkability between sessions, but permits time based subscriptions. Further work on the topic, where a token is used to authenticate on each access is provided by Blanton [9]. As with [53], it is possible to prove the service subscription, maintaining the anonymity of the subscription holder. A user may not even be identified between multiple requests in the same session. The disadvantage of this system is the speed of the cryptography used, which is slow on a desktop computer (~1.5 seconds) and unusable on mobile devices. Another similar implementation uses controlled release of certified attributes to perform the same functionality [5].

3.3 User centric identity management

User centric identity management first started to evolve due to the increasing demand of users for privacy, and *control over their digital identities*. Unlike the two previous sections, this evolution regards the way in which the architecture is implemented, rather than the model that it follows. User centric identity is an iteration of IdPB IdM, rather than moving the authentication to another system component. As mentioned in Section 2.3, Miller described the most important issues to be that of maintaining profile data and managing authentication rather than that of privacy [46].

Essentially, user centric identity consists of open APIs, such as OpenID [54], which permit the user to choose her identity provider. A summary of the current state of the art is provided by Johansen, Jørstad & van Thanh [37]. Due to the open nature of these APIs, services are able to communicate with different identity providers. Hence, the user is back in control of her profile. A major disadvantage is the lack of trust in the identity provider by the services. One can argue that this is no different to the trust placed in the user, however this still leads to limited benefit to the service provider.

OpenID is simply an authentication protocol, which is only part of identity management. Profile management is another major issue. Two different systems have been developed for OpenID to this end: OpenID AX [33] (Attribute eXchange) and OpenID SReg [1] (Simple Registration). Both technologies attach themselves as payloads to the OpenID request and response. AX is extensible with a list of global types[2] given by an XRI[3] which may be expanded upon or even ignored by a particular implementation. SReg is made to be much simpler and has a set of predefined types in the specification. However, the use of two competing methods just increases implementation complexity. Both systems are one way and permit the transfer of profile data from the identity provider to the service. Identity providers cannot correct, or request the removal of attributes after they have been passed.

Cross domain single sign-on systems which provide user centric identity are more prone

[1] `http://openid.net/specs/openid-simple-registration-extension-1_0.html`

[2] Previously `http://axschema.org/`, superseded by `http://openid.net/schema/`.

[3] eXtensible Resource Identifier, the generic form of what was originally a URL.

to attacks due to the passing of data between different networks. An attack on OpenID is described by Oh & Jin which is capable of gaining access to user accounts through the use of replay attacks [50]. Although this particular attack is invalid on later versions of OpenID, it raises the awareness that a larger set of security problems need to be considered in such distributed implementations.

Authentication in these scenarios refers to users accessing services. A use case which has not been considered is that of service A using service B on behalf of a user. This particular problem is solved by the OAuth specification [3], which permits users to grant third party services access to another service. By providing this functionality, users are no longer required to give their authentication details of one service to another. This allows a user to withdraw access to a single service, leaving their own authentication details and the access of all other services unchanged. Despite the obvious merits of this system, service access authentication is not considered further in this thesis, as we are concerned only with user authentication and profile management. OAuth is compatible with any other authentication technique used for API access only, and is not a stand alone identity management solution.

The PRIME project was a European initiative to perform research into the future of identity management [16]. The project was summarised in two papers, one describing the project and its successes [45], the other describing the framework produced [26]. Its results were in the areas of human computer interface, usability, privacy and security. In a paper relating to the HCI aspects, Pettersson et al. describe a TownMap in order to manage identity attributes [51]. This view permits users to see all their identity attributes and where they are stored. The problem with this solution is the size that the map would need to be for most users as the number of services they use is so great. Other enhancements suggested in this paper concern terms of sharing attributes. Previous work by Agrawal & Srikant showed users to be more willing to disclose information where privacy policies are in place [1]. It is suggested that a simplified and full version of the terms is made available in order to simplify a users understanding of what a particular attribute will be used for and for how long it will be kept. The authors note that the terms should not be displayed until both (the full and abridged versions) are cached locally, so as not to put users off reading the full version. On a mobile device, this may be futile due to the screen size. The true power of identity attribute management is centralised management, as hinted by the TownMap idea, however no solution for centralised management is proposed in PRIME.

In the previous section, the use of an authority to certify user attributes was discussed, namely in the form of the implementation of InfoCard. User centricity has not ignored this area of certified information. A project developed under the name Higgins [1] provides a unified framework for InfoCards. This solution is often referred to as an Identity Meta System since it operates above the identity attribute distribution (in this case the "cards" themselves). These systems are classified as user centric due to their ability to federate attributes together rather than allowing the use of an open identity provider. Inherent problems occur when card signatures are not trusted by services, but this is no worse than other user centric implementations.

Autonomous sharing based on contexts was also introduced in the era of user centricity. Hong & Landay developed a system which was capable of being told which attributes may be shared and re-distributed [35]. This is a relevant topic in the world of social media. Trusted nodes are used to notify the owner of any unauthorised distribution. This does not provide any protection or revocation abilities post notification. Similar work was performed with sharing files between mobile devices based on the current context by Hakkila & Kansala [32] and by Kalyani & Adams using the XPref language [40].

In a similar area, Cheng et al. provide a solution for services to obtain user attributes [20]. Users are able to share information fully, partially (similar to [6]) or not at all with a service, based on a particular reward for doing so. For example, sharing payment details permits you to purchase an item, or sharing your email address enters you into a prize draw. By assigning a purpose to the identity attributes, the work helps users understand why they should provide certain attributes, developing the themes set out in PRIME.

Another approach taken by Yelmo, Trapero & Alamo is to link together different identity

[1] `http://www.eclipse.org/higgins/`

providers [67]. Users maintain separate identities on multiple identity providers, however users are able to combine identities together in order to share information between them, similar to OAuth but for identity providers. Yelmo et al. utilise computer readable service level agreements (SLAs) and SAML. Due to this it is possible for identity providers to ensure that SLAs are followed by the SAML requests.

The most recent development has been made by Mozilla, who have created BrowserID [1]. This system utilises a user's email address as authentication with the user managing their authentication with their email provider separately. This is a similar solution to that posed in [59] where a mobile phone is utilised. However, unlike mobile phone numbers which may be ported between providers, users have to re-create their profile each time they change their e-mail address. This solution also only considers authentication, with profile storage remaining the same as with SB systems.

3.4 Conclusion

In this chapter, we can see the development of identity has been driven from the perspective of the service provider and the identity provider. The service provider has driven developments in identity management due to necessity. In the absence of pre-built solutions, services providers must provide unique solutions, all of which have their own specifications, user requirements and flaws. Where multiple services are hosted by a single service provider, a certain amount of unification may be utilised, however in the large scale environment of the Internet, this is a rarity.

Unification of authentication has begun with large service providers with many users creating APIs for other service providers to utilise the pre-existing userbase. Since there is no service provider used by every user, this is ultimately futile. Instead, the development of open APIs and user-centric identity management not only permits users of these different services providers (where the API is supported) to authenticate, but users may setup their

[1] `http://www.browserid.org`

own identity providers. Despite these advancements, backwards compatibility and profile management is rarely considered.

The following chapter will consider the these areas in more detail, identifying the different stakeholders in identity management and their unique requirements. A common scenario is also introduced in order to develop the requirements in respect to an objective example.

CHAPTER 4

AN ATAM EVALUATION

To know what has to be done, then do it,

comprises the whole philosophy of practical life.
— Sir William Osier

We will start our evaluation of identity management with the use of a generic method for architecture evaluation. This will provide a baseline of what existing methods are capable of providing in the way of evaluation.

4.1 Performing an ATAM evaluation

Previously (Section 2.4.1), the importance of trade-off analysis was highlighted for IdM architectures. To determine the fitness of existing architectural evaluation an experiment was carried out. A well known trade-off evaluation method known as "Architectural Trade-off Analysis Method" or ATAM [41] was used. ATAM is designed to determine the risk points and trade-off points in architectures. The process is intended to be performed by a group of experts in the areas to be evaluated, we use a controlled class room experiment to simulate the process, applicability and meaningfulness of the metrics/method. Nine groups of Masters students took part, each with four members enrolled in MSc Computer Security or MSc Internet Software Systems. Students on these courses had the necessary preparation and the solid background on distributed Internet scale software architecture, requirements, Internet security and Software Engineering. Many come with some industrial experience in relevant subjects. Students were taught ATAM as part of their course and were introduced to various IdM architecture styles as part of a group project coursework with one month duration. A jury of three security and software architect experts/academics were used to mentor the process and evaluate the end results.

ATAM is meant for use in business environments and therefore had to be adapted slightly for this case study. The stages of ATAM are described as:

1. Present ATAM
2. Present Business Drivers
3. Present the Architecture
4. Identify Architectural Approaches

5. Generate Quality Attribute Utility Tree
6. Analyse Architectural Approaches
7. Brainstorm and Prioritise Scenarios
8. Analyse Architectural Approaches
9. Present Results

Presenting ATAM is performed in order to ensure that all parties are aware of the procedure that will take place and their expected contributions. The students were already familiar with the procedure as part of their course. The remainder of this section will provide an overview of ATAM for the reader.

Presenting the business drivers is the action of determining which particular elements are especially important to the stakeholders. The issue with performing this stage is that the stakeholders are numerous. The service provider has an obvious stake, but also their drivers are well defined and well stated. A single business driver from the service providers' point of view causes a major issue, namely, the wish to please service users in order to gain and maintain their use/custom. As such, the service users become major stakeholders. However at this point they are unknown, and have possibly conflicting drivers.

The approach taken to resolve this conflict was to instruct half of the group to take on the role of the service provider, and the other half to present the drivers of the users. An interesting aside to this technique being performed in the evaluation is to determine how well this method of presenting user opinion works in ATAM.

Guidance on the areas to consider business drivers was given. The list is detailed below, however it was made clear that business drivers did not need to be restricted to these areas.

- Ease of use. The objective of drivers in this area is to ensure that users spend less time managing their identities and more time using the service. This should be considered for registration, authentication and maintainability.

- Computational overhead. It is important to consider how much computational work is required for both the end user and the service provider. If computational overhead increases then a number of detrimental effects may be observed; mainly longer waiting times and higher power usage. From the point of view of the service provider more overhead on their systems requires more processing power, and therefore higher running costs. From the end-user's perspective, long waiting times can cause frustration. More importantly, on mobile devices, increased power requirements affects battery life.

- Bandwidth. Similarly to computational overhead, higher bandwidth use results in longer waiting times as the data is transferred. Obviously, this has more extreme effects on slower connections such as that on many mobile devices. Again, the use of large amounts of data causes service providers higher costs for greater bandwidth and generally mobile devices use more power.

- Privacy. An important driver for the end user is privacy. However, from a service provider's point of view privacy is important for a number of reasons. Firstly, from a legal perspective, service providers are required to maintain a certain level of privacy for data acquired by them about users. Secondly, service providers may want to ensure that user data is not obtained by third parties, especially competing services. Finally, trust in a service is a major usage factor, therefore ensuring privacy of user data increases the likelihood of gaining and maintaining a large user base.

- Security. Often used synonymously to privacy, security has a number of distinct differences. Ensuring that the use of an account is solely provided to the account owner is a security issue. This is not a privacy issue if no profile data is available when access to an account is granted. The need for security on both the side of the user and that of the service provider is, however, identical to that of privacy.

- Availability. The availability of profile attributes to users is important when attempting to share attributes between services. Given a high enough availability, users are not required to re-enter profile attributes which, in turn, increases usability. From the point of view of the service provider this is less appealing due to the inherent privacy issues, although motives come in the form of higher rates of user attribute acquisition (due to the ease of data input) and the possibility of greater numbers of users due to ease of IdM.

- Freshness. The freshness of profile attributes is usually directly correlated to the ease of profile management. A number of identity attributes are constantly being modified, while others are modified rarely, and certain attributes may be static. Examples of these are: credit card details changing on their expiry date (every few years), home address changing when moving house (fairly infrequent) and name changing when married (usually very infrequent). If a service is rarely used, these attributes may be out of date and cause issues, such as refused payment. For the service provider, keeping these attributes up to date is beneficial for marketing purposes and reducing the administration overhead of failed orders or similar problems. For the user, freshness of attributes is helpful, but also introduces privacy issues, hence it should be carefully considered.

- Speed. When a user performs an action within a service, if that action requires the use of an identity attribute then that must be retrieved. Again, a usability issue as different methods of storing and retrieving identity attributes can severely reduce the response time of a service.

Stages three and four (presenting the architecture and identifying architecture approaches) were presented to the groups together, making these constants in the study. As each of these stages is described, the information provided to the students will also be given.

The architecture for the online store was described to the students at a high level, and they were given the opportunity to clarify questions they had about it. An overview of this explanation is given in Figure 4.1. To reinforce their understanding of the architecture they were requested to produce a UML representation of the architecture.

Consider a new Online Bargains Shop (OBS). Retailers and wholesalers can subscribe to OBS to advertise for their online products, promotional deals, updates on their offerings etc. Online Buyers over the age of 18 can sign up to OBS to browse offerings, searching for products, retailers, wholesalers and deals, bid for products, rank and review products, rank retailers and wholesalers, share and discuss reviews, place order(s), pay for order(s), track a delivery etc. OBS can also provide recommendations to the buyers based on reviews, budget, preference information, patterns of use etc. Subscriptions and payments are handled online by a third party consortium. OBS uses the buyers' profiles to disseminate advertisements and promotional offers on their site and by e-mail. One critical security requirement of OBS is managing the identity of users and their profiles (e.g. personal data, credit card details, preference and lifestyle information). If such data is maliciously accessed, disclosed, leaked, or manipulated, it could breach confidentiality and data protection acts. OBS needs to adopt sensible IdM sub architecture, which will be a central element of the OBS architecture.

Figure 4.1: Brief given to students for a system architecture to evaluate identity management solutions.

As mentioned in the brief, the IdM sub-architecture needs to be chosen for the situation. It is this area that is considered in this evaluation. Two architectures, one SB and one IdPB were given to the students to evaluate. Students were instructed to utilise OpenID as their IdPB implementation. This is a well documented and widely available implementation, and therefore was easiest for students to understand and study.

The students were requested to model the two implementations in UML, further concreting their understanding of them. This will also aid in the discovery of communication and processing requirements of the interacting entities within each architecture.

At this stage the groups were asked to perform the final stages of the ATAM evaluation. The second analysis stage proposed in ATAM is used to incorporate the views of stakeholders. The mentor/jury team had acted as the external stakeholders. Other external sources of input included expert views from relevant e-mail lists and ongoing projects on IdM.

Following their evaluation, the results were collected in order to determine whether an applicable and useful evaluation had been performed. The results are presented below.

4.2 ATAM Evaluation Results

The utility trees generate a list of specific scenarios under a list of specific qualities expected of the system. The high-level qualities were determined to be: Performance, Security, Usability, Privacy, Ease of Development and Availability; although not all groups derived all of these qualities. It is important to note that the groups specifically focused on the security and privacy aspects of financial data being processed in the scenario, and justified their use of security as a high level quality for this reason: "security and privacy are essential for online shopping systems as we have to deal with financial data".

Qualities were further refined to more specific qualities, such as "personal data is kept secure", "good authentication", "data confidentiality", "data integrity", "easy authentication", "easy profile management", "updates and support are easy in the future", "transaction throughput". As the final part of the utility tree these qualities were transformed into a number of scenarios which may be tested and evaluated.

These scenarios took three different forms: specific hard requirements ("response sent in less than 5ms"), vague implementation requirements ("authentication credentials are encrypted") or general statements ("profile easy to update"). At this point it becomes clear that deriving scenarios for IdM trade off analysis is a difficult subject. Additionally, measuring them in order to make a comparison becomes a simple matter of judgement. One group concluded that OpenID was "the weakest in terms of security and privacy because users use only one login [...] for all services". This misunderstanding of the purpose of single sign-on demonstrates one of the problems of expert judgement evaluation techniques which do not require strict metrics.

Each scenario was analysed by the groups to determine the risk and assign an importance value to each scenario in the form of low, medium or high. This information will be of use later when deriving the key trade-off points. In this manner, scenarios formed four non-exclusive sets: risk, non-risk, sensitivity points and trade-offs. Risk points determined were: "Identity provider tracking user access", "User loses their password", "User loses their device", "Implementation of service based architecture is costly". Groups determined the non-risk points as a set of statements with regards to all identity architectures providing identity functions. Although this should be obvious, it is useful to remember this point when performing evaluation.

Sensitivity and trade-off points are generated in a similar way to the risk and non-risk points. The sensitivity points describe the interplay of architectural components, these link quality attributes to design decisions. Groups derived the following points:

- Privacy is sensitive to the location of authentication and credential data storage,
- Performance is sensitive to the amount of authentication data that is exchanged among entities,
- Cost is sensitive to the implementation and integration effort required,
- Data security is sensitive to where authentication data is stored,
- Usability is sensitive to the effort required by the user to go through the authentication process,
- Usability is sensitive to how easy it is to remember authentication credentials,
- Security is sensitive to the mechanism used in the system,
- Data protection is sensitive to the standards that are followed by the system.

Similarly, trade-off points were derived to be:

- Increasing encryption will improve the security level, but will increase processing time,
- The use of IdPB IdM will affect user privacy,
- Use of SB architecture is familiar to users, making it easier to use.

The last of these contradicts the evolution of IdM systems and provides justification for research into IdM.

The final step performed was the combination of these different areas of evaluation to conclude the appropriate identity architecture for the system. Since the purpose of this study was to determine the trade-off points and methods of measuring them, these are irrelevant.

The ATAM process is useful for the derivation of very specific scenarios; however an adequate coverage of all areas for consideration is unlikely, even for expert analysis. This is due to the extensive range of areas which need to be considered. The key areas identified for evaluation were:

- Performance,
- Usability,
- Profile freshness and availability,
- Development,
- Security,
- Privacy.

A noticeable problem with the evaluations was the high level of subjectivity and results tend to be limited to the views of the participants.

From these results it is possible to derive some additional evaluation aspects above those produced in Section 5.1. However, the lack of specific expertise in all areas has caused the results to be spurious and contradictory. This needs to be accounted for when utilising this method for evaluation in general, and will be considered in the creation of a combined evaluation approach in the following section.

4.3 Conclusion

The study derived a number of metrics for the evaluation of IdM architectures. Out of the six groups that took part in the study, three chose OpenID and three chose the service based approach. Although this is a small group, this 50-50 split shows a distinct issue with the generic approach. There is no preference toward either of the architecture types for a given scenario.

The problems utilising ATAM motivates us to determine a new method of evaluation, specifically designed for identity management research. In order to produce a methodology which is suitable, distinct metrics need to be found, which are able of capturing all aspects of identity management. The measurement of these metrics also needs to be systematic in order to provide deterministic results, independent of the evaluator.

One of the reasons for ATAM failing to provide a distinct result, is the lack of, or different interpretations of system requirements. A new methodology which does not operate on the requirements of a specific scenario, not only removes bias from the evaluation, but also makes it reusable. Once the evaluation has been performed, the trade-offs between metrics may be evaluated against the design decisions, in order to determine the most suitable architecture.

ATAM also fails to provide any pointers as to how certain properties change, with changing architectures. Having this feature allows engineers to perform educated modifications to identity management architectures, evolving new architectures which give rise to different qualities.

The discussion above motivates us to look for a better method of evaluating identity management, and its wide range of qualities. In the following chapter, we will look at existing research into identity management specific evaluation techniques, in order to derive a new systematic evaluation approach. The evaluation performed in this chapter was performed at a relatively low level. In the remainder of this thesis, evaluation will be aimed at providing results at a higher level. This is to permit the evaluation of identity management architectures before the design stage. Therefore, metrics such as ease of development are not applicable as the method of implementation is unlikely to have been determined.

CHAPTER 5

SYSTEMATICALLY EVALUATING IDENTITY MANAGEMENT ARCHITECTURES

I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you can not measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind

— Lord Kelvin

It is evident from the work described in Chapter 3 that development of identity management (IdM) has been far from static over the past decade. Many solutions have been proposed for many different issues. The driving force for these improvements are from the commercial perspective, attempting to simplify user and access management, although there have been some developments have been made with the aim to develop user privacy and ease, and address the problem of ever increasing authentication details and distributed profiles. The main focus in previous research has been toward developing privacy, such as the recent work of Suriadi [63]. This work develops the idea of federated IdM, bringing user-centric ideas to the profile.

From the perspective of a software engineer, this field has seen little development. Quite clearly research has taken place to determine important factors in IdM. Glässer & Vajihollahi define most of these concepts in [28]. However, a review of these areas including determining the trade-offs between requirements has yet to be performed. As discussed in Chapter 4, generic evaluation methods are lacking when it comes to evaluation of identity management. In this chapter, we attempt to develop a better identity management specific methodology. As with all architectures, trade-offs are a major factor in selection of an appropriate solution. Additionally, it may be argued that such trade-offs may be used to determine the focus of further developments in the areas.

In this chapter a systematic review of existing IdM architectures and frameworks is performed in order to identify the metrics and methodologies used. By performing this analysis, a full picture of the field of IdM will be built.

Combining observations from the classroom experiment performed in Chapter 4 and those from the review of existing literature, a methodology for evaluating IdM architectures with a set of metrics is described. In order to evaluate the methodology and metrics they are used to re-evaluate the identity architectures in the classroom experiment. By following the method provided in this chapter, evaluation of IdM becomes a more objective process. In turn, IdM researchers are able to determine areas where improvements may yet be made. Trade-off analysis and expert knowledge is included in the framework by combining existing evaluation methods.

5.1 Systematic review of existing identity management architectures

The systematic literature review was motivated by the lack of specific guidance on metrics and criteria which security software architects require upon evaluating IdM architectures. In particular, the systematic review seeks to provide answers to the following research questions: (1a) What are the requirements of an identity management architecture? (1b) Which metrics need to be evaluated?

5.1.1 Research method

Compiling this body of work, very little in the way of architectural evaluation for identity management was noted. To ensure this was due to a lack of work, rather than a lack of discovery, a systematic review is required. A methodology for systematic review is presented by Kitchenham [43]. This is a repeatable process for locating past research in a particular area (in this case IdM architectures and frameworks). In order to perform this review the following steps are observed:

1. **Plan**

   The objectives of the research are laid out and the sources are determined.

2. **Search**

   Criteria need to be described in order to locate the papers.

3. **Filter**

   Exclusion criteria need to be defined to refine the search results.

4. **Collate**

   Once the papers have been chosen, the information contained needs to be collated into a form which is suitable for evaluation.

1. Plan

The objective of research was laid out in the chapter introduction. Papers relating to the evaluation of identity management architectures/frameworks are of interest. Specifically, we are looking for work which is providing methods or metrics where evaluation may be performed. The key evaluation areas and techniques for evaluating these areas are of interest in order to develop an encompassing trade-off analysis method.

2. Search

Work in the area of IdM is often performed in industry. However, the work performed in this area is generally geared toward solving a specific issue, or is not publicly available in great enough detail to warrant inclusion in a generic Internet search. Instead, the use of digital libraries of scientific work is the only reliable source of in-depth descriptions of architectural evaluation. Due to the nature of this work, an assumption is made that written literature (books) will not contain any significant additional research that has not been previously published in a scientific journal or paper. Therefore the search is limited to a number of electronic databases.

The databases chosen for this review are as follows:

- ACM Digital Library[1] (`http://dl.acm.org/`)
- IEEE Xplore (`http://ieeexplore.ieee.org/Xplore/guesthome.jsp`)
- SpringerLink [*][2] (`http://springerlink.com/`)
- ScienceDirect - Elsevier [†] (`http://www.sciencedirect.com/`)

[1]This search has the option to include more than the ACM library; this option is not used so as to remove the possibility of duplicate entries.

[*]Springer does not allow you to view more than 500 search results; as such only the first 500 results in the engineering and computer science sections (sorted by relevance) were considered.

[2]Only papers in the areas of engineering and computer science were included.

[†]ScienceDirect does not allow you to view more than 1000 search results; as such only the first 1000 (sorted by relevance) were considered.

These databases were chosen due to the general high impact of their publications in the area of software engineering. The search criteria utilised was “identity management architecture evaluation”. The terms “identity management” and “evaluation” are obvious for the goal of this review. The use of “architecture” in the search criteria is to ensure that the evaluation is being made from a high-level software engineering perspective.

3. Filter

The filter stage contains a number of steps to remove false positives from the search results. These are as follows:

- **Title sift** Due to the nature of the search engines, the terms identity and management are not considered a phrase, and equally architecture may refer to an architecture utilising IdM rather than an IdM architecture. It is possible to remove these by removing any titles which do not reference IdM and those which are referring to architectures other than those of IdM. For evaluation the terms “taxonomy” and “requirements” in titles were hint at techniques within, and are therefore included. From the perspective of IdM, the terms “trust”, “access control” and “attributes” were considered substitutes and therefore also passed.

- **Abstract sift** Often titles do not contain enough information to make a decision upon inclusion or exclusion, and therefore must be studied in greater detail. The same criteria used in the title search is used again, however over the more descriptive abstract.

- **Full text sift** Again, abstracts may not provide enough detail to ensure that the evaluation directly relates to the IdM architecture. By reading the full text it is possible to determine the extent of which the paper refers to the search criterion.

4. Collate

The different approaches in the papers identified will be combined to create a complete evaluation method for IdM. Areas that are not considered by the papers are given consideration. However, these should be given a specific area literature review in the future in order to ensure no papers have been omitted by the generic search.

5.1.2 Execution of search

The search was performed as stated. At this stage a vast number of papers were returned, leading to the conclusion that the search terms could most likely be refined further. The following comments have been recorded in order to aid refinement in future reviews. These comments describe the reasoning behind many of the false positives returned in the search results for each stage of the search process.

- *False positives at title sift*: A number of IdM mechanisms were returned, rather than evaluation of them. Cryptographic evaluation of IdM schemes has been performed, this was not of interest as we are working on architectural evaluation. A number of papers performed evaluation of IdM policies. Finally, non-end user IdM (such as key management) papers were also returned.

- *False positives at abstract sift*: Many of the titles hinted at an evaluation. However this was generally not the case when it came to reading the abstracts and instead the contributions were in a different area.

- *False positives at full text sift*: Some abstracts were too short to determine whether evaluation techniques were provided supplementary to the core contribution. Other papers performed evaluation, but in a manner that did not lend itself to be reproduced in other circumstances.

The ACM Digital Library returned 3906 results in total, refined to 20 after a title sift, 10 after an abstract sift and 3 after a full text sift. IEEE Xplore was much more succinct with the search results, returning just 29. This may indicate a better search process or perhaps a more lossy one. A title sift over these results returned three papers of interest, which maintained their interest through the abstract sift. One of these papers was removed by the fulltext sift, leaving two relevant articles. SpringerLink returned 7406 results, but only the first 500 could be viewed due to restrictions of the search database. Categorisation allowed the first 500 of the Computer Science and Engineering sections to be evaluated, and of these 18 passed the

	Results	**Titles viewed**	**Title Sift**	**Abstract Sift**	**Text Sift**
ACM	3906	3906	20	10	3
IEEE	29	29	3	3	2
Springer	7406	500+500	18	3	0
Elsevier	8700	1000	5	2	0
Total	20041	5935	46	18	5

Table 5.1: Summary of systematic literature review results

title sift, 3 the abstract sift and none passed the full text sift. ScienceDirect returned 8700 results but, similarly to Springer, only the first 1000 results were made available. Of these results, 5 passed the title sift, 2 passed the abstract sift and none were selected after the full text sift. This is summarised in Table 5.1. The five papers left after the full text sift are detailed in the following results section.

5.1.3 Results

Of the 20,000+ papers considered by this search, just five are pertinent to the metrics and methodologies for evaluating identity management. All of which have a very specific focus, rather than taking a holistic view. This lack of directed research is unsurprising due to the focus by funding bodies (e.g. EPSRC, EU, NSF) to motivate research in the area of identity management.

Let us now consider the five papers which were returned. Barisch defines usage scenarios, based on the assumption that users utilise multiple devices for accessing Internet services, and from these derives requirements specifically for authentication [7]. Haidar & Abdullah also perform their evaluation on authentication also [31]. This evaluation is from the perspective of organisations rather than the end user. Pfitzmann & Waidner specifically evaluate privacy for user profile attribute exchange (i.e. user profile data being passed between services and/or an identity provider) [52]. Finally, Suriadi, Foo & Du suggest a meta-framework for IdM systems [62]. This system's proposed purpose is deriving a model where multiple IdM systems may co-exist. They provide a number of metrics to evaluate these meta-frameworks, of which most are applicable to evaluation of any IdM system.

The requirements and metrics in these papers may be compiled into groupings. These are shown in Tables 5.2 and 5.3.

Schell et al. identify a number of system components and utilise them to describe the distribution of functionality and data within a system [58]. We show this in Section 5.2, where system diagrams are created, marking components with letters assigned to their role. They define the performance metric to be the delay imposed by the system containing the "enforcement" component. It is also proposed that use of a simulation framework "OMNeT++" may be used to model the effect of failed components (either by making wrong decisions or going offline) on an identity architecture.

These results will be used in order to derive a more complete evaluation method in Sections 5.2 and 5.3. First, let us reconsider the work of Schell et al. and perform a high level analysis of two IdM architectures, SB and IdPB (specifically, the OpenID implementation).

5.2 High level modelling of identity architectures

The first stage to IdM analysis is the derivation of the architectural primitives in the IdM schemes to be evaluated. This is widely missed by existing IdM evaluations and is capable of describing the component based differences between identity architectures, hence providing reasoning behind many of the exhibited properties.

Schell et al. [58] provide a method for this evaluation through the use of six architectural components which are distributed within the system. Those components consist of *Client*, *Enforcement provider*, *Authentication provider*, *Authorisation provider* and *Provisioning provider*.

The model produced by Schell et al. is not directly applicable to certain architectures where multiple systems share a single component - for example, if threshold security were to be utilised [42]. In this particular case *Attribute* storage is distributed between systems; singly they do not provide useful information. In order to accommodate these, we propose adding links between stores and components.

	[7]	[31]	[62]	[52]
Authentication user simplicity				
Usability		√		
Common interface			√	
Users should be able to understand the actions required of them			√	
Mental and physical actions are tolerable even if performed repeatedly			√	
A common façade to handle operations regardless of the data source			√	
Authentication availability				
Remote activation	√			
Task distribution	√			
Scalability of IdM architecture		√		
Credentials may be retrieved by parties requiring them			√	
A specific method should not be imposed to handle users' credentials			√	
Authentication privacy				
Minimise leakage through URLs				√
Authentication security				
Capture device characteristics	√			
Discovery of devices	√			
Secure exchange	√			
Establish security associations	√			

Table 5.2: Table of groupings to requirements or metrics for authentication from the systematic literature review.

	[7]	[31]	[62]	[52]
Profile availability				
Distributed data handling	√			
A consistent representation of identity information is supplied regardless of the origin			√	
Accept token requests in a number of different protocols			√	
Various token types may be generated			√	
Profile security				
Secure exchange	√			
Authenticate destination site				√
Permit the use of public-key cryptography				√
Profile privacy				
Policy information				√
Flexible privacy policy				√
Hide wallet address				√
Allow unknown attribute issuers				√
Clear information about personally identifiable information used, provided and the related privacy policies			√	
Must be able to handle service, identity provider and user privacy policy expressions			√	
Should be able to understand service, identity provider and user privacy policy expressions in multiple formats			√	
Profile freshness				
Real-time release				√
Query wallet holder				√

Table 5.3: Table of groupings to requirements or metrics for user profiles from the systematic literature review.

The *Client* is fairly self explanatory as a component. By modelling the user as a component within the system their interaction must be defined. Users interact through devices, which consist of mobile phones, laptop and desktop computers and even televisions. The user provides the overall authentication, providing all identity attributes, either directly or indirectly. The purpose of the *Client* component is to describe where the end user interacts with the system. Their device may also provide additional functionalities and and is therefore modelled as a system. For example, the device may be able to store authentication credentials which were either entered by the user, or generated on behalf of the user.

An *Enforcement provider* operates in order to restrict access to a component. It gauges the user's current abilities, and requests authentication for the particular system that it is protecting. It utilises the *Authentication provider* and *Authorisation provider* to determine whether access should be granted. As such, this component's trust is limited to the attributes bestowed upon it, and it has no access to other parts of the user identity.

The *Authentication provider* verifies the identity of the accessing *Client* in order to ensure that it is the same as the one that created the identity. There are many methods utilised to perform this. The traditional method is a username and password combination.

Working with the *Authentication provider*, an *Authorisation provider* determines whether an identity is permitted access to a *Service*. The role is similar to that of the *Enforcement provider* which determines which resources need their access restricting. However its task is to confirm that the identity is permitted to have access.

The *Provisioning provider* exists in order to create a new identity. Depending on the IdM architecture, this may be ad-hoc, require certain pre-requisites or be executed by system administrators.

An extension to the original scheme is the addition of the *Service* component. This provides the content that the *Client* was attempting to access before they entered the authentication process. We add this component so that it is clear where it resides in the architecture.

Three stores are also required, the *Attribute store*, the *Credential store* and the *Policy store*.

The *Attribute store* is where a user's profile resides. Depending on the architecture, there may be a single store, or multiple decentralised stores. These may be indexed centrally, meaning that it is distributed over a number of different types of system.

A *Credential store* keeps track of the credentials needed to verify that the user has access to the service. The credentials may vary between architectures. However they must be verified against this store in order to ensure that they are registered to a particular identity.

If different authentication policies exist (for example, in role-based authentication or where different subscription levels are available) a *Policy store* maintains the relation between identities and their abilities. A policy store needs to be located on a system that is trusted by the *Service*, otherwise its role may not be trusted.

Finally, a pseudo-component needs to be introduced into the architecture; this is the *Trust* component. This component relates to the *Credential store*, *Attribute store* and *Authentication provider*. If the system which contains one of these components is trusted with the contents or ability of this component, this is identified by a *T* within the component. In certain cases, a system may hold information or abilities, but without collaboration with another it is unable to make use of them. This is the situation where a non-trusted system is provided with these abilities. Identifying this component lets us explore alternative architectures by removing trust from systems which traditionally hold it. There are a number of benefits to be obtained by making this abstraction, such as creating a system which has the role of simply providing *Trust*.

The suggested graphical representation for the components decribed above is shown in Figure 5.1.

This initial evaluation provides an overview of high-level differences between architectures. It may be linked to the results of further analysis to assist in future developments and selection of an appropriate identity architecture. A more objective analysis allows us to derive meaning from the high-level architecture, and therefore postulate improvements. These improvements may be used to develop new IdM systems, as demonstrated in Chapter 7. These, in turn, may be evaluated in the same way to verify whether they have contributed to an improvement.

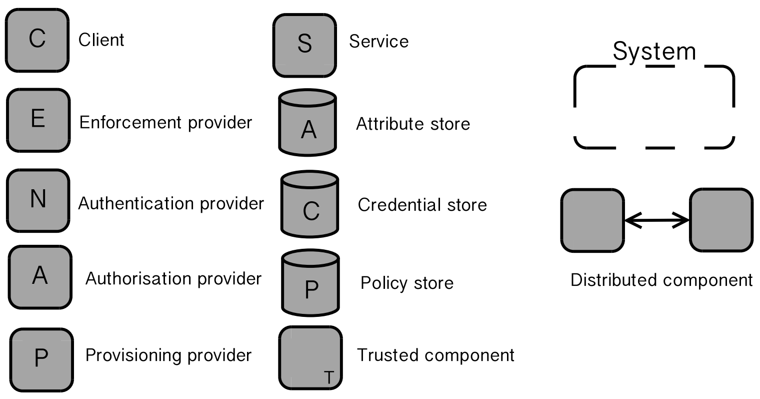


Figure 5.1: Key for high-level architecture evaluation

Documenting this evaluation in graphical form, systems are represented as boxes with dashed lines containing the components which they implement within. Components are shown as either boxes or cylinders, as noted above. Where components are fulfilled by multiple systems they should be linked by a double arrow. This simple overview allows, with minimal description, a comparison between multiple identity architectures and a quick overview of the architectures.

5.2.1 Evaluating by example

Throughout this chapter we will be utilising the example in Chapter 2 as a base for evaluation. Two identity architectures will be used. One is a traditional service based (SB) approach, where the service performs all of the identity management functionality. The other is the identity provider based (IdPB) approach, sometimes known as single sign-on. This utilises an external system (the identity provider) to perform authentication.

The model explained above was utilised for analysis of both SB, shown in Figure 5.2, and IdPB, shown in Figure 5.3. These figures show the high level overviews of the two architectures based on the four distinct components and the two types of attributes described in Section 5.2. As is visible in these figures, the SB architecture combines all of the identity entities into the service, whereas the identity provider and trust are outsourced in the IdPB model, taking with them the credentials. It is worth noting, that although the identity provider and trusted entity are separate, they may still exist within the same service provider. This situa-

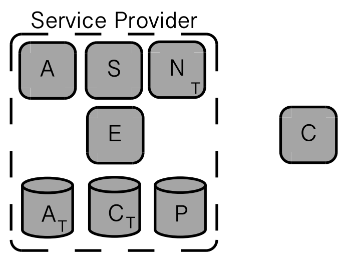


Figure 5.2: Framework model for service based identity management

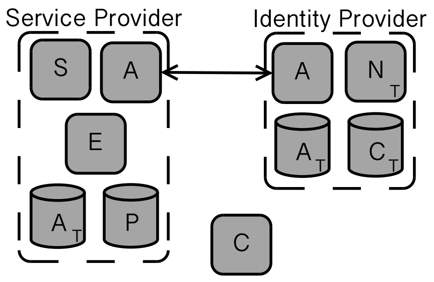


Figure 5.3: Framework model for OpenID identity management

tion may lead to a combined trusted entity for the service and identity provider. This may vary the level of trust placed in this entity by the user.

5.3 Metrics for identity architectures

For a more detailed analysis, a set of quantitative and qualitative metrics are needed. ATAM (used in Chapter 4) requires the use of expert judgement to assess the trade-off between different factors utilising a set of domain specific metrics. Instead, the results from the ATAM evaluation performed, existing metrics defined in the papers identified in the systematic literature review, and the observations made within Chapter 3 are combined in this section to derive a set of metrics for the IdM domain. Dependent on the situation, both qualitative and quantitative metrics are utilised. Where quantitative metrics are available, they may be skewed by implementation issues and therefore a qualitative metric is also provided. Quantitative metrics are also difficult or impossible to evaluate before system implementation, whereas qualitative ones permit early evaluation. Other cases exist where quantitative measurements are simply not possible, such as interface design. The evaluation criteria are split into those for authentication and those regarding the user attributes (profile).

Saaty proposed a similar framework for making decisions (AHP) [56]. His work assigned weight based on the importance of one objective over another, as will be observed in the following definitions on a per-metric level. Saaty produced a single score over all of the areas considered. Instead, a trade-off evaluation of the results is utilised, discarding the combining of results. For this reason, the normalisation against the maximum score gained in that area proposed is discarded. Instead, normalisation is performed against the maximum score possible in an area, taking into account impossible combinations.

To capture the features of IdM architectures, one must ensure that the evaluation method is capable of assessing architectures with variability points. IdM architectures may have a number of features which depend on the specific implementation, or on the other systems with which they are interacting. This is especially true in the case of user centric IdM, where open APIs permit numerous implementations to interact. Therefore it is important to note both the minimum standard provided by an IdM architecture (commonality) and the maximum possible within that architecture (variability). This is not part of AHP or ATAM, but will be a key feature of the methodology presented below.

The groupings provided in Section 5.1 will form the basis of the metrics. However, these groupings still do not give a complete picture of the areas which must be evaluated. There are certain aspects of some groupings that have been ignored and also some groupings that have not been considered. Schell et al [58] note that performance should be evaluated, and utilise their own tool for this evaluation. The use of bandwidth also needs to be considered due to the limitations of mobile devices [4]. Speed of profile access also needs to be measured to ensure that the response time is tolerable for users.

As a final note before compiling the list of metrics, two further areas which have yet to be mentioned should be considered. First, *profile authenticity* has not been included as an area of study in the past. The ability to have verifiable attributes or not is a binary decision, and therefore only needs to be considered when providing a full trade-off analysis. Second, the *speed of profile attribute access* - this is partially covered by the performance analysis mentioned previously. However past work focused on authentication speed rather than the time to obtain profile attributes.

Metric scoring

The metrics are made up of a number of features, for example, a_1, a_2, b_2, a_3, b_3 and c_3. The features are grouped into those which are comparable; here there are three groups, where $b_2 > a_2$ and $c_3 > b_3 > a_3$. Each feature is given a score, similar to AHP. The better the feature that it relates to, the higher the score allocated. Unlike AHP, each feature defaults to a score of 1. If a feature supersedes another feature in the same metric the score is higher. In the example above, b_2 and b_3 are given scores of 2, as they are better than a_2 and a_3 respectively; c_3 is given a score of 3, as it is better than b_2. The ranking is derived from this definition, meaning that the numeric value does not indicate how much better one feature is over another. It merely shows that feature A is an improvement over feature B. Each metric is then normalised to provide a value which is easily compared during the trade-off stage. The maximum score is not simply an addition of all the scores, as higher values may supersede lower ones, making them irrelevant. Therefore the maximum score for each metric is the sum of the "best" features scores. In the example this is the sum of the scores allocated to a_1 (1), b_2 (2) and c_3 (3), meaning, in this case, the maximum score would be 6. Some metrics may be better, but also compatible, meaning that the maximum would be higher.

The remainder of this section is split into two parts: *Authentication* and *Profile*. These subsections contain a list of all metrics identified, and how they may be measured. Each area to be evaluated will be given a score: these are for comparison with the same metric on other architectures and should not be totalled. This is the basis of the IdM trade-off evaluation. To ensure that the scale in each area does not skew the results, they are scaled to score a maximum of 10 points in each. A number of metrics are inverse (the higher the score the worse their performance), these should have their normalised result subtracted from 10.

5.3.1 Authentication

Authentication user simplicity

The number of services used has dramatically increased as the Internet has become more pervasive and with the development of mash-ups [48]. This has given rise to most of the latest developments in IdM, therefore measuring the impact the architecture has on user simplicity is important. Haidar & Abdullah state: "The number of separate identities that a user has to manage has a substantial impact on usability in several major ways" [31]; they go on to state that usability is directly proportional to the number of security domains. Additionally, Suriadi, Foo & Du have a number of usability points.

	Metric	Group	Score
a	User can use a single set of credentials for all services	*i*	1
b	Common user interface on all services (providing it may be used on multiple services)	*ii*	1
c	Authentication credentials are remembered between sessions	*iii*	1
d	No authentication credentials need to be provided by the user (mutually exclusive to *c*)	*iii*	2

Maximum score: 4.

Authentication computational overhead

The use of mobile devices makes the use of complex cryptography very costly. The work of Schell et al ([58]) and the results of the ATAM evaluation point to this metric being required for evaluation of the authentication scheme. This particular metric is a inverse, because the more cryptography that is used, the worse the overhead and should be given the value of 10 minus the normalised result. A reliable quantitative measure may be made by measuring the time required to execute the different authentication protocols on different devices and averaging the result. However, it is beneficial to be able to perform a rough guideline test by evaluating the types of cryptography used. The values given are based on an estimation of the time needed to perform the tasks required for those functions relative to others. The result is simply for indication as the number and regularity of computation is not taken into account until the quantitative evaluation.

	Metric	Group	Score
a	Proof of knowledge calculations used	*i*	3
b	Use of asymmetric encryption	*i*	3
c	Use of asymmetric signatures	*i*	2
d	Use of symmetric encryption	*i*	1

Maximum score: 9. This metric is inverse and should have its normalised value subtracted from 10.

Authentication bandwidth

Similar to computational power, mobile devices often have limited bandwidth, causing slow operation. Again, this may be measured quantitatively by the size of incoming and outgoing data. As above, a qualitative metric is also supplied in order to provide a guideline at the architectural level. The result of this metric requires inverting in the same manner as it is a negative metric.

	Metric	Group	Score
a	Every request requires communication between the device and the identity provider	*i*	1
b	Every request requires communication between the device and another service	*ii*	1
c	Authentication requires communication between the device and the identity provider	*iii*	1
d	Authentication requires communication between the device and another service	*iv*	1
e	Registration requires communication between the device and another service	*v*	1

Maximum score: 5. This metric is inverse and should have its normalised value subtracted from 10.

Authentication availability

Availability is mentioned in three out of the four papers with requirements for authentication ([7, 31, 62]). There is some overlap when converting these requirements to measurable architectural functionality, the result is the following list.

	Metric	Group	Score
a	Multiple identity providers may be used	*i*	1
b	Multiple devices may be used to authenticate with the identity provider	*ii*	1
c	Authentication may be passed between devices within a session	*iii*	1
d	Multiple authentication protocols are supported	*iv*	1

Maximum score: 4.

Authentication privacy

A key role of IdM is to provide users with privacy. The metric determines architectures with components which have higher regard for user privacy. Only Pfitzmann & Waidner made considerations for privacy in authentication [52]. Their work described the requirement for minimising leakage of information through URLs. Through generalisation, and drawing from the results of the ATAM evaluation, the following five functionalities may be determined:

	Metric	Group	Score
a	Personally identifiable information is not visible to service provider	*i*	1
b	Personally identifiable information is not visible to identity provider	*ii*	1
c	Service provider usage information is not visible to the identity provider	*iii*	1
d	Users have the option to be anonymous to the service provider	*iv*	1
e	Users have the option to restrict service providers from linking visits	*v*	1

Maximum score: 5.

Authentication security

An obvious quality of an authentication scheme is its security. The security of the protocol is assumed, as this is a topic for more in-depth analysis than an architectural evaluation. Instead this metric focuses on security that is provided based on the protocol. Therefore, it gives high scores to architectures which have less components which are able to circumvent authenticity of the user. Barish motivates the need for secure exchange and security associations [7]. These are extended with a some additional security features to provide the list below.

	Metric	Group	Score
a	The identity provider is unable to authenticate as the user	*i*	1
b	Authentication alerts the user through a secondary communication channel	*ii*	1
c	Authentication credentials are encrypted during transmission	*ii*	2
d	End-points of authentication transmissions are verified	*ii*	2
e	High-entropy keys are used for authentication	*ii*	2

Maximum score: 8.

5.3.2 Profile

Profile availability

With a large number of services, re-entering and maintaining a distributed profile becomes difficult. It is preferable to distribute and maintain profile data from a central location. Measuring this at the architectural level consists of determining how many different components are capable of storing and distributing identity attributes.

	Metric	Group	Score
a	Consistent identity attribute storage and retrieval	*i*	1
b	Support multiple protocols for identity attribute request	*ii*	1
c	Support multiple protocols for identity attribute response	*iii*	1
d	Identity attributes may be stored on a device	*iv*	1
e	Identity attributes may be stored at the identity provider	*iv*	1
f	Identity attributes may be exchanged between services	*v*	2
g	Identity attributes may be exchanged between devices	*v*	1

Maximum score: 8.

Profile security

If profile data is made available to multiple systems, it is important that access to it is limited. The security of the profile is dependent on the components which are permitted access to it, assuming that the underlying protocol is secure.

	Metric	Group	Score
a	The identity provider is unable to read identity attributes	*i*	1
b	Identity attributes are encrypted during transmission	*i*	2
c	Identity attributes are encrypted in their storage location	*i*	2
d	Verification of destination site is used (for example, PKI)	*ii*	1

Maximum score: 6.

Profile privacy

Similarly to the authentication privacy, it is important to ensure that as little information as possible may be inferred through the profile distribution. Therefore, this metric measures what data is made available, when a service is permitted access to any given attribute. Architectures which make use of protocols that are able to express properties about profile attributes without releasing the data itself, such as [6], will perform well in this area.

	Metric	Group	Score
a	Identity attribute privacy policies are supported	*i*	1
b	Privacy policies are flexible	*i*	1
e	Supports multiple privacy policy formats	*i*	1
c	Source of identity attributes is withheld from the receiver	*ii*	1
d	Identity attributes may be provided without an issuer (certifying authority)	*iii*	1
f	Profile attributes may be sent individually	*iv*	1
g	User may choose which attributes are sent	*v*	1

Maximum score: 7.

Profile authenticity

In certain applications, profile data is used to authorise a user. These situations require that the profile data be verified by a third party which is trusted by the service, otherwise the user may assert false data. This metric determines the suitability of the given architecture to this form of authentication.

	Metric	Group	Score
a	Support for identity attributes to be certified by an authority	*i*	1
b	Support for automated verification of certain attributes	*ii*	1

Maximum score: 2.

Profile access speed

Despite other features, a profile must maintain its usability. A quantitative metric is utilised in this instance, measuring the time required to retrieve identity attributes for a simple registration form (name, email and address). This metric is an inverse metric with higher scores being worse and should therefore be inverted in the usual manner. A qualitative metric is again given to provide a result before implementation, as in previous quantitative measurements.

	Metric	Group	Score
a	Each identity attribute request requires a separate request by the identity provider	*i*	1
b	Each identity attribute request requires a separate request by the user (mutually exclusive to *a*)	*i*	2
c	Identity attributes are stored decentralised	*ii*	1
d	Identity attributes are not stored on the device	*iii*	1

Maximum score: 4. This metric is inverse and should have its normalised value subtracted from 10.

Profile computational overhead

As with authentication, the computational overhead of accessing profile attributes is also important. Again, this metric is best quantified through the use of a quantitative metric. However, the same qualitative metrics used for authentication are re-applied for architectural level analysis. Again, a smaller score is better with this metric, meaning its value should be inverted.

	Metric	Group	Score
a	Proof of knowledge calculations used	*i*	3
b	Use of asymmetric encryption	*i*	3
c	Use of asymmetric signatures	*i*	2
d	Use of symmetric encryption	*i*	1

Maximum score: 9. This metric is inverse and should have its normalised value subtracted from 10.

Profile freshness

Finally, it is important to maintain the profile data to ensure that user data does not become out-dated. For example, if they were to change their email supplier, move house or get a new credit card. By maintaining access to all profile data, users are able to ensure that it is updated whenever required.

	Metric	Group	Score
a	A database is maintained with the location of all profile data	*i*	1
b	Services maintain a link to a repository of user data	*ii*	1
c	A user is able to centrally view all profile data (mutually exclusive to *e*)	*iii*	4
d	A user is able to centrally modify all profile data (mutually exclusive to *f*)	*iv*	5
e	A user is able to centrally view some profile data	*iii*	3
f	A user is able to centrally modify some profile data	*iv*	4
g	A user is able to view all profile data on services (mutually exclusive to *i*)	*iii*	2
h	A user is able to modify all profile data on services (mutually exclusive to *j*)	*iv*	3
i	A user is able to view some profile data on services	*iii*	1
j	A user is able to modify some profile data on services	*iv*	2

Maximum score: 16.

It should be noted that some architectures are capable of providing differing levels of certain metrics depending on circumstances or implementation specifics. In these cases, both the upper and lower values should be noted for a holistic comparison.

5.3.3 Evaluating by example

As with the high-level evaluation, we will be considering the example in Chapter 2 with the SB and IdPB architectures. This will provide both an evaluation of the metrics against the results gained in Chapter 4, and provide an insight into the results of the high-level evaluation performed in Section 5.2.1.

To start, consider the SB identity architecture. Adhering to the metrics laid out in Section

5.3, let us consider the authentication and profile details of this architecture.

Service based

The first architecture to be evaluated is that of the SB. Utilising the same scenario as in the ATAM evaluation, the Online Bargain Store (OBS), the architecture will be considered. Each of the six authentication metrics and six profile metrics will be considered in turn, then graphed.

The maximum score for each metric is shown next to the title in brackets for reference. If the metric is an inverse metric this is indicated by an ‘i’ next to the maximum score.

Authentication user simplicity (4)

Users are required to utilise a username and password combination to authenticate with the OBS. It is recommended that users do not utilise the same password as one used on another service, since a security breach at one service compromises them all. Additionally, restrictions on the length and character make-up of usernames and passwords may be different to other services.

The user interface design is proprietary, integrated into the style of the OBS website.

In order to save user effort, the designers of OBS have the option to save information on the users’ browser, through the use of cookies. These permit them to maintain authentication for future access to the service. However, they may be disabled in the browser. Additionally, many modern browsers are capable of storing authentication details either naively or through an extension. Both of these are options, as the browser may not support either.

The score awarded for user simplicity is between 1 and 2. This is scaled using the maximum, 4, which gives the values between $\frac{1}{4} * 10 = 2.5$ and $\frac{2}{4} * 10 = 5.0$.

Authentication computational overhead (9i)

To provide communication layer security, OBS can optionally choose to utilise a secure socket layer, commonly known as HTTPS. This system utilises both symmetric encryption for data transfer and asymmetric signatures for verifying the authenticity of the remote server. If secure communication is used then this scores 3. Since this is an inverted metric the value is calculated (with 9 being the maximum for this metric) as $10 - (\frac{3}{9} * 10) = 6.7$.

Authentication bandwidth (5i)

Authentication is performed simply by sending the credentials to the service, this simple case scores 0. Being an inverse metric this gives a value of 10.0.

Authentication availability (4)

Availability is an interesting topic for SB. Since authentication is performed on the service that is being used. If the authentication is unavailable the service is also unavailable, making the metrics moot. However, the ability to pass a session between devices is of use to users, but not supported by this form of IdM. Despite it being unimportant in a number of the scenarios, the score is still 0.0.

Authentication privacy (5)

As all information is collected, and stored by the service provider, there is no way to hide information from it. As the role of the identity provider is taken by the service, no more systems need be given any personally identifiable information.

Since users are required to authenticate directly to OBS, no anonymity may be afforded, meaning all access to the service may be logged and linked.

The score awarded for privacy is therefore 3, giving a value of 6.0.

Authentication security (8)

It was mentioned earlier that OBS has the option to utilise HTTPS to encrypt traffic between the two end points. If this option is taken then the authentication credentials are encrypted during transmission. Another feature of HTTPS is the public key infrastructure which verifies the identity of the server, hence verifying the end point of the authentication.

Since there is no external identity provider it is not able to authenticate as the user.

Given that HTTPS is chosen by OBS, a score of 5 is awarded, otherwise only 1 may be given. This gives a value for this metric of between 1.3 and 6.3.

Profile availability (8)

With SB identity, the only option to store identity attributes between services is a browser based solution. Unfortunately, there are no standards for retrieval of particular attributes and a large amount of guess work is required by the browser in order to determine the type of a particular field. The interface is controlled by the implementing service, and as such OBS is unlikely to have their design the same as any other service.

There is no support for exchange of profile attributes or an API for storage of attributes on the user device. This results is a score of 0.

Profile security (6)

As with authentication OBS has the option to utilise HTTPS to secure communications, this decision determines whether identity attributes are secured during transmission. This also ensures that the destination site is verified.

Identity attributes, once at OBS, are accessed by the same system that they are stored on. Any encryption utilised to store attributes would need to be decrypted in the same system, making it redundant.

No identity provider is used for this architecture, and as such it is not able to read the identity attributes.

The score for this metric is between 1 and 4, giving values between 1.7 and 6.7.

Profile privacy (7)

Privacy policies on the OBS system consists of a human readable policy which must be agreed to upon registration. This does not permit automated policy checking. All attributes are provided by the user, hence the source is known but it is not provided by an certifying authority. By filling in the registration form the user has the option of which profile attributes to enter. This gives a score of 3, a value of 4.3.

Profile authenticity (2)

Attributes supplied to OBS by the user are provided through the browser interface and therefore are unverified. However, the user email address may optionally be verified by sending an email with a link in it meaning a score of between 0 and 1 may be achieved. This gives a value of either 0.0 or 5.0.

Profile access speed (4i)

Identity attributes are provided during the registration process in a single form and then

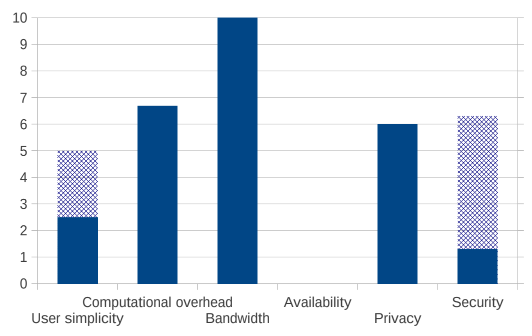


Figure 5.4: Qualitative analysis of service based authentication

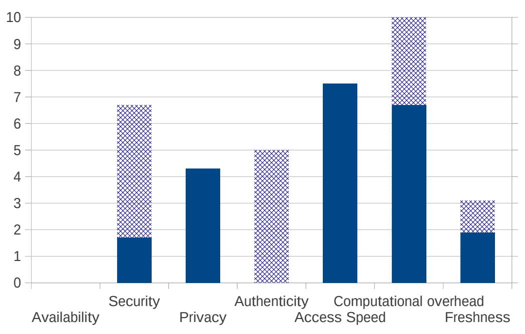


Figure 5.5: Qualitative analysis of service based authentication profile data

retrieved locally. Applying this to the metric gives a score of 1, and being an inverse metric, a value of 7.5.

Profile computational overhead (9i)

Profile access in SB IdM is performed locally, and as such no encryption is required. However, for the user to update the data or set it initially during registration it much be sent to or from the user device. In order to do this securely a HTTPS connection should be utilised. Therefore the score ranges between 0 and 3, and being an inverse metric the value awarded is between 6.7 and 10.0.

Profile freshness (16)

The profile attributes shared with OBS are not recorded by any authority, and hence may only be altered by the user accessing their profile directly through the OBS service. Dependant on design decisions, users may or may not be allowed to modify their profile data directly. This gives a score of between 3 and 5, providing a value of between 1.9 and 3.1.

Graphing

The values above have been taken and plotted onto two graphs, one relating to authentication (Figure 5.4) and one for profile (Figure 5.5). The value is shown as a bar on the graph, and where a range exists this is shown by a hatched bar above the minimum.

Identity provider based (OpenID)

For comparison, an IdPB solution will undergo the same qualitative testing. As before, the same architecture that was evaluated in the ATAM section will be used, OpenID.

Authentication user simplicity (4)

The main advantage to OpenID is the use of a single set of authentication credentials for multiple services. Users are also forwarded to their identity provider's login page in order to authenticate giving a common interface between services.

Similarly to SB, OpenID identity providers can store authentication details as cookies on the user's device, and optionally browser based solutions may store the authentication credentials. This means that remembering credentials between sessions is possible, although optional.

This means that a score of between 2 and 3 may be achieved by OpenID, a value of between 5.0 and 7.5.

Authentication computational overhead (9i)

Due to OpenID's identity provider utilising a similar authentication mechanism to SB the computational overhead is also similar. There is the option to utilise HTTPS on the identity provider. The difference is that since the identity provider's sole purpose is to protect the identity, they are more likely to utilise HTTPS. Assuming secure communication is used then this scores 3, and since this is an inverted metric the value given is 6.7.

Authentication bandwidth (5i)

Authentication in OpenID forwards the user to the identity provider where the user performs their login. Following this, the user is redirected back to the service which verifies the authentication with the identity provider over a back channel. This gives a score of 1, and being an inverse metric a value of 8.0.

Authentication availability (4)

When authenticating with OpenID the user has the option to choose which identity provider to make use of. They may also authenticate with the identity provider on any device they own. Unfortunately, OpenID does not provide support for any other protocols, and this would need to be added to the service. This gives OpenID a score of 2, giving a value of 5.0.

Authentication privacy (5)

OpenID is simply an authentication gateway; it provides no support for anonymous authentication. Therefore personally identifiable information is not transmitted to or from the identity provider other than a uniquely identifiable string, namely the username. However, in order to return the user to the service after login, the return address is sent to the identity provider allowing them to build a repository of services accessed by the user.

Certain OpenID providers permit users to utilise a temporary username which is linked to their account. By doing this, services are unable to determine whether any two users are in fact the same, other than through existing techniques such as cookies and IP addresses.

Depending on the features implemented by the identity provider a score of between 2 and 4 is possible, giving values between 4.0 and 8.0.

Authentication security (8)

As previously mentioned, the identity provider in OpenID adopts the same authentication mechanism as SB IdM. Therefore, the use of HTTPS determines whether authentication attributes are secured during transfer and whether the identity provider's identity is verified. Since the identity provider is trusted in OpenID, it is able to authenticate as the user. This gives a score of between 0 and 4, a value ranging between 0.0 and 5.0.

Profile availability (8)

OpenID has two extensions which permit attributes to be supplied by the identity provider; Simple Registration (SReg) and Attribute Exchange (AX). Both of these extensions have a limited number of fields which may be requested, SReg having a small pre-determined set meant to speed registration and AX having an extensible set of attributes. This provides a consistent attribute storage interface for a small number of attributes where the extensions are supported by the identity provider.

Other protocols are not supported by the OpenID schema, meaning that identity information made available from sources other than the identity provider may not be utilised by the services.

Dependant on whether the extensions are installed a score of between 0 and 2 may be gained, giving values between 0.0 and 2.5.

Profile security (6)
Assuming an attribute extension is used, identity attributes are returned to the service through the URL. In order to ensure these are encrypted at every stage both the communication with the identity provider and the communication with the service need to be encrypted (use HTTPS).

As with SB, identity attribute storage is provided by the identity provider in the same manner as it was at the service. Therefore the identity provider is able to read all the identity attributes and they are not encrypted.

This means that the security score ranges between 0 and 3, a value of between 0.0 and 5.0.

Profile privacy (7)
Neither SReg or AX support the use of privacy policies. Instead privacy is afforded by the individual implementation of the OpenID protocol and extensions. In most cases a list of the attributes requested will be displayed to the user upon login and they can choose to share them or not.

Since all attributes are stored and obtained at the identity provider, the source is not at all obscured from the service. If identity attributes are not given by the identity provider or the extension is not supported, the service will be required to obtain the attributes directly from the user. This means that issuers are not required for identity attributes.

OpenID scores three, the same as SB, giving a value of 4.3.

Profile authenticity (2)
There is no support for providing certified attributes through SReg or AX, however in the same way that SB may automate the validation of attributes, the service may do so with attributes obtained through OpenID. Therefore a score of between 0 and 1 is awarded, a value of 0.0 to 5.0.

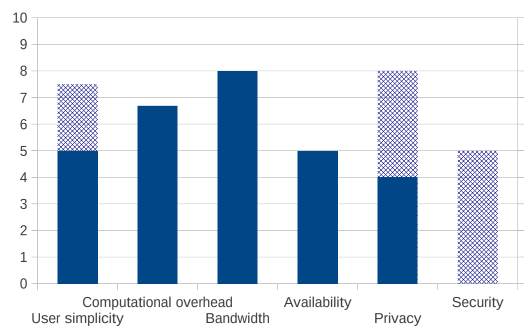


Figure 5.6: Qualitative analysis of OpenID authentication

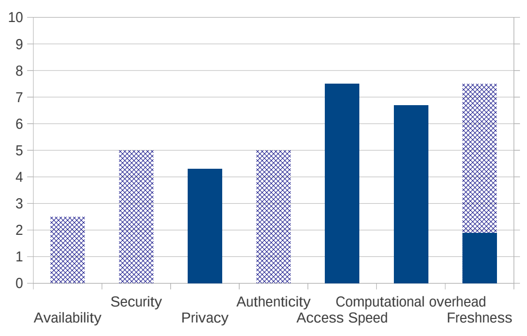


Figure 5.7: Qualitative analysis of OpenID profile data

Profile access speed (4i)

If attributes are provided, they are retrieved from the identity provider during the login stage. This means that no additional requests are required above those used to authenticate. This gives a score of 1, and being an inverse metric a value of 7.5.

Profile computational overhead (9i)

Profile access requires transfer of information between the identity provider and the service via the user's device. An assumption was made earlier that HTTPS was being used by the identity provider, therefore the score is 3, giving a value of 6.7.

Profile freshness (16)

As the attribute exchange in OpenID is performed as part of the authentication, every time the user logs in the profile is requested and retrieved from the identity provider (providing the extensions are supported). However, this relationship is one-way and the identity provider is unable to push updates to the services with out-of-date identity attributes. Since the identity provider needs to support each particular attribute type, not all profile data for all services may be centrally stored. Providing attribute exchange is implemented, a score of between 10 and 12 is possible. However if it is not, then the same score awarded to SB is given, i.e. between 3 and 5. This gives a value ranging between 1.9 and 7.5.

Graphing

As before, the values above have been taken, and plotted onto two graphs, one relating to authentication (Figure 5.6) and one for profile (Figure 5.7).

5.4 Quantitative analysis

A number of the metrics given in Section 5.3 note that they may be better assessed from a quantitative approach rather than using the qualitative metrics, assuming that a prototype has been built. Although this form of evaluation requires the implementation of the identity architecture, it gives more definitive results. The metrics proposed for quantitative evaluation are *Authentication computational overhead*, *Authentication bandwidth*, *Profile computational overhead* and *Profile access speed*.

In order to evaluate speed and overhead, it is important to fix the experimental variables. With Internet services, a number of variables exist which may skew the result, and this should be accounted for in the measurements. The major varying factor is that of network delay, which can be caused by high traffic on the network, or high load on the server. In order to minimise the variance, a local server should be chosen. Alternatively, a number of tests should be run on varying days at varying times with outlying results discarded.

The results of the quantitative testing, similarly to the qualitative metrics, will not occupy the same scale. In order to compare the results, the maximum result will be used to scale the others in the form: $Scaled\ result = \frac{Result}{Maximum}$. All of the quantitative metrics are inverse, that is that the higher the value the worse the result. Hence, the result will be (scaled to 10): $Value = (1 - Scaled\ result) * 10$.

Authentication computational overhead

The simplest method to measure the overhead of an authentication scheme is to measure the time taken to perform the computation. This encapsulates more than simply the computational overhead of the protocol itself and captures the entire process, something that is equally as important and should therefore not be compensated for. However, a number of identity architectures require user input during the authentication. The time spent rendering the page for display and acquiring user input should be discounted. The speed of the user is not under consideration in this metric, nor is their user agent (or browser).

Certain architectures utilise authentication as the method for profile transfer, and it is important that this is considered separately. In order to ensure the metric is fairly considered, an authentication request which does not request any registration information should be utilised.

Authentication bandwidth

In order to measure the bandwidth used, the data sent and received is added together for all transactions required for authentication. The transactions being considered are simply those with the user device as it is assumed that servers will likely have relatively high bandwidth and stable connections. Similar to the computational overhead, this should be measured without a request for identity attributes as these will obviously increase the size of the request.

A problem with measuring bandwidth use is that many protocols permit multiple implementations. For example, OpenID requires the user to authenticate with the identity provider, the details of which are left to the implementer. Google utilise their accounts API and transmit a large amount of data to display the webpages. The reference implementation provided by the OpenID Foundation uses a local database and transmits very little data for the graphical interface. Additionally, background traffic between the service and identity provider to verify authentication needs to be included.

Profile computational overhead

As with computational overhead for authentication, profile overhead may also be considered by the speed at which a request is handled. This measurement, for obvious reasons, also covers the topic of *Profile access speed.* Therefore both of these metrics are handled through the single measurement of profile access time as previously explained for authentication.

Profile bandwidth

Profile bandwidth was considered as part of the authentication bandwidth in the qualitative metrics as a single point. It is also measured in part by *Profile access speed* from the qualitative point of view. From a quantitative point of view it may be measured in the same manner as authentication bandwidth but for identity attribute requests.

5.4.1 Evaluating by example

This section utilises a skeleton instance of the OBS service and measures these metrics for both, a SB architecture and an IdPB architecture (OpenID is used, as before).

There are many problems with the validity of such tests on Internet applications. Consequently, an environment and a means of measurement which are capable of fairly evaluating the different approaches must be developed. Unfortunately, due to many factors which will be discussed later, this is a challenging problem.

Considering the implementations as a black box permits us to test many implementations whether or not the source is available. A black box analysis utilises external measurements to evaluate the internal processes. This makes the measurement process identical across different implementations, independent of language and process.

In order to provide quantitative results, a service was created which utilised both the SB and the IdPB identity architectures. The service consists of a login page, a registration page and the service itself, which only served to authenticated users.

Registration required the provision of an email address and some form of unique identifier, used link the user between sessions. In the SB architecture users navigated to a registration page where they were requested to enter a username, email and password (twice, for confirmation). The IdPB variation utilised OpenID (as earlier described), using both Attribute Exchange and Simple Registration extensions to request an email address from the identity provider. Tests were performed using the Google OpenID service.

The quantitative results measured were the amount of time spent on the page requests in total (ignoring user input) and the total amount of sent and received data. Both of these measurements were provided by the developer tools built into the Google Chrome browser.

Results for the tests are shown in Figure 5.8 for the login and Figure 5.9 for registration. As is evident, the increase in computational complexity and bandwidth in the IdPB architecture correlate with those predicted in the qualitative assessment; however the scale is more evident.

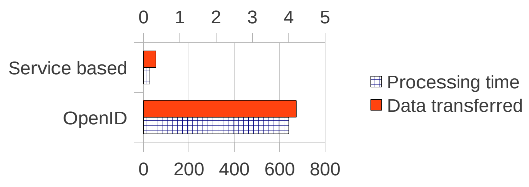


Figure 5.8: Quantitative analysis of service authentication

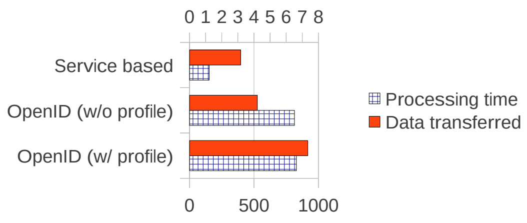


Figure 5.9: Quantitative analysis of service registration

5.5 Trade-offs

Now that the evaluation of the two architectures, SB and IdPB (specifically, OpenID), are complete, the trade-off analysis may be performed. In this situation, no goals were stated for the IdM architecture, (iėṅot optimising for any metrics). Instead the differences between the two architectures evaluated may be stated for a third party, to determine which is most suitable to the situation. In fact, Chapter 7 will utilise the information gained to produce an architecture which performs better in a number of areas.

The results in Figures 5.4 and 5.6 give a picture of the major difference between SB and OpenID. In the OpenID implementation, a much higher reading is given to user simplicity. This is due to the lack of multiple authentication credentials and reflected in the availability metric. Since the identity may only be used on the one service in the SB approach there is no availability of the identity. Unfortunately, the bandwidth required to provide this simplicity is increased due to the redirection to and from the identity provider. This bandwidth requirement is quantified in Figure 5.8. Another issue with the IdPB approach is the lower privacy and security due to the identity provider having access to the users' services and being able to log the services accessed by the user.

In Figures 5.5 and 5.7 the profile abilities of the two architectures are shown. Since with the evaluated IdPB approach, OpenID, it is optional for the identity provider to implement profile handling extensions a varied result is returned. The advantage to the centralised profile storage are seen in the possibility of increased freshness and availability of profile attributes. This is advantageous to both users as they have less issue with updating attributes, and for services as they are more likely to have up to date information about their users. Unfortunately, the computational overhead is likely to be increased and security reduced due to the transmission of attributes required between systems. This concern is reinforced by Figure 5.9 which shows the amount of extra processing time required for the remote login. However, the amount of processing time required to transfer attributes changed very little compared with a single authentication. This is a quirk of OpenID piggy-backing attribute exchange over the authentication protocol as described earlier.

5.6 Comparison with ATAM evaluation

To conclude this chapter, a comparison between the generic trade-off analysis methodology used in Chapter 4 and the results of the new combined analysis system is made. It is important to note that ATAM is designed for use by experts for evaluating a number of architectures for a particular set of requirements. The key to this method is the use of experts in order to identify the trade-off points and the list of requirements in place in order to make a decision. By specialising, the need for expert judgement is reduced. Another vital difference is the presentation of the results produced by the new trade-off analysis which is capable of showing graphically the differences between architectures, removing the need for requirements.

An interesting note to make about the ATAM evaluation is the inconsistency of the results. One favours the development time and lack of cost in outsourcing their IdM, considering trust only from the perspective of the service. Another concludes that keeping the IdM in-house is better due to the performance, security and privacy improvements it has over OpenID. This is backed up by another which states that usability is actually increased by the use of SB IdM. The overarching theme was the trade-off between security and usability. Without solid metrics, it is hard to have a consistent result. The evaluations were performed by students rather than long term engineers; however, these students (had they not furthered their education through postgraduate study) may have been making decisions such as these in industry.

Ideally, repeating the classroom experiment with the new methodology would provide a conclusive results as to the consistency of the new method over ATAM. Unfortunately, time constraints have meant that this is a study that will need to be performed in the future. The main advantage over ATAM is the ability to note variability points within a family of instantiations. Certain features may be enabled in certain situations, this is especially true when using external systems, such as an identity provider. The ease of determining values from a set of standardised boolean statements also provides a more deterministic process.

A final difference between the two approaches is the use of hard requirements in ATAM. Most reports made statements with limits on execution time or bandwidth usage. Where quantitative metrics are used in the new analysis method they are simply compared. This form of requirement based engineering is fairly popular, however it should not be utilised in a trade-off analysis. The purpose is to determine which areas are affected, and by how much due to improvements made to an architectural style.

CHAPTER 6

PRELIMINARIES

Getting information off the Internet is like taking a drink from a fire hydrant.

— Mitchell Kapor

A number of concepts are introduced in the implementation chapter which may be unknown to the reader. This chapter provides a brief overview of those concepts for reference and to improve the flow of the remainder of this thesis.

6.1 Group Signature Scheme

Group signature schemes are defined such that a member in the group may sign a message. This message may be verified as originating from the group, but the particular signer remains anonymous. Furthermore, anonymity may be revoked at a later date by a given party to reveal the signer [19]. Many improvements have been made: a static public key with dynamically added members [2], the revocation of members [15], increased security [30] and decreased key sizes [11]. As a mechanism to provide authentication to devices it matches the DB architecture fairly well. Due to this, it is worth exploring as a method of implementation.

Group signatures utilise between three and four types of entities: member, verifier, group manager and, optionally, an anonymity revocation manager. A member is part of the group signature and is capable of signing messages on behalf of the group (these may be authentication messages). A verifier is able to verify that a message has been signed by a member in a group (the authentication message is valid). A group manager maintains the public key, and optionally adds and removes members from the group. Finally, a anonymity revocation manager (which is often the same as the group manager) is able to determine the member that signed a particular message.

Multiple methods utilise the Fiat-Shamir heuristic [25] to provide a group signature [11, 15]. These are capable of being modified through the use of a counter to fulfil the requirements proposed in the next section. These are secure in the random oracle model.

Camenisch & Groth describe a more efficient group signature scheme [13]. Its improved efficiency does not require the public key to be changed when adding users to a group. However, this improvement allows an untrusted group manager to add users without any evidence of them doing so.

Other efficient schemes such as [30, 10] do not maintain a value with all members contained. As such, users may not be revoked securely from the group. By extending these schemes with a dynamic accumulator, it is possible to add this functionality.

6.2 Monotonic Counter

A monotonic counter is an entity which is capable of producing a unique and verifiable output to an increment instruction to the owner. Its current value must also be readable to any interested party in a verifiable manner. In its simplest form, a single number may be stored and this number is incremented on request. The signature of an increment or read value is provided by a signed transaction message. It is important to note that a monotonic counter can never produce the same output after two separate increment requests for its entire lifetime.

Our proposed scheme utilises a monotonic counter as a trusted entity. This allows changes to the public key to be linked to a specific counter value. This facilitates protection against replay attacks by providing a trusted version to each unique user set. Only the set of users assigned to the current counter value will be accepted.

One option would be to utilise a central trusted global clock. A mechanism for providing a secure counter on an untrusted system with a global clock is provided in [57]. The proof expands for each increment, and hence would become unfeasibly large.

Utilisation of a hardware counter (such as that provided by the Trusted Platform Module (TPM)) for each identity provider produces a similar problem. This is due to a counter being required for each user, and hardware devices only supporting a limited number, thus in essence becoming a global clock.

In the same paper ([57]), an extension to the TPM is suggested which provides an efficient mechanism to provide (theoretically) infinite virtual trusted counters. This requires a single function to be added to the specification of the TPM. By providing the trusted entity in hardware, the need for an external trusted service is removed.

Finally, the use of an external trusted entity to provide a counter for each identity is possible. This method provides a secure and trusted counter. However there is little commercial viability for such a service to exist. It is possible that a set of roots may be run by public institutions, such as those used for root servers in the domain name resolution scheme.

6.3 Dynamic accumulators

Implementations of group signatures have utilised many different cryptographic primitives. One such method is that of a dynamic accumulator [15, 14]. A value is generated from multiple inputs, similar to that of a secure hash function. Inputs may be added and removed from the generated value without knowledge of the other committed values. Furthermore, it is possible to prove that a single value has been committed without divulging any other committed value through the use of a witness. However, it is infeasible to prove that a value has been committed when it has not.

The construction due to Camenisch & Lysyanskaya [15] is proved secure under the strong RSA assumption. Five functions are defined:

- `Setup`: Generates a standard RSA modulus and a random start value
- `Commit`: Accumulate the new value into the accumulator and provides a witness
- `Delete`: Removes the value from the accumulator using the private key
- `Update Witness`: Performs the add and remove functions on the previous witnesses, effectively disabling witnesses for deleted values
- `Verify`: Can verify that a value exists within the accumulator with the value, the accumulator and the witness

6.4 Diffie-Hellman key exchange

Diffie-Hellman (Williamson) key exchange is a method of two parties interacting over an open connection to derive a shared cryptographic key without other parties listening in on the interaction being able to determine the value of the key [23]. This method utilises mathematical constructs to permit the two parties to easily derive a key through exponentiation, whereas for an outside party to perform the same action they are required to either perform a brute-force attack or determine the logarithm. For sufficiently large numbers, this is infeasible. The definition of sufficiently large obviously varies as computers become more powerful.

Diffie-Hellman, although secure (given large enough numbers) to passive attackers, is not secure against those which are able to intercept and modify transmissions between the two communicating parties. This kind of attack is known as a man-in-the-middle attack. It is widely recognised as an issue with the exchange, as such additional features must be included in protocols which utilise this form of key exchange to mitigate this problem.

CHAPTER 7

DEVICE BASED IDENTITY

Constant development is the law of life, and a man who always tries to maintain his dogmas in order to appear consistent drives himself into a false position.

— Mohandas K. Gandhi

7.1 Evolving identity management

Miller, in an article written in 2009, states that he has "become convinced that the entire system of usernames and passwords is broken"[48]. Solutions intended to address this issue have been made, most of which have been discussed in previous chapters. There are fundamental scalability problems with a username and password system where all users are on a particular identity provider. The problem of scalability can be ameliorated by utilising numerous identity providers. However, users still need to remember these details. Instead, a system where users are not required to have usernames or passwords is the final goal. This must be achieved, while not only maintaining current security standards, but increasing them.

Identity attributes are often overlooked in identity management. Koch & Wörndl note that a central repository does not increase privacy [44]. Regardless, the issues of distributed identity attributes still exist. A new method of attribute storage and/or distribution is required. This should be considered at the same time as authentication, not separately.

In the previous chapter, the lack of profile availability and usability in systems was identified as being improved in the OpenID implementation. It is reasonable to conclude that these may be improved further by fulfilling more of the requirements for the metrics. The overview given in Figures 5.2 and 5.3 show how the credentials and attributes are stored at the identity provider, instead of at the service. The privacy and security decrease is caused because the attribute provider and authentication provider moves also. By moving the attribute provider and authentication provider to the user device (or client) the benefits of both architectures may be combined.

It is a fair assumption that service providers will only implement changes to their authentication if it is in their self-interest. This is a possible reason for the lack of adoption of a single sign-on architecture that is compatible across service providers, despite the advantages displayed in the previous chapter. The cost of implementation is not worth the benefits, from the service provider's perspective. There are a number of options to make new identity management architectures more appealing to services. These include the provision

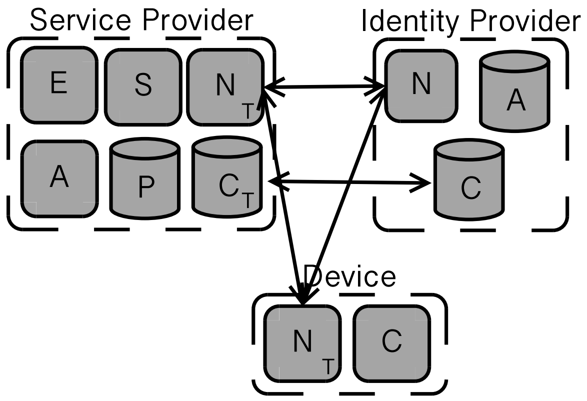


Figure 7.1: Framework model for device based identity management

of extra features for the service, and large-scale adoption by the user base through a popular identity provider.

Due to the trust being placed in the device, this form of identity management will be known as device based identity management (DB IdM). This is similar to the Portable Authentication Device proposed by Jøsang & Pope [39]. However there may be multiple devices within the identity. Using the modelling defined in Section 5.2, the high-level overview of this form of IdM is shown in Figure 7.1. A user is able to add and remove a device from their identity, all other authentication operations are automatic. Since there is no single credential for the identity, if a device is compromised the user can easily resolve the issue. Rather than resetting a password on all services and devices, a single operation restores security. If the user only owns one device, or their devices have been maliciously revoked, it is possible to reset security with the aid of the trusted service. The evaluation method will now be executed on this architectural idea in order to demonstrate the differences between this and the previously evaluated identity architectures. This evaluation will be for guidance due to the lack of implementation details, but will still provide a result determining whether this line of development is worthwhile.

Authentication user simplicity (4)

DB authentication requires a username to determine their identity on the identity provider. This is required for activation of the device only and is then remembered until the device is removed from the identity. A password simply authenticates the user with the identity provider and may not be required in all situations, however it is recommended for a second layer of security. The authentication provided by the device does not use this password. Instead, it utilises authentication credentials specific to the service, obtained from the identity provider.

Authentication with the DB architecture utilises a common interface over different services and devices if supported. Credentials may be automatically sent to services from the device through the generic authentication module on the devices. In case the service does not support the mechanism, a password store may be implemented in the same way as conventional browser based password storage.

Given these architectural definitions, a score of between 2 and 4 can be obtained dependent on the support provided by the service. This results in a value between 5.0 and 10.0 ($\frac{2}{4} * 10$ and $\frac{4}{4} * 10$).

Authentication computational overhead (9i)

Communication of authentication credentials between the device, identity provider and service requires at least the use of HTTPS, and id dependent on implementation specifics, any number of other cryptographic methods. Along with bandwidth, this metric is very uncertain without implementation specifics. The score range produced is between 3 and 9, giving a value between 0 and 6.7 ($10 - (\frac{9}{9} * 10)$ and $10 - (\frac{3}{9} * 10)$).

Authentication bandwidth (5i)

Authentication requests require obtaining authentication credentials from the identity provider, and are dependent on the location of the identity attributes performing a request to external services. Therefore, the best score possible is 1, with the worst being 4, if attributes are not cached and are required for each request. The value, due to this being an inverse metric, ranges between 8.0 and 2.0.

Authentication availability (4)

Availability depends on the implementation. If identity attributes may be stored at the identity provider, then there is the possibility of implementing multiple authentication protocols. It is also theoretically possible to pass authentication credentials between devices, given this centralised storage. By this reasoning, DB identity is capable of scoring a full 4 in availability, although it could also gain a score of 2, if centralised storage is not possible. This gives a value range of 5.0 to 10.0.

Authentication privacy (5)

As the authentication and attribute providers are now located in the device, only information distributed by the user is visible to any third party. Verification of credentials may be required by services permitting tracking by the identity provider. Since there is no entity available that is trusted by the service provider, the user cannot provide a different identifier to the service provider. This means that the service provider is able to link the user's sessions together unless anonymous authentication is permitted. Where certified attributes are supported, anonymous authentication is also supported, as the attribute provider is able to provide input to the authentication. The score provided is therefore between 1 and 4, values ranging between 2.0 and 8.0.

Authentication security (8)

It was stated earlier that all communications would be encrypted, and therefore all credentials will be encrypted during transmission. Given service provider support, high-entropy keys may also be utilised to authenticate. In fact, even when not supported by the service, a long random password may be used and stored when centralised storage is supported. As the authentication provider is part of the device, the major improvement over IdPB IdM is the removal of authentication power from the identity provider. DB identity scores between 5 and 7 in security, values of 6.3 and 8.8.

Profile availability (8)

With the attribute provider at the device a consistent interface exists for attribute retrieval over multiple services. Dependent on the implementation of the client software, multiple protocols may be supported for responding to attribute queries. Requesting attributes depends on the authentication of the attribute storage and therefore only one protocol is supported. In order to keep the attributes fresh they must be centrally managed and therefore not stored on the device. The implementation of the authentication will determine whether it may be used to secure attributes at the identity provider, and therefore store them centrally. Otherwise, attributes may be exchanged between services. This scores between 3 and 5, giving a value between 3.8 and 6.3.

Profile security (6)

As with authentication, attributes are transferred during transmission. Dependent on the implementation, attributes may be stored at services which means they are unencrypted, or at the identity provider in an encrypted form. The use of HTTPS verifies the destination. This scores between 4 and 6, values of between 6.7 and 10.0.

Profile privacy (7)

Privacy policies have not been considered, although they may be implemented on the device software. This is not a requirement for services to adopt the architecture as this may dissuade usage. Transmission of all attributes from the device masks the source and are not required to be certified. Profile attributes sent via the device are screened by the user to ensure they are correct, and (importantly) that they wish to send them. This scores between three and five, giving a values between 4.3 and 7.1.

Profile authenticity (2)

There is no support for automated verification of attributes as the device can not be trusted to perform as such. However, attributes obtained from other sources may be certified and passed through. This gives a value of 5.0.

Profile access speed (4i)

Requesting attributes from a central store only requires a single request. However multiple requests may be required if they are not in a central location. Requests may be distributed through the identity provider, so long as the identity provider is unable to modify the requests to obtain the attributes itself. This gives a score of between 1 and 3, a value of between 2.5 and 7.5.

Profile computational overhead (9i)

Like the computational overhead of the authentication, it is entirely dependent on the implementation. However, the same assumption of minimum security is made, giving a min-

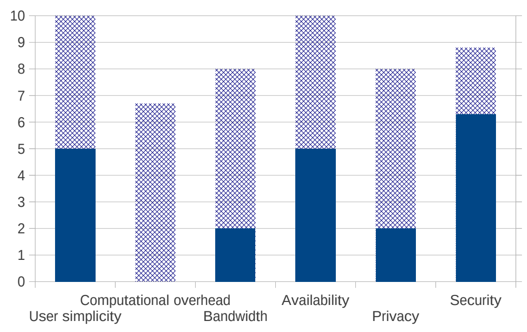


Figure 7.2: Qualitative analysis of device based authentication

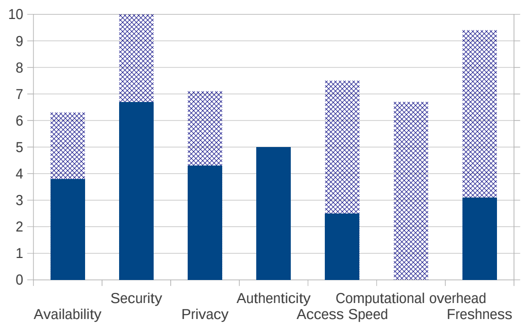


Figure 7.3: Qualitative analysis of device based profile data

imum score of 3 and the maximum unknown (therefore 9, the maximum possible), leading to values ranging from 0 to 6.7.

Profile freshness (16)

Profile freshness is a complex metric. There are two situations with DB IdM. The first is where attributes are stored at services and re-distributed to others; the second is where attributes are stored centrally and distributed, both of which operate via the user device. In both cases the location of profile data is maintained in a central location. In the first case the user is able to view all data on services and modify some. In the second, they are able to view and modify all profile data at the identity provider attribute store. This gives a score of 5 in the first case and 15 in the second, providing a range of values between 3.1 and 9.4.

Graphing

Once again, the values above have been taken and plotted onto two graphs, one relating to authentication (Figure 7.2) and one for profile (Figure 7.3).

Comparing Figure 7.2 with Figure 5.6 the potential for user simplicity is increased in the DB architecture over the OpenID implementation. Security is also given a higher minimum standard than is possible in single sign-on with the ability to become even more secure. Unfortunately, the minimum level of privacy is reduced and there is the prospect of far more computational overhead and bandwidth required. Overall, dependent on implementation specifics, the DB architecture seems a reasonable progression for security and usability in authentication.

Moving our attention to identity attributes, the profile abilities shown in Figure 7.3 show an increase in availability, privacy, authenticity, freshness and security over both SB and IdPB architectures (shown in Figures 5.5 and 5.7). These improvements are possible due to the changes made in authentication, and the movement of the attribute provider to the device. Unfortunately, this slows down access speed and increases computational overhead. Implementation will give a more detailed result as to whether this is unbearable for the user.

The remainder of this chapter will explore two possible implementations of DB IdM. This will provide the ability to perform quantitative assessments of the scale of additional overhead required to utilise this architectural style. The first implementation, described in Section 7.2, will utilise an external trusted entity in order to verify authentication. Through the use of a trusted service the DB architecture may be implemented without the trusted identity provider. By making the trusted service simple it should be easier to find a mutually trusted provider of this service, such as a hardware device in the identity providers' servers. A second implementation is then proposed in Section 7.3 which utilises a dual authentication technique to secure data from the identity provider and from unauthorised devices.

7.2 Device based identity management with an external trusted service

Group signatures (as described in Section 6.1) is very similar to the requirements of the device based identity management, as shown in Figure 7.1. The conflict arises with the group manager role being provided by the identity provider. However the ability to add members removes the authentication control from the device. This obviously needs more consideration. In DB IdM architecture it is desirable to ensure that the group manager should not be able to add or remove members without the knowledge of the current group members.

To provide this ability an extension to traditional group signature schemes is required in order to mitigate the trust from the group manager. Instead, an external trusted entity is utilised to ensure that devices are not added or removed without the knowledge of the user.

Although adding another system seems counter intuitive as this extra system could simply perform the task of the identity provider, if it is established that limited resources are required a simple service or hardware in the identity providers' servers may perform this task. This follows a similar method to that of public key infrastructure, where a limited number of trusted roots sign many public keys. Another example of this is DNS where a set of root servers delegate to lower levels. This enhances the security of the scheme, such that both the group manager and trusted entity would need to collaborate in order to compromise the group.

In the past, the main use of group signatures has been to provide members of the group anonymity. For example, users of the Trusted Platform Module (TPM) are able to verify to a third party certain details without revealing their identity. A third party may only detect that the signer is a member of the group of all people in possession of a TPM. This gives the verifier confidence that a trusted entity signed the message, but privacy to the signer.

The use of group signatures in identity management has previously been proposed by Isshiki et al. [36]. Their work similarly chooses to utilise the signature for access control. However, if the group signature is to be trusted for authentication and the release of private data, the user must trust that all members of the group are legitimate devices. Given the potential sensitivity and quantity of private data available, it is unwise to trust the identity provider. However, since the group manager must be available at all times, it is logical to assign this role to the identity provider. Therefore, the user must be sure that all actions performed by the identity provider (as group manager) are traceable, hence eliciting the need for a group signature where the group manager is not trusted.

Utilising these properties a new authentication framework is shown in Figure 7.4 which is compatible with DB IdM. This framework removes the trust from the identity provider and utilises the user devices for managing the identity. Previous solutions utilise a traditional authentication scheme, such as a password. Instead, the use of a cryptographic signature provides greater security. This is due to the difficulty of performing a brute force attack since cryptographic keys contain more entropy than most passwords. The user burden is

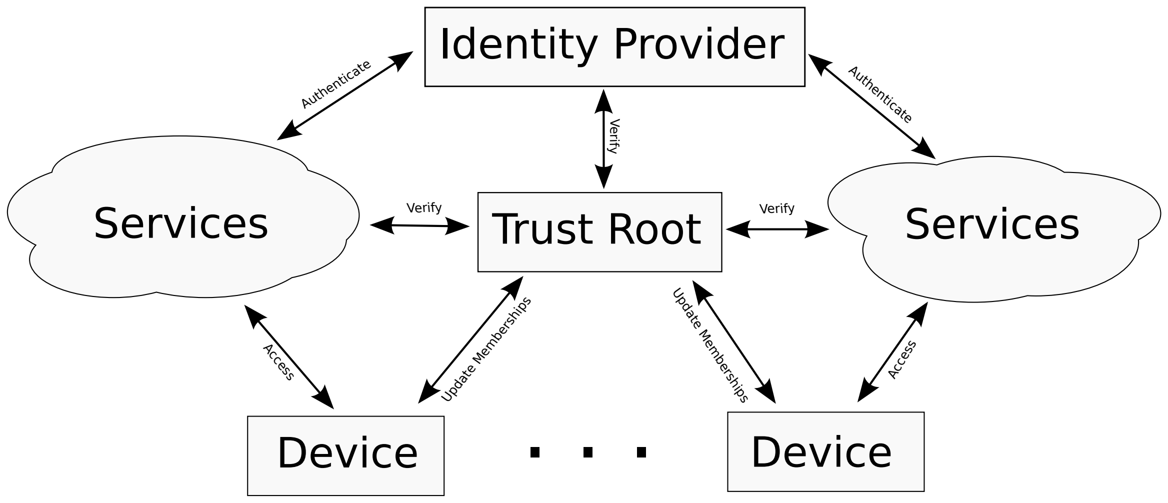


Figure 7.4: Device based authentication framework

also reduced as they do not have to remember passwords, and instead simply activate and deactivate devices with their identity. Finally, credential security is improved due to the distribution of the public key and trust over two systems, therefore requiring collaboration in order to break the security.

With a secure mechanism for identity verification in place, signatures may be used for other identity management functions. For example, signed requests for data transfer between services may be verified to originate from the owner of the data. This allows for a distributed profile where services are the attribute store, as required by DB IdM.

The remainder of this section describes the group signature scheme for use in DB IdM. This involves a review of existing work in the area of group signatures, an introduction to monotonic counters and a definition of the requirements for the group signature scheme. Following this the group signature itself is presented and evaluated for security.

7.2.1 Requirements

Mobility The mobile nature of service utility requires us to ensure that any protocol maintains a low bandwidth, and little client side storage or processing. Limiting these requirements from services and identity providers, will aid in adoption with minimal cost. To facilitate distribution of identity between all user devices, no hardware specific requirement from the client should be made other than those provided by a basic Internet enabled device.

Security

1. The group manager must not be able to sign a message on behalf of the group without the the owner being aware. This property must be maintained even if the the group manager learns the private key of a group member.

2. A member of the group is able to revoke any other member at any time, such that any message signed by the revoked member is invalid.

Trust The system must have a source of trust to ensure that the security requirements are upheld. This has to be available at all times, and to all group members. A simple source of trust is a verifiable monotonic counter. This may be implemented as a piece of trusted hardware owned by the group manager or by a trusted third party service. A monotonic counter is defined as any verifiable counter which never produces the same output twice. It is not required that the counter specifically count in any order. Example implementations of monotonic counters are discussed previously.

7.2.2 Implementation

In order to guarantee the security requirements given in the previous section, two properties are defined in regards to the monotonic counter. These ensure that any membership action on the group is traceable by members of the group.

1. The group manager must not be able to perform a `Join` without incrementing the counter value. Any attempt to do so will cause subsequent signatures to be invalid.

2. All `Revoke` actions must increment the counter value. Any member requesting the revocation of another member will be able to verify that the counter has been incremented and the chosen member removed from the public key.

3. The public key must contain the current counter value. No public key may exist with the same counter value such that it may be determined as valid by a verifier. Any attempt to pass a public key which is not current must be detectable by the verifier.

Example implementation

By extending the dynamic accumulator based group signature schemes in [15] and [2], as described in Section 6.1, a reference implementation is possible which fulfils the requirements given. It should be noted that the scheme being extended is secure in the *random oracle model* and is sub-optimal. Therefore, this implementation is as well. However, recent improvements to provide secure methods in the *standard model* and enhance efficiency, lack the ability to allow an untrusted group manager to update the public key. The underlying enhancements may be applied to future works which are able to maintain the security properties. The protocol is also designed such that it is resilient to communication failures between the group member and group manager.

`Setup`:

Figure 7.5 describes the protocol for initialising the identity. The value PK is the public key construct for the group signature scheme in [15], described in Section 6.1. The basic process is described below:

1. The group key pairs are generated by the user to ensure no values are accumulated at the start.

2. The private key is added and accumulated according to the original group signature scheme.

3. The public key, name of the device and the device's public key are sent to the trusted entity.

4. A counter is initialised for the identity and the starting value is stored with the public key along with a signature over the key.

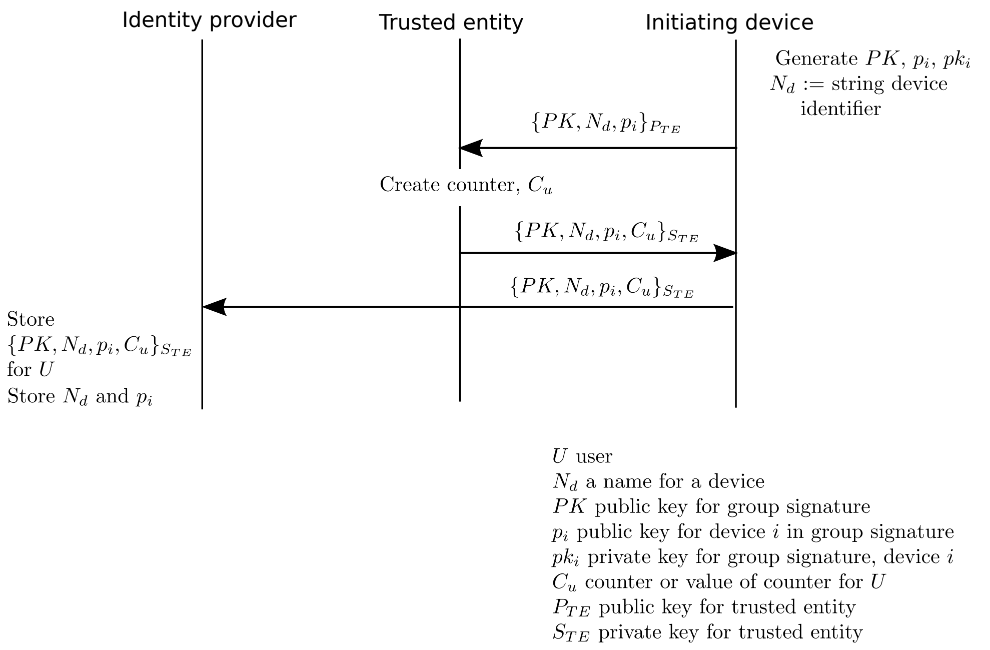
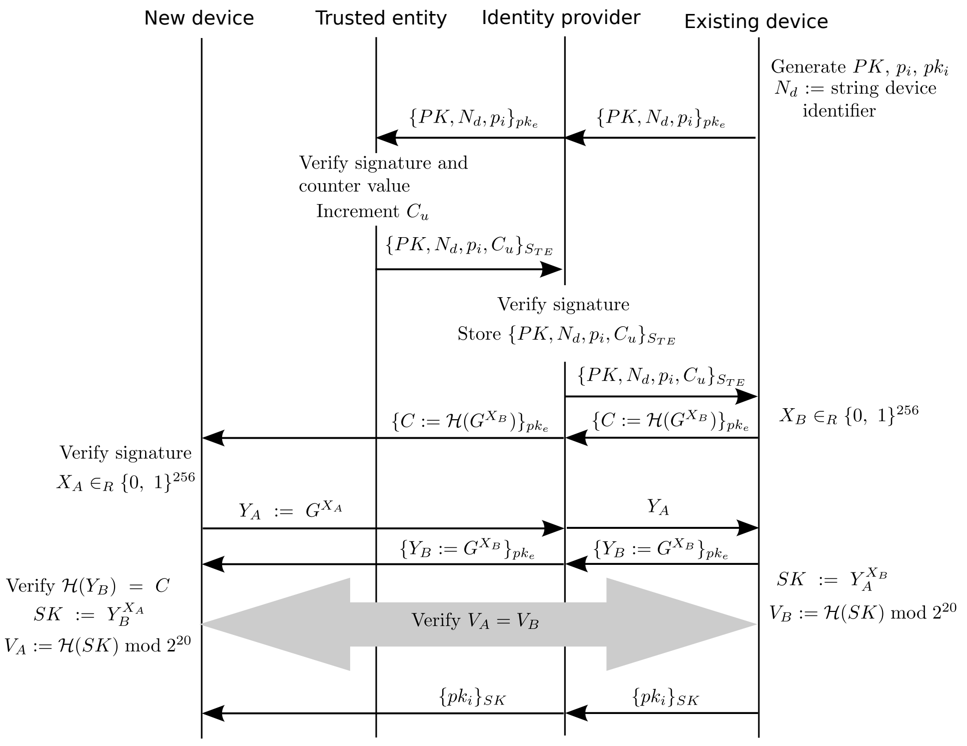

Identity provider
Trusted entity
Initiating device
Generate PK, p_i, pk_i
N_d := string device identifier
$\{PK, N_d, p_i\}_{P_{TE}}$
Create counter, C_u
$\{PK, N_d, p_i, C_u\}_{S_{TE}}$
$\{PK, N_d, p_i, C_u\}_{S_{TE}}$
Store $\{PK, N_d, p_i, C_u\}_{S_{TE}}$ for U
Store N_d and p_i
U user
N_d a name for a device
PK public key for group signature
p_i public key for device i in group signature
pk_i private key for group signature, device i
C_u counter or value of counter for U
P_{TE} public key for trusted entity
S_{TE} private key for trusted entity

Figure 7.5: Setup protocol for group signature device based identity management

5. The signed tuple are send to the identity provider and stored as a new identity.
6. A private key in the group signature scheme provides an additional value x which is able to revoke anonymity. This may be omitted by the user from the private key if they do not require this functionality.

`Join`:

1. An existing device creates a second membership to the group as described in the group signature scheme.
2. A signed message is sent from the existing user with the new public key to the trusted entity via the identity provider. This ensures that the updates are not lost, if the device connectivity fails at this point.
3. The trusted entity verifies the signature and increments the counter, returning a signature over the group public key and the new counter value to the identity provider.
4. The signed updated value of the public key is stored at the identity provider after verifying the signature and that the counter value is correct.


New device
Trusted entity
Identity provider
Existing device
Generate PK, p_i, pk_i
N_d := string device identifier
$\{PK, N_d, p_i\}_{pk_e}$
$\{PK, N_d, p_i\}_{pk_e}$
Verify signature and counter value
Increment C_u
$\{PK, N_d, p_i, C_u\}_{S_{TE}}$
Verify signature
Store $\{PK, N_d, p_i, C_u\}_{S_{TE}}$
$\{PK, N_d, p_i, C_u\}_{S_{TE}}$
$\{C := \mathcal{H}(G^{X_B})\}_{pk_e}$
$\{C := \mathcal{H}(G^{X_B})\}_{pk_e}$
$X_B \in_R \{0,\ 1\}^{256}$
Verify signature
$X_A \in_R \{0,\ 1\}^{256}$
$Y_A\ :=\ G^{X_A}$
Y_A
$\{Y_B := G^{X_B}\}_{pk_e}$
$\{Y_B := G^{X_B}\}_{pk_e}$
Verify $\mathcal{H}(Y_B)\ =\ C$
$SK\ :=\ Y_B^{X_A}$
$V_A := \mathcal{H}(SK) \bmod 2^{20}$
Verify $V_A = V_B$
$SK\ :=\ Y_A^{X_B}$
$V_B := \mathcal{H}(SK) \bmod 2^{20}$
$\{pk_i\}_{SK}$
$\{pk_i\}_{SK}$

U user
N_d a name for a device
PK public key for group signature
p_i public key for device i in group signature
pk_i private key for group signature, device i
pk_e private key for group signature, existing device
C_u counter or value of counter for U
P_{TE} public key for trusted entity
S_{TE} private key for trusted entity
G Diffie-Hellman public base

Figure 7.6: Protocol to add a device to the identity in group signature device based identity management

5. Once the public key is updated, the new membership is transferred to the new device using a Diffie-Hellman key exchange. This is verified using an out of band communication of a short verification value.

It is also possible for the new device to generate its own membership to the group. The updated public key is then sent to an existing member to update the counter value. This method is preferred when there is insecure communication between the devices as the public key is the only transmitted data. However, verification that the correct public key arrived must be performed. The above method requires less device-to-device bandwidth between the devices as the member key is shorter than the public key which may be retrieved from the identity provider. This is advantageous when device-to-device communication is limited, such as through user input, or low-bandwidth wireless connections.

`Revoke`:

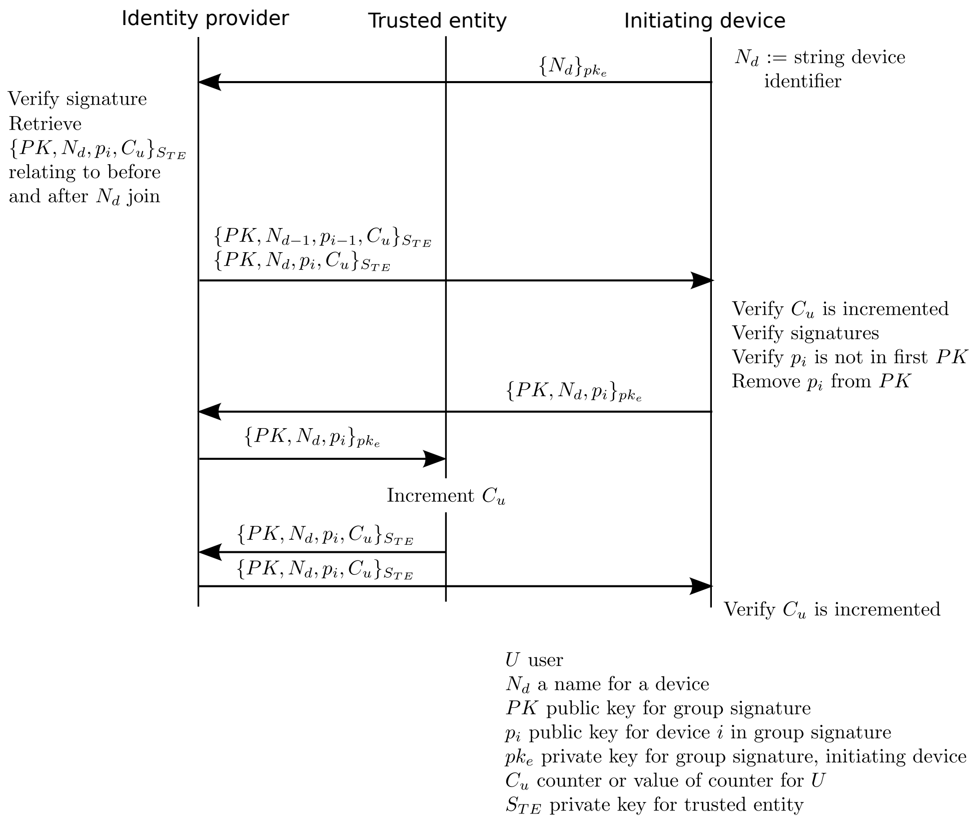

Identity provider
Trusted entity
Initiating device
$\{N_d\}_{pk_e}$
N_d := string device identifier
Verify signature
Retrieve $\{PK, N_d, p_i, C_u\}_{S_{TE}}$ relating to before and after N_d join
$\{PK, N_{d-1}, p_{i-1}, C_u\}_{S_{TE}}$
$\{PK, N_d, p_i, C_u\}_{S_{TE}}$
Verify C_u is incremented
Verify signatures
Verify p_i is not in first PK
Remove p_i from PK
$\{PK, N_d, p_i\}_{pk_e}$
$\{PK, N_d, p_i\}_{pk_e}$
Increment C_u
$\{PK, N_d, p_i, C_u\}_{S_{TE}}$
$\{PK, N_d, p_i, C_u\}_{S_{TE}}$
Verify C_u is incremented
U user
N_d a name for a device
PK public key for group signature
p_i public key for device i in group signature
pk_e private key for group signature, initiating device
C_u counter or value of counter for U
S_{TE} private key for trusted entity

Figure 7.7: Protocol for revoking a device from the identity in group signature device based identity management

1. The user retrieves the public key of the device they wish to revoke from the list at the identity provider using the string identifier.
2. The public key is verified by checking the signature over the value of PK before the user `Join` operation and after the next operation performed. The public key should not be part of the accumulator in the first instance, but should be in the second.
3. The public key is modified by the user to remove the device from the public key.
4. A signed message is sent to the trusted entity with the updated public key via the identity provider.

5. The trusted entity verifies the signature and increments the counter, returning a signature over the group public key and the new counter value.

6. The updated public key is stored at the identity provider after verifying the signature and that the counter value is correct.

`Witness Update`:

1. Each update to the public key is downloaded from the identity provider.

2. The signatures are checked and the user is requested to review the changes, identifying devices by the value of d.

3. Assuming the user accepts the updates, a standard witness update is performed. The updated local public key is then verified against the remote public key to ensure the update was as expected.

4. If the user does not accept the updates, they have the option to perform operations to fix the group or contact the identity provider regarding anomalous operations on the group.

`Sign`:

1. A member generates a standard group signature on any message.

`Verify`:

1. Obtain the public key and verify σ is valid and c is the current counter value.

2. Verify the signature with PK using the standard group signature scheme.

7.2.3 Evaluation

Mobility

With the limited resources of mobile devices, it is important to assess the usage required by the given scheme. Additionally, the scheme needs to be resilient against devices losing connectivity during a group operation.

Transmission of signatures and keys are limited by the bandwidth on the device. Users will expect the delay for an authentication to be very short. The signature over an authentication message is not changed by the introduction of a counter. Signatures in the scheme extended in this paper are of length $k + 16l_p$, where k is the length of the hash function, and l_p is the size of the moduli. Assuming the use of a 1024-bit moduli ($l_p = 1024$) and SHA1 ($k = 160$), a signature is of length 16544 bits (~16kb). Over a GPRS connection this permits a signature to be sent in two to four seconds. Although this is not the instant response expected from users, most mobile devices now make use of faster connections.

All client operations are based on signing and verifying public key signatures. Most mobile devices have the ability to browse SSL secured websites. Therefore no special hardware is required of the client, nor is there too much processing required.

The `Join` and `Revoke` operations are both resilient to connection failure, which is often a problem for mobile applications. The trusted entity sends the updated public key directly to the identity provider. Therefore, if a mobile device loses connectivity, there is no inconsistency with the public key. Once the connection is restored, a device is able to verify whether the operation completed by reading the current public key. If the operation was not completed it may be re-run.

Security

Group signatures are defined to have the following properties, and hence security is mostly given due to the base group signature scheme:

- Correctness: Signatures produced by the `sign` method must be accepted by `verify` unless this conflicts with any other property.

- Unforgeability: Only current group members are able to correctly sign messages.
- Anonymity: The verifier is unable to determine the member within the group which produced a signature.
- Unlinkability: It is infeasible to determine whether two signatures were generated by the same member.
- Exculpability: Neither the group manager nor another group member may sign a message such that it appears to be signed by a different group member.
- Traceability: Given a signature, the open entity may determine which member of the group signed a particular message.
- Coalition-resistance: A colluding group of members, perhaps including the group manager, cannot violate any of the properties.

The extensions proposed do not affect these properties, however the following extra properties which require assessment are defined:

- Tamper-evident: Any addition or revocation of group member must be detailed to each valid member before they may sign any further message.
- Public key verifiability: A verifier is able to assert that the copy of the public key obtained from the group manager is fresh.

Theorem 1 (Tamper-evident). *Given a secure monotonic counter, no entity may join the group without making all other members aware.*

Proof. Let us assume that the group manager is malicious and has compromised a device (D_c) which is a member of the group. Since the group manager distributes the public key, it is also able to send a different public key to the verifier than that exposed to group members.

We will attempt to add a device (D_m) using D_c without making any other member aware. Initially, a join operation on D_c for D_m is performed. Optionally, the counter value may or may not be updated. In the case where it is updated, the public key is correct but the device is visible in the public key, whereas if it is not updated the public key is not valid for D_m.

First consider the case where the counter is updated. In order to mask D_m from the user a second device (D_p) is added to the group and again the counter is updated. This masks the device D_m from other members, instead making a third device visible in the public key. When a valid device attempts to sign a message they must update their local public and private keys. Since the group manager does not want to to give details of D_m, they instead combine the witness update of D_m and D_p together and send that to the valid user. When the user does not recognise D_p, they remove it from the identity by deleting the user that was just provided. Since the update was combined, removing D_p also removes D_m. Another option for the identity provider is to wait until the user adds the new device D_p themselves, and will therefore recognise it and not remove it. In this situation, even if the identity provider intercepts the communication to the trusted entity and adds D_m, the adding device is able to detect the modification to the public key.

In the second case, the counter is not updated. The identity provider is able to add a device D_m to the public key. In order for the public key to be valid, the identity has to break the trusted entity's signature over the public key and counter value.

Theorem 2 (Public key verifiability). *Given a secure monotonic counter, a group manager is unable to assert an out of date public key*

Proof. The counter value is unforgeable as the value is retrieved for each verification. The value of the public key is signed along with the counter value. Assuming the signature σ is unforgeable and the counter may never return the the same value for c, the public key may be verified to be fresh. Otherwise either the value of c will not match or σ will be an invalid signature over the public key.

Trust

Two options have been given for the source of trust: the use of a piece of trusted hardware and a trusted third party. Trusted hardware is currently unable to provide a reasonable number of counters for an identity provider. However, extensions discussed in the monotonic counters discussion in Section **??** provide this ability. The advantages of a hardware solution are the lack of an external service being required, and a simplified protocol as the current value may be sent with the public key. A single hardware counter does present issues concerned with hardware failure or unavailability. An external service is able to be more resilient, but requires a trusted entity to run it. In general, the use of an external service provides the most resilient solution. This also provides a central contact to protect an identity against a rogue identity provider.

7.3 Device based identity management through three-factor security

In order to remove the trusted entity, a complete change of strategy is required. If a reasonable amount of trust is given to the identity provider, an external trusted entity is no longer required. By trusting the identity provider to only distribute profile attributes to user defined devices, the revocation security afforded by the trusted entity is transferred to the identity provider. Through the use of careful design, the other properties are maintained.

7.3.1 Overview

As mentioned previously, our solution utilises an identity provider to store encrypted data and to ensure that encrypted data is kept secure. Through the use of multi-factor security we are able to provide a level of security which was previously unobtainable from a centralised system.

Figure 7.8 shows the flow of profile data through our proposed scheme. Encryption and de-

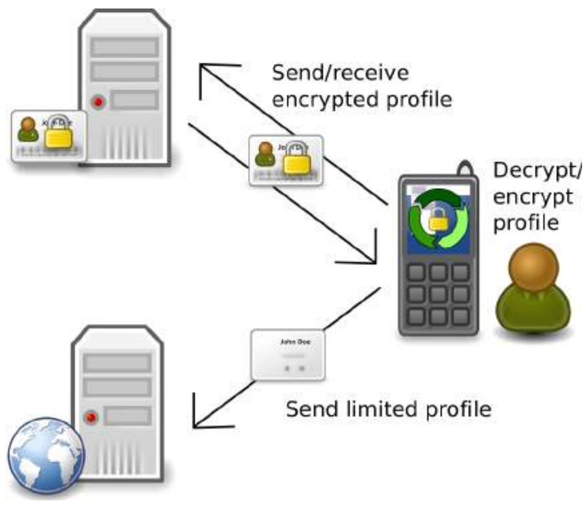


Figure 7.8: An overview of profile data flow in our proposed identity management scheme.

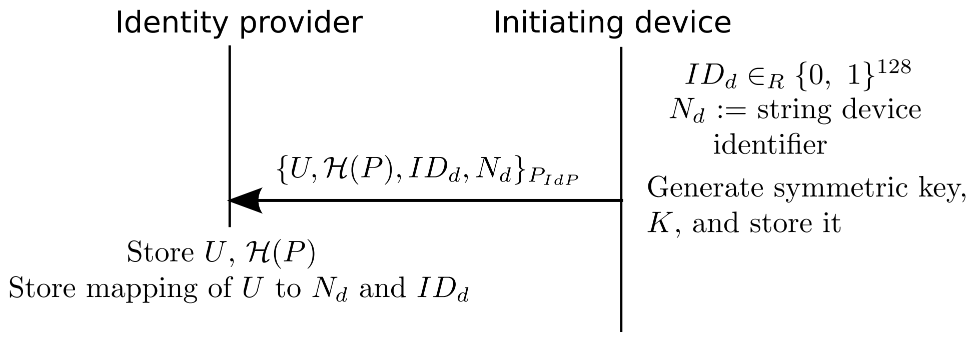
cryption may only be performed by devices which contain the symmetric encryption key. This keeps profile data secure from the identity provider. Later, we describe how this symmetric key is passed between devices in a secure and user friendly manner. Users can store profile data at the identity provider and provide this to services. Password storage at the identity provider allows backwards compatibility with previous identity management architectures, providing essentially a centralised (yet secure) automatic form filler.

What is not shown is the authentication with the identity provider. In order to request encrypted attributes from the identity provider, the user must first pass an authentication protocol. This entails sending their username for their account at the identity provider, their password for the identity provider and the device's unique identification number. The key to revoking devices is the identification number for the device. This is randomly chosen when the device is added to the identity and is stored by the identity provider. A sufficiently large number is chosen such that guessing a valid identification number is highly unlikely, even if the username and password are known. If a device is lost or stolen, its identification number is removed from the list of permitted devices by the user. The race condition is aided by the use of a password.

In order to add a new device to the identity, the symmetric key must be transferred from an existing device. It is important that the key is not exposed to the identity provider or any third

party throughout this transfer. We utilise a modified version of Diffie-Hellman key exchange (see Section 6.4) in order to utilise the identity provider as a communication gateway. Our modifications make it highly improbable that the identity provider is capable of intercepting the key during the transfer. At the start of the protocol a new random identification number is generated. The identification numbers are kept private between the device and the identity provider. In this way, a compromised device has no knowledge of other uncompromised devices.
The remainder of this section is dedicated to describing each of the required protocols for this identity management scheme in detail.

Identity creation


Identity provider
Initiating device
$ID_d \in_R \{0,\ 1\}^{128}$
$N_d :=$ string device identifier
$\{U, \mathcal{H}(P), ID_d, N_d\}_{P_{IdP}}$
Generate symmetric key, K, and store it
Store U, $\mathcal{H}(P)$
Store mapping of U to N_d and ID_d

U user identifier
$\mathcal{H}$ secure hash function
P user defined password
ID_d a random, unique within user, identifier for device
N_d a unique string identified for device
P_{IdP} public key for identity provider

Figure 7.9: Protocol for creating a new identity.

In order to utilise an online identity, it first needs to be created. This process consists of finding an identity provider and registering an account with them. A device string identifier, N_d, is given by the user and sent to the identity provider along with a randomly generated device identity (see *device addition* for details). This is accepted by the identity provider, as the identity was previously blank. The device identity is secret, therefore it must be encrypted. The device then needs to generate a symmetric encryption key, K, which is stored on the device. Any information which the device wishes to store on the identity provider is encrypted using the symmetric key, thereby ensuring its security.

Device addition

Adding a device consists of transferring the symmetric key from an existing device to a new device. The new device identity and a string identifier is also registered with the identity provider in the process.

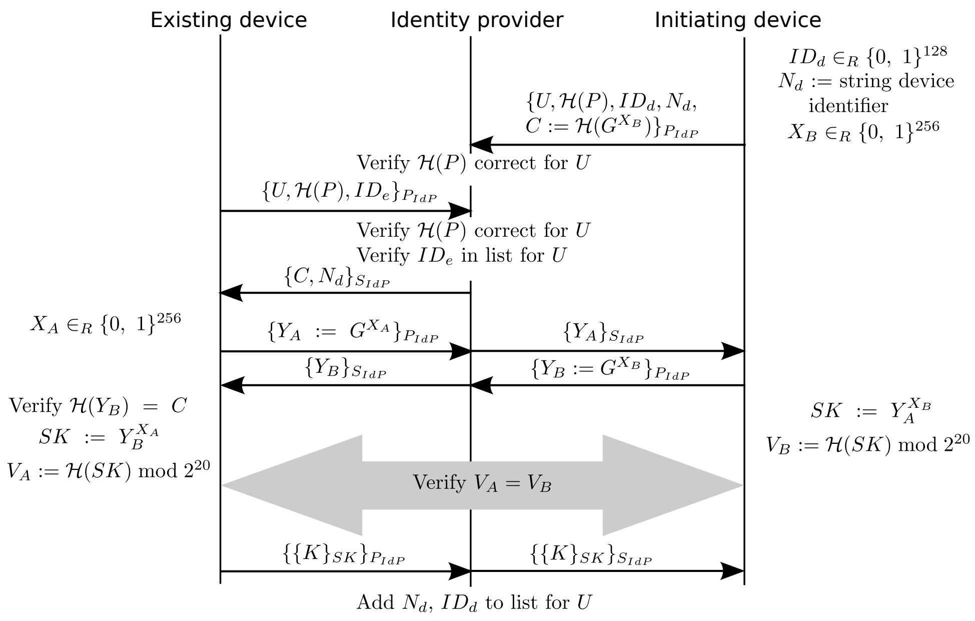

Existing device
Identity provider
Initiating device
$ID_d \in_R \{0,\ 1\}^{128}$
$N_d :=$ string device identifier
$X_B \in_R \{0,\ 1\}^{256}$
$\{U, \mathcal{H}(P), ID_d, N_d, C := \mathcal{H}(G^{X_B})\}_{P_{IdP}}$
Verify $\mathcal{H}(P)$ correct for U
$\{U, \mathcal{H}(P), ID_e\}_{P_{IdP}}$
Verify $\mathcal{H}(P)$ correct for U
Verify ID_e in list for U
$\{C, N_d\}_{S_{IdP}}$
$X_A \in_R \{0,\ 1\}^{256}$
$\{Y_A := G^{X_A}\}_{P_{IdP}}$
$\{Y_A\}_{S_{IdP}}$
$\{Y_B\}_{S_{IdP}}$
$\{Y_B := G^{X_B}\}_{P_{IdP}}$
Verify $\mathcal{H}(Y_B) = C$
$SK := Y_B^{X_A}$
$V_A := \mathcal{H}(SK) \bmod 2^{20}$
$SK := Y_A^{X_B}$
$V_B := \mathcal{H}(SK) \bmod 2^{20}$
Verify $V_A = V_B$
$\{\{K\}_{SK}\}_{P_{IdP}}$
$\{\{K\}_{SK}\}_{S_{IdP}}$
Add N_d, ID_d to list for U

Figure 7.10: Protocol for adding a device to the identity.

Figure 7.10 shows the protocol which is used to perform this transfer, hence adding a new device to the identity. The protocol is based on a Diffie-Hellman key exchange with minor modifications in order to ensure that the identity provider is unable to perform a man-in-the-middle attack on the exchange. Once the protocol has executed, the symmetric key, PK will have been sent from the existing device to the new device without any other party learning it. The initial commitment ensures that the identity provider is unable to force the value of V_A to equal V_B with a different X_B. The verification of $V_A = V_B$ out of band ensures, with a probability of 1 in 2^{20}, that the identity provider has not changed the values of X_B and C.

The protocol may be considered as five distinct steps.

1. Register a new device with the identity provider. This is performed over an SSL connection for two reasons: (i) To protect the value of ID_d, which is private to the device and identity provider, (ii) To protect the hashed value of the password. A password is set on the account to prevent flooding of device addition requests, and therefore performing a denial of service on an account.

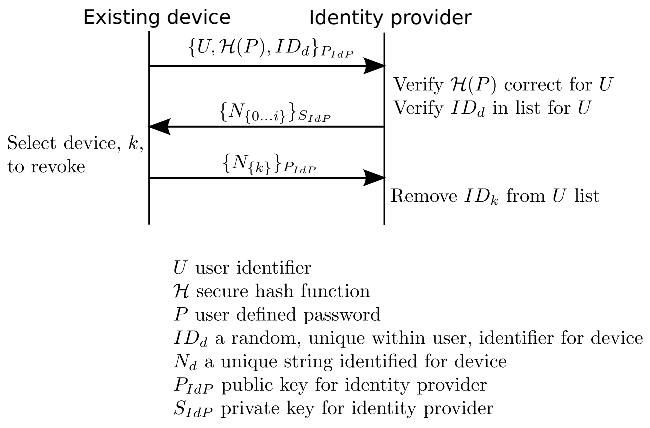
2. Select the new device from a list of waiting devices at the identity provider from an existing device. This process initiates the transfer of the key to new device from an existing one.

3. Perform a Diffie-Hellman exchange with the new device using the identity provider as a proxy. The commitment sent as part of Step 1 is used to verify that the identity provider did not tamper with the exchange. This is required as the verification in Step 4 only operates on 2^{20} bits which could be easily pre-calculated and stored at the identity provider, permitting a man-in-the-middle attack. With a commitment, an attack would need the hashes of G^{X_B} and $G^{X_A X_B}$ would need to collide for the value of X_B for the attacker and the device. Also, the commitment of X_B would need to collide. This is extremely improbable, especially within a time period which goes unnoticed to the user.

4. Verify the Diffie-Hellman exchange was not intercepted, by calculating a short (2^{20} bit) hash of the key and confirm it over a side channel. It is suggested that the value is displayed on the screens of the devices and then accepted by the existing device. This is then able to:

5. Transfer the identity key to the new device, encrypted by the key generated, via the identity provider. At this point the identity provider adds the new device's private identifier, ID_d, to the list of known devices for the user.

Device revocation



Existing device
Identity provider
$\{U, \mathcal{H}(P), ID_d\}_{P_{IdP}}$
Verify $\mathcal{H}(P)$ correct for U
Verify ID_d in list for U
$\{N_{\{0\ldots i\}}\}_{S_{IdP}}$
Select device, k, to revoke
$\{N_{\{k\}}\}_{P_{IdP}}$
Remove ID_k from U list
U user identifier
$\mathcal{H}$ secure hash function
P user defined password
ID_d a random, unique within user, identifier for device
N_d a unique string identified for device
P_{IdP} public key for identity provider
S_{IdP} private key for identity provider

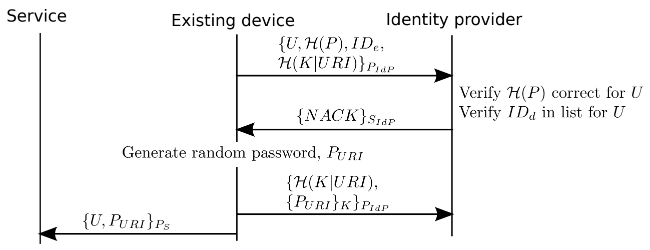
Figure 7.11: Protocol for revoking a device from the identity.

As noted in our trust model, revoking a device is dealt with by the identity provider. In ideal circumstances, a user will have removed their identity key from the device making it impossible to read profile specific data. However, this mechanism stops devices which still contain this private data from obtaining any encrypted data from the identity provider.

The process, as shown in Figure 7.11, is relatively simple in that it follows a similar pattern to all other communications with the identity provider. Initially, the device authenticates with the identity provider over a secure channel in order to protect ID_d. Assuming the device authenticates, a list of all known and active string identifiers for devices are sent to the device. The user simply picks the device they wish to revoke (N_k) and the corresponding value of ID_k from the list of authorised devices. Now any attempt to perform identity functions with the identifier ID_k will be refused by the identity provider. It is easily apparent that if the identity provider were to obtain the user's device, they would have access to the full identity. We envisage this to be a very unlikely event, but it is mitigated by users who properly remove their identity key from devices which they "log out" of.

Service registration

When an authenticated device encounters a service they will attempt to provide their authentication token to it. This is stored encrypted at the identity provider. We see in Figure


Service
Existing device
Identity provider
$\{U, \mathcal{H}(P), ID_e, \mathcal{H}(K|URI)\}_{P_{IdP}}$
Verify $\mathcal{H}(P)$ correct for U
Verify ID_d in list for U
$\{NACK\}_{S_{IdP}}$
Generate random password, P_{URI}
$\{\mathcal{H}(K|URI), \{P_{URI}\}_K\}_{P_{IdP}}$
$\{U, P_{URI}\}_{P_S}$

U user identifier
$\mathcal{H}$ secure hash function
P user defined password
ID_d a random, unique within user, identifier for device
N_d a unique string identified for device
P_{IdP} public key for identity provider
S_{IdP} private key for identity provider
P_S public key for service
K the identity's symmetric key
URI the URI for the service

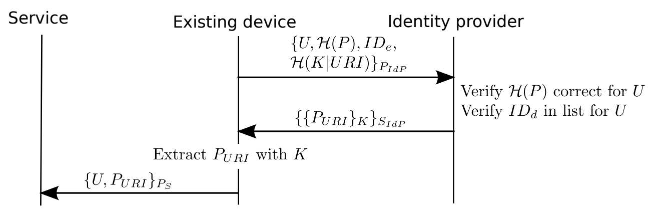
Figure 7.12: Protocol for registering an identity with a service.

7.12 that a standard authentication message is sent with a hash of the URI which is being accessed, and the private key. This is to preserve the privacy of the URIs which are being accessed by the user. Furthermore, we hash the key into the URI in order to ensure privacy is preserved, otherwise identity providers would be able to locate information about specific known services. The identity provider checks the device is part of the identity, and the password is correct. Since the user has not registered with the service a negative response is received from the identity provider.

Upon receiving the negative response, the device creates a password for the service. The new password is stored at the identity provider, encrypted by the identity key and indexed by the hash of the service URI and the identity key. Once the password has been stored, the device performs a standard registration with the service.

Service authentication

Accessing a service which a user has already registered with begins with the same process as with the registration. In this instance, shown in Figure 7.13, the identity provider replies with the encrypted password for the service. The device is able to decrypt the password and use it to authenticate with the service in the usual manner.


Service
Existing device
Identity provider
$\{U, \mathcal{H}(P), ID_e, \mathcal{H}(K|URI)\}_{P_{IdP}}$
Verify $\mathcal{H}(P)$ correct for U
Verify ID_d in list for U
$\{\{P_{URI}\}_K\}_{S_{IdP}}$
Extract P_{URI} with K
$\{U, P_{URI}\}_{P_S}$

U	user identifier
$\mathcal{H}$	secure hash function
P	user defined password
ID_d	a random, unique within user, identifier for device
N_d	a unique string identified for device
P_{IdP}	public key for identity provider
S_{IdP}	private key for identity provider
P_S	public key for service
P_{URI}	the data stored (password) for a given URI
K	the identity's symmetric key

Figure 7.13: Protocol for authenticating with a service.

Profile storage

Profile storage operates in an identical manner to service registration, and authentication where URI is the attribute name. This has two major advantages: simplicity of the design, and indistinguishability of profile and authentication data.

7.3.2 Evaluation

Implementation

We implemented our protocol as a browser extension for Google Chromium, Firefox and as an Android App with the role of the identity provider provided by a Google AppEngine application. These were written in JavaScript and Python respectively. The Android API does not supply any hooks for the browser, and as such it is currently impossible to use the system for logging into sites on an Android device. The application is extremely useful for making the identity portable.

Due to the stateless nature of AppEngine applications we were required to break the protocol down into HTTP requests with the client sending data to initiate the connection and the server responding. As all requests had to originate from the extension (devices) polling loops were required. These happen asynchronously so as not to lock up the browser. The breakdown of the requests are shown in Figure 7.14. Purposely omitted from this figure is

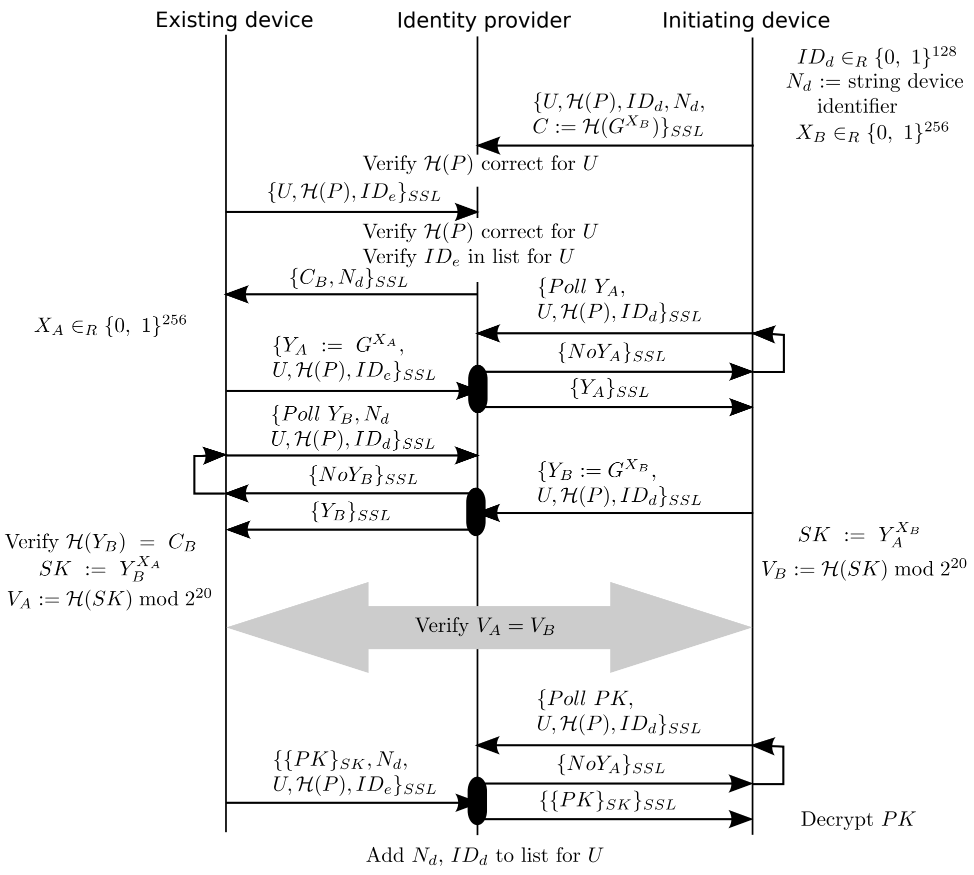

Existing device
Identity provider
Initiating device
$ID_d \in_R \{0,\ 1\}^{128}$
$N_d :=$ string device identifier
$X_B \in_R \{0,\ 1\}^{256}$
$\{U, \mathcal{H}(P), ID_d, N_d, C := \mathcal{H}(G^{X_B})\}_{SSL}$
Verify $\mathcal{H}(P)$ correct for U
$\{U, \mathcal{H}(P), ID_e\}_{SSL}$
Verify $\mathcal{H}(P)$ correct for U
Verify ID_e in list for U
$\{C_B, N_d\}_{SSL}$
$\{Poll\ Y_A, U, \mathcal{H}(P), ID_d\}_{SSL}$
$X_A \in_R \{0,\ 1\}^{256}$
$\{Y_A := G^{X_A}, U, \mathcal{H}(P), ID_e\}_{SSL}$
$\{NoY_A\}_{SSL}$
$\{Y_A\}_{SSL}$
$\{Poll\ Y_B, N_d\ U, \mathcal{H}(P), ID_d\}_{SSL}$
$\{NoY_B\}_{SSL}$
$\{Y_B := G^{X_B}, U, \mathcal{H}(P), ID_d\}_{SSL}$
$\{Y_B\}_{SSL}$
Verify $\mathcal{H}(Y_B) = C_B$
$SK := Y_B^{X_A}$
$V_A := \mathcal{H}(SK) \bmod 2^{20}$
$SK := Y_A^{X_B}$
$V_B := \mathcal{H}(SK) \bmod 2^{20}$
Verify $V_A = V_B$
$\{Poll\ PK, U, \mathcal{H}(P), ID_d\}_{SSL}$
$\{\{PK\}_{SK}, N_d, U, \mathcal{H}(P), ID_e\}_{SSL}$
$\{NoY_A\}_{SSL}$
$\{\{PK\}_{SK}\}_{SSL}$
Decrypt PK
Add N_d, ID_d to list for U

Figure 7.14: A breakdown of HTTP requests to implement the device addition protocol in a "pull" environment.

the verification of $\mathscr{H}(P)$ and ID_d/ID_e other than in the first instance. This is for clarity; however these are checked by the identity provider whenever they are received. It should also be noted at this point that the value N_d is unique for each device; it may be used as an identifier by the existing device to reference the transaction. Furthermore, the identity provider utilises standard practice of limiting requests in order to slow brute force attacks. Finally, SSL is utilised as it is a commonly available API, especially in the HTTP environment.

A standard code base was used for all three implementations meaning that updates are easily rolled out to all three platforms. In Figure 7.15 we see a device (in this instance, the Chrome extension) being added to the identity. Figure 7.16 shows a list of devices in the identity, clicking any of which will cause them to be removed from the list of permitted devices on the identity provider. You will also be able to see the additional feature of changing the user password. This functionality helps to ensure that a compromised device has no access to the identity, as they are unable to add their device to the list of activation pending devices for the identity.

An additional timeout was added for devices pending activation. If they have not initiated an activation in a five minute time they are removed from the pending list. This functionality is simply to clear the list of accidental or failed activation attempts in order to free the device name as they are unique within the identity (as explained earlier).

If the username is not recognised on the system the user is presented with an opportunity to register a new identity, as shown in Figure 7.3.2. Since the details for addition of a device have already been given, all that is required is a confirmation of the password, to ensure it was not mis-typed, and an email address for service related contact. An additional feature of the email address may be as a secondary proof of identity to lock out devices. It cannot be used for recovery of a lost identity, however, owing to the identity provider not having access to the key for user data.

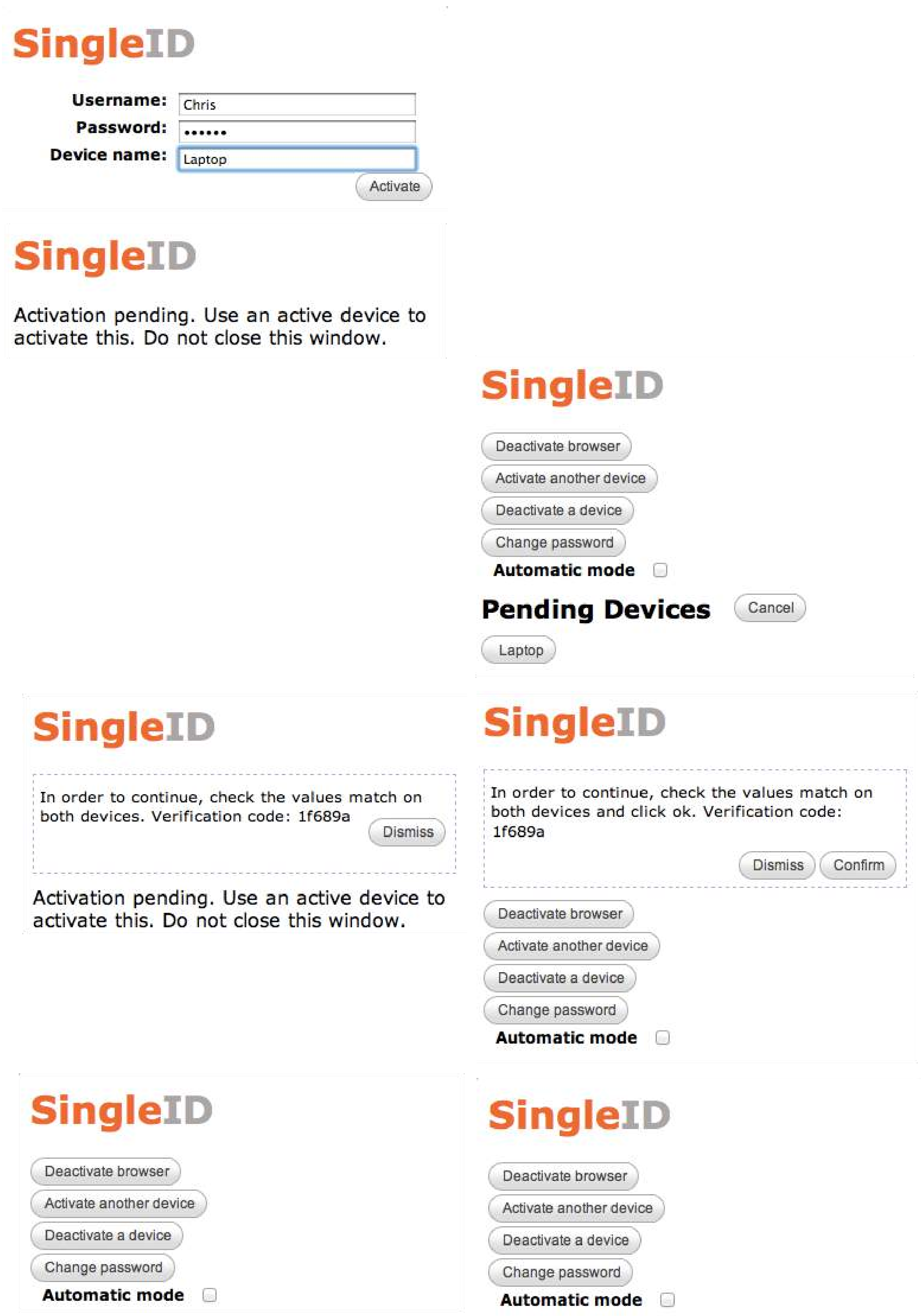
Security analysis

Obtaining unauthorised access profile attributes consists of obtaining both the encrypted profile data, and the private key. Encrypted profile data may be obtained from the identity provider by providing the username, password (or password hash as no nonce is used), and an active device identifier.

Let us first consider how an attacker may obtain the private key. There are two methods available:

1. Intercept a key exchange when a device is being activated
2. Steal an active device which has the private key stored within it.

Our protocol protects from a man-in-the-middle attack on a key exchange by using two commitments. Our first commitment fixes the value of X_B from the start of the protocol. A third party would need to either generate a value for X_B which matches the hash in a timely manner or have a pre-generated table of hashes. Both options are unlikely for a sufficiently large



Username: Chris
Password: ••••••
Device name: Laptop

Activation pending. Use an active device to activate this. Do not close this window.

In order to continue, check the values match on both devices. Verification code: 1f689a

In order to continue, check the values match on both devices and click ok. Verification code: 1f689a

Figure 7.15: The user view of a device addition to the identity on both the existing and new device.

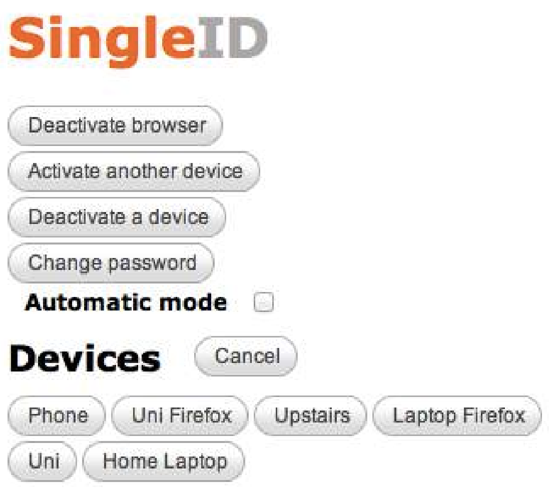
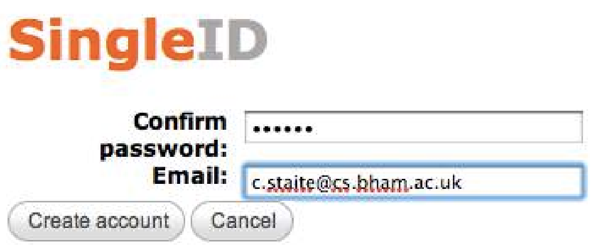


Figure 7.16: Displaying the devices in the identity in order to remove one.



Confirm password: ••••••
Email: c.staite@cs.bham.ac.uk

Figure 7.17: The form for registration of a new identity if the username does not currently exist.

hash (we propose the use of SHA-256). Instead, let us assume the attacker changes the value of C to match their own value of X_B, let us call it X_M. This must be chosen before Y_A is disclosed. The second commitment protects against this attack by placing a 20-bit commitment value on the user devices in the form of a hexadecimal string. The user verifies that these match out of band, meaning an attacker is unable to view or effect this transaction. In order for the attack to succeed $\mathscr{H}(Y_A^{X_M}) \bmod 2^{20}$ must be equal to $\mathscr{H}(Y_A^{X_B}) \bmod 2^{20}$. Since the attacker must choose X_M before they learn Y_A the probability of success is one in 2^{20}. An exchange takes about ten seconds of user interaction, therefore, on average, an attack would take sixty days of continuous user interaction.

The second attack is much simpler as the key is obtained by the attacker as soon as they gain access to the device. We assume that the attacker gains knowledge of the username, password, private key and device identifier at this time. In an ideal scenario, the attacker would not receive this much information as the password should time out. Despite having these attributes, the device identifier is no longer valid for accessing the encrypted profile from the identity provider.

Let us now consider obtaining profile data from the identity provider. The strongest attacker is one that has obtained the username, password and private key in use by stealing a device. Such an attacker simply needs to guess a device identifier to obtain profile attributes. Since the only log of device identifiers is at the identity provider, they simply query the identity provider for a valid device identifier, which is randomly chosen. Each operation requires the device to authenticate, and so the attacker (knowing the username and password) may simply attempt a brute force attack. We protect against this by limiting the number of unauthorised accesses by an IP to five per minute. In the 256-bit identifier space this would require brute force attacks to take, on average, 2^{228} years. Additionally, the add procedure can provide a security risk. When adding a device, only the username and password are authenticated. A new value of ID_d is checked that it is unique within the identity. We limit device additions to two per minute per IP. This has an even stronger protection against brute force attacks. This may cause a slowing for legitimate users, but it is unlikely that users will want to add more than two devices per minute.

Analysis of addition protocol with ProVerif

ProVerif is a software tool for the analysis of protocols for observational equivalence [8]. It is capable of determining whether there exists a trace in which an security requirement is breached. We wrote our protocol as defined in Section 7.3.1. Although ProVerif is unable to warn of the pre-calculation attack, which was the purpose of our additional commitment in the protocol, it was able to verify that there is no other leak of information permitting a third party to retrieve the value of PK. Our ProVerif source is shown in Figure 7.18.

```
type G.
type exponent.

free pk: bitstring [private].

free oc: channel.
free nc: channel.
free pc: channel [private].

fun senc(G, bitstring): bitstring.
reduc forall k:G, m:bitstring;
         sdec(k, senc(k, m)) = m.

const g: G [data].
fun exp(G, exponent): G.
equation forall x:exponent, y:exponent;
                  exp(exp(g,x),y) = exp(exp(g,y),x).

fun h(G): bitstring.

query attacker(pk).

let NewDevice = new xb:exponent;
         out(nc,h(exp(g,xb)));
         in(nc, ya : G);
         out(nc,exp(g,xb));

         let k = exp(ya,xb) in out(pc, h(k));
         in(nc, m : bitstring);

         let pk = sdec(k, m) in 0.

let IdP = in(nc, c : bitstring); out(oc, c);
         in(oc, ya : G); out(nc, ya);
         in(nc, yb : G); out(oc, yb);
         in(oc, epk : bitstring); out(nc, epk).

let ExistingDevice = new xa:exponent;
         in(oc, c : bitstring);
         out(oc, exp(g,xa));
         in(oc, yb : G);
         in(pc, tk : bitstring);

         let k = exp(yb,xa) in
                  if tk = h(k) && h(yb) = c then
                           out(oc, senc(k,pk)); 0
                  else
                           0.

process NewDevice | IdP | ExistingDevice
```

Figure 7.18: The add protocol implemented in ProVerif

CHAPTER 8

EVALUATION REVISITED AND DISCUSSION

The most beautiful thing in the world is, precisely, the conjunction of learning and inspiration. Oh, the passion for research and the joy of discovery!
— Wanda Landowska

Let us consider the research questions posed in Section 1.2. The first question asks: How can an identity management architecture be evaluated systematically? Chapter 3 guided us through the evolution of identity management. This was ultimately used in Chapter 2 to answer question 1a: What are the requirements of an identity management architecture? We described the three stakeholders in identity management: Individuals, Identity providers and Services. Each has their own needs and requirements from an IdM architecture.

In Chapter 5 we sought to answer the final sub-questions using existing methods and literature. Question 1b asks: which metrics need to be evaluated? Followed by 1c: what method should be followed to perform systematic evaluation? To attempt to answer these questions, a systematic literature review was performed in order to seek out possible existing answers. The conclusion was that there were no holistic evaluation methods, and all existing literature provided a small subset of, often incomplete, metrics.

Using generic methods also failed to perform a systematic evaluation, with a classroom study utilising ATAM failing to produce a consistent result. Instead, the results of both the previous research located with the systematic literature review, and those from the ATAM evaluation were combined. These produced a holistic and systematic evaluation methodology for identity management. We discuss the advantages of this and answer question 1d in Section 8.1.

Now, let us consider question two: What improvements may be made to identity management architectures? 2a asks: Where should the effort toward improvements be focused? It was noted in Chapter 3 that existing developments have been focused on the needs of the services. Recent developments have provided a little gain in usability, but this could be furthered. Chapter 2 answers (2b) Which areas of identity management are important to which parties? The importance of security, privacy and usability for the end-user, and the ease of user management and desire to attract users from the perspective of the service provider.

In Chapter 7 a new identity management architecture was derived from the evaluation of service based identity and identity provider based identity. In doing so, it answers question 2c: How can such systematic evaluation inform the design of a new identity management architecture? This is further reflected upon in Section 8.1. Chapter 7 also provides security evaluations of both implementations.

Finally, question 2d: How may improvements made in architectural design be implemented? Two such implementations are described in Chapter 7. Their ability to meet the architectural description of device based identity is evaluated in Section 8.2. A summary of all identity management architectures evaluated in this thesis concludes this chapter in Section 8.3. This provides an overview of the improvements and trade-offs made.

8.1 Identity architecture evaluation

Let us consider research question 1d: what are the benefits of a systematic analysis? The evaluation method given in Chapter 5 was presented with two goals. The first, to combine existing evaluation in the area of IdM into a single methodology, and second, to provide guidance for the future development of IdM. Section 7.1 attempted to utilise this property of the evaluation performed over SB and IdPB implementations, in order to derive a new IdM architecture.

With regard to the first goal, the evaluation method is able to produce a reasonable description of the differences between two IdM architectures from a high level description. Given implementations it provides the areas which may be focused on for quantitative analysis. As shown by the systematic literature review, this is the first method which is capable of performing this trade-off analysis over such a wide spectrum of metrics. Furthermore, the classroom experiment demonstrated the flaw of generic methods of expert knowledge and bias. Due to the boolean statements given in the evaluation methodology proposed, this personal bias or lack of knowledge in a specific area cannot affect the result.

The boolean nature of the metrics causes issues where different versions exist, optional extensions may be used or where the implementation specifics are uncertain. The results permit this uncertainty as a minimum and a maximum value may be given. This does not undermine the system as minimum levels may be compared giving insight to the changes between architectures. Maximum values are also useful in order to allow developers to focus their attention in the areas which are susceptible to lacking performance. For example, implementing OpenID, it is possible to see in Figure 5.7 that profile availability should be considered leading to the implementation of one of the profile extensions, SReg or AX.

The ultimate aim of the analysis was to produce a simple holistic review of an identity architecture, integrating the expert knowledge acquired from many sources. In doing so, the comparison of these evaluations permits decisions to be made about which architecture suits a particular use-case. Until this point, evaluations have mostly focused on the results shown in the metrics. Information about the underlying reasons for these changes are visible by looking at the component distribution throughout the identity systems. By considering the changes made, the secondary goal of developing an identity architecture that makes improvements in other areas is possible. The systematic evaluation informs the design by determining changes made to architectural components and the qualities elicited by the entire system. Through deduction, further changes to the high-level architecture may be posed in an attempt to specifically target areas for improvement. In Chapter 7 the architecture is modified from single sign-on in order to increase usability, privacy and security.

Considering Figures 5.2 and 5.3, it is clear that the addition of the identity provider, removing the *Authentication Provider* and *Credential Store* from the service, increases user simplicity due to the single username and password. The evaluations in Figures 5.4 and 5.6, demonstrate the loss in security, bandwidth and the likely loss in privacy afforded by this. In order to regain security and privacy it was proposed in Chapter 7 that moving the *Authentication Provider* to the device may be a better solution. This was evaluated at an early stage to demonstrate this possibility, which was confirmed in Figure 7.2. As an added bonus DB identity performed much better in availability, security, privacy, authenticity and freshness over OpenID, as shown in Figure 7.3.

Given this result from the new architecture, two implementations were derived in order to attempt to provide the features described. These implementations are described in Sections 7.2 and 7.2. In order to determine whether the implementations coincide with the evaluation of the initial architecture, a full evaluation of them will be performed in the following section.

8.2 Device based identity

Chapter 7 introduced DB IdM and performed an evaluation on the architectural design of the framework. Due to the lack of implementation specific details, this evaluation proved to contain a wide spectrum of values. Post-evaluation, two implementations have been proposed to suit the architecture. In order to evaluate how well the methodology performed and how well the implementations performed, it is logical to perform the evaluation again on the two implementations.

8.2.1 Type one

First to be evaluated is the first implementation of DB IdM, based on group signature, described in Section 7.2.

Authentication user simplicity (4)

Through the use of signatures users authenticate with services using a single public key. The provision of this signature is through the device interface which is common across services. This gives a score of 4, a value of $\frac{4}{4} * 10 = 10.0$.

Authentication computational overhead (9i)

The group signature scheme used utilises proof of knowledge calculations for authentication and attribute access. Attributes are transferred utilising symmetric encryption to secure them. This gives a score of 4, and being inverse a value of $10 - (\frac{4}{9} * 10) = 5.6$.

Authentication bandwidth (5i)

If anonymous authentication is utilised then the device must request the location of the attributes from the identity provider and request the data from the service. Otherwise, this is only required at registration. This gives a score between 1 and 3 and value of between 4.0 and 8.0.

Authentication availability (4)

Similarly to other open single sign-on systems type-one DB supports the use of multiple identity providers. The group signature scheme also permits users to utilise any active device to authenticate. Giving a score of 2 and a value of 5.0.

Authentication privacy (5)

All storage is at the service provider in this architecture. However not all information is stored at all service providers. Transmission from service to service, or via the device prevents the identity provider from having access to any profile attributes. The request for the public key from the identity provider may permit logging of service used. However, it may also be passed from the device maintaining privacy. As the service needs to be able to verify the public key visits may be linked. The user can otherwise remain anonymous. This gives a score between 2 and 3 and values between 4.0 and 6.0.

Authentication security (8)

Due to the nature of the authentication, there are no credentials. It is as if they were encrypted. The signature scheme also simulates a secure high-entropy key for authentication. Due to the use of the trusted external entity, the identity provider is unable to authenticate without notifying the user. This gives a score of 5 and a value of 6.3.

Profile availability (8)

A centralised lookup of profile attributes permits a common interface for attribute retrieval. However, this is a specialised protocol which is unable to support any other protocols. An exchange of attributes may be requested by the user to transfer profile data between services. This gives a score of 3, a value of 3.8.

Profile security (6)

Transferred attributes are encrypted utilising a symmetric key during transmission. Due to storage at service providers, it is unlikely that the attributes are encrypted in their storage location. As the identity provider cannot request data from services the profile is unreadable by the identity provider. This gives a score of 5, a value of 8.3.

Profile privacy (7)

Attributes may be sent via the device, and therefore the source may be hidden. The attributes may also be requested individually by the request of the user. Privacy policies may be utilised in the device software. A score between 2 and 5 may be awarded, giving values between 2.9

and 7.1.

Profile authenticity (2)

There is no support for automated verification of attributes as the device can not be trusted to perform as such. However, attributes obtained from other sources may be certified and passed through. This gives a value of 5.0.

Profile access speed (4i)

Each request for an identity requires the construction and signature of a request message sent to the relevant service. This gives the worst possible score of 4 and value of 0.0.

Profile computational overhead (9i)

Profile transfer operates in the same manner as authentication, with signed messages. Therefore the overhead is the same, a score of 4 and value of 5.6.

Profile freshness (16)

The identity provider maintains a database of all profile storage locations. This permits the user to obtain any attribute and push updates to them. It is therefore unnecessary for the services to provide an interface for the maintenance of the profile. This gives a score of 10 and value of 6.3.

8.2.2 Type two

In the manner which you might expect, the evaluation of the second implementation, described in Section 7.3, is also given.

Authentication user simplicity (4)

As before, the user utilises a username to identify themselves and their identity provider. Providing the extensions are available on the service, a common authentication interface exists. Otherwise, a fall-back will be required which utilises the interface of the service provider. No credentials need to be remembered, as they are stored at the identity provider. This gives a score of 4 and value of 10.0.

Authentication computational overhead (9i)

Identity attributes, including authentication credentials are stored at the identity provider utilising symmetric encryption. Optionally, HTTPS may be used to verify the identity provider. This gives a value between 1 and 3, a score between 6.7 and 8.9.

Authentication bandwidth (5i)

Services should maintain a session after authentication. This authentication is performed by retrieval of a set of credentials for the service from the identity provider at authentication. This gives a score of 1, a score of 8.0.

Authentication availability (4)

Once again, the use of different identity providers is supported and multiple devices may be used to authenticate. Different protocols may also be supported with the generic store of authentication data. Utilising the central storage, authentication may also be passed between devices. This scores the maximum of 4, a value of 10.0.

Authentication privacy (5)

The attributes and credentials are encrypted at the identity provider making them unreadable. Accordingly, they need to be decrypted at the device, making usage information inaccessible to the identity provider also. Users are able to remain anonymous to the service also as the username need not be provided, authentication with the identity provider being performed by the device. This gives a score of 3, a value of 6.0.

Authentication security (8)

Credentials are transmitted as they are stored, encrypted. If HTTPS is utilised then the endpoints are also verified. Authentication is performed through the use of the decryption key for the credentials. As the identity provider does not have access to the key it is unable to decrypt the credentials. This gives a score between 5 and 7, values of 6.3 to 8.8.

Profile availability (8)

Attributes may be stored centrally; however this is not a requirement. Requesting attributes and handling requests is dealt with by the device software, and therefore is capable of supporting multiple protocols. The encrypted attributes are sent to, and stored at, the identity provider. This gives a score of 5 and a value of 6.3.

Profile security (6)

Attributes are stored encrypted at the identity provider, and are sent in this form. Connection to the identity provider is optionally through HTTPS, verifying the destination. This gives a score between 5 and 6, a value between 8.3 and 10.0.

Profile privacy (7)

Privacy policies may be implemented within the device software, where attributes are distributed. Since attributes are decrypted at the device, the source of the attributes is hidden. Any kind of attribute may be stored including certified ones. Attributes are stored separately at the identity provider, and passed via the device for approval. This gives a score between 4 and 5, and a value between 5.7 and 7.1.

Profile authenticity (2)

There is no support for automated verification of attributes as the device can not be trusted to perform this. However, attributes obtained from other sources may be certified and passed through. This gives a value of 5.0.

Profile access speed (4i)

Attributes are stored at the identity provider, but may be requested in bulk by the device, giving a score of 1, a value of 7.5.

Profile computational overhead (9i)

Requesting attributes from the identity provider is the same as requesting credentials, giving the same score of between 6.7 and 8.9.

Profile freshness (16)

Profile attributes are stored centrally, and may be viewed and modified. However, a log of where all attributes are sent is not made. This means that a similar situation to SB is maintained. This gives a score of 12, a value of 7.5.

8.2.3 Comparison

The values given for authentication and profile metrics are shown in Figures 8.2 and 8.1 respectively for type-one. Type-two are shown for authentication and profile metrics in Figures 8.4 and 8.3.

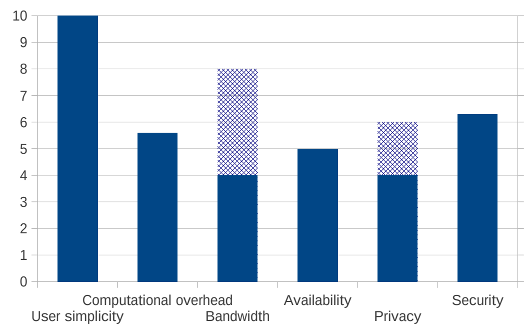


Figure 8.1: Qualitative analysis of type one device based authentication

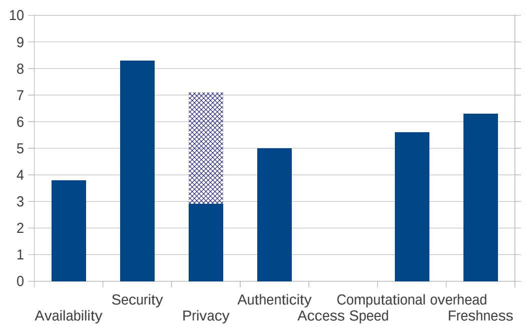


Figure 8.2: Qualitative analysis of type one device based profile data

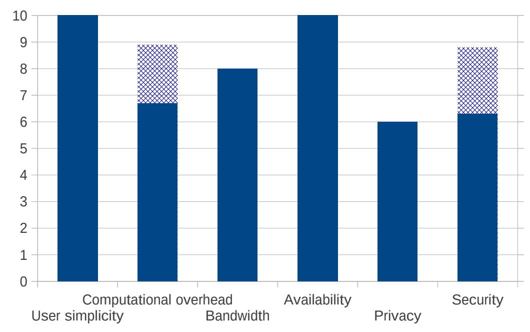


Figure 8.3: Qualitative analysis of type two device based authentication

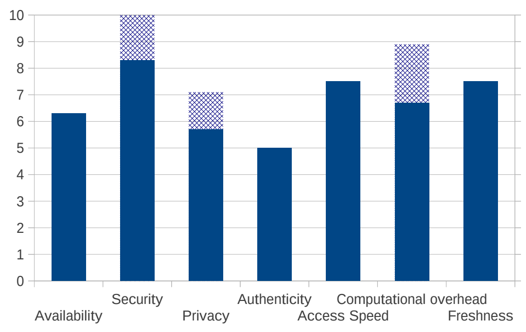


Figure 8.4: Qualitative analysis of type two device based profile data

	DB prediction score		DB prediction value		Impl. 1 score		Impl. 1 value		Impl. 2 score		Impl. 2 value	
Metric	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
Authentication user simplicity	2	4	5.0	10.0	4	4	10.0	10.0	4	4	10.0	10.0
Authentication computational overhead	3	9	0	6.7	4	4	5.6	5.6	1	3	6.7	8.9
Authentication bandwidth	1	4	2.0	8.0	1	3	4.0	8.0	1	1	8.0	8.0
Authentication availability	2	4	5.0	10.0	2	2	5.0	5.0	4	4	10.0	10.0
Authentication privacy	1	4	2.0	8.0	2	3	4.0	6.0	3	3	6.0	6.0
Authentication security	5	7	6.3	8.8	5	5	6.3	6.3	5	7	6.3	8.8
Profile availability	3	5	3.8	6.3	3	3	3.8	3.8	5	5	6.3	6.3
Profile security	4	6	6.7	10.0	5	5	8.3	8.3	5	6	8.3	10.0
Profile privacy	3	5	4.3	7.1	2	5	2.9	7.1	4	5	5.7	7.1
Profile authenticity	1	1	5.0	5.0	1	1	5.0	5.0	1	1	5.0	5.0
Profile access speed	1	3	2.5	7.5	4	4	0.0	0.0	1	1	7.5	7.5
Profile computational overhead	3	9	0.0	6.7	4	4	5.6	5.6	1	3	6.7	8.9
Profile freshness	5	15	3.1	9.4	10	10	6.3	6.3	12	12	7.5	7.5

Table 8.1: Comparison of device based identity implementations against initial architecture evaluation

8.2.4 Comparison with prediction

In Chapter 7 an evaluation was performed against the high-level overview of a device based identity management architecture. Since then, two different implementations have been developed in an attempt to exemplify the architecture. In the previous section, the two implementations were evaluated, we will now compare these results with the preliminary ones. This is to: (i) determine how well the evaluation method is capable of evaluating architectures without implementations (ii) see how well the two implementations perform with regards to the expected abilities of a device based identity architecture.

Table 8.1 shows the results of the initial evaluation performed on DB IdM in Section 7.1, alongside the results of the implementation evaluations made in Section 8.2. We can see that the implementation results only strayed out of the bounds of the initial prediction four times: on the computational overhead for both authentication and profile for the second implementation, and the profile privacy and access speed for the first implementation. The overhead reduction was due to the need for HTTPS not being required in certain circumstances (mainly due to backwards compatibility). The privacy suffered slightly due to the ability to send attributes directly between services. The access speed is also worse than ex-

	SB score		SB value		OpenID score		OpenID value		Impl. 1 score		Impl. 1 value		Impl. 2 score		Impl. 2 value	
Metric	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
Authentication user simplicity	1	2	2.5	5.0	2	3	5.0	7.5	4	4	10.0	10.0	4	4	10.0	10.0
Authentication computational overhead	3	3	6.7	6.7	3	3	6.7	6.7	4	4	5.6	5.6	1	3	6.7	8.9
Authentication bandwidth	0	0	10.0	10.0	1	1	8.0	8.0	1	3	4.0	8.0	1	1	8.0	8.0
Authentication availability	0	0	0.0	0.0	2	2	5.0	5.0	2	2	5.0	5.0	4	4	10.0	10.0
Authentication privacy	3	3	6.0	6.0	2	4	4.0	8.0	2	3	4.0	6.0	3	3	6.0	6.0
Authentication security	1	5	1.3	6.3	0	4	0.0	5.0	5	5	6.3	6.3	5	7	6.3	8.8
Profile availability	0	0	0.0	0.0	0	2	0.0	2.5	3	3	3.8	3.8	5	5	6.3	6.3
Profile security	1	4	1.7	6.7	0	3	0.0	5.0	5	5	8.3	8.3	5	6	8.3	10.0
Profile privacy	3	3	4.3	4.3	3	3	4.3	4.3	2	5	2.9	7.1	4	5	5.7	7.1
Profile authenticity	0	1	0.0	5.0	0	1	0.0	5.0	1	1	5.0	5.0	1	1	5.0	5.0
Profile access speed	1	1	7.5	7.5	1	1	7.5	7.5	4	4	0.0	0.0	1	1	7.5	7.5
Profile computational overhead	0	3	6.7	10.0	3	3	6.7	6.7	4	4	5.6	5.6	1	3	6.7	8.9
Profile freshness	3	5	1.9	3.1	3	5	1.9	7.5	10	10	6.3	6.3	12	12	7.5	7.5

Table 8.2: Evaluation of service based identity management

pected due to the requirement of signing messages on the device.

It is also possible to see that the second implementation performs better on every metric compared to the first. However, this is due to trust being placed in the identity provider to maintain security of profile and credential information against devices removed from the identity.

The initial evaluation determined that the major qualities of DB IdM would be user simplicity and security with a possible trade-off in the areas of computational overhead and bandwidth. These qualities were demonstrated in the two implementations, both providing high levels of usability and security relative to existing implementations. In the next section we will consider the evaluations of all four implementations considered in this thesis.

8.3 Summary of evaluation results

Table 8.2 shows the results of all the evaluations performed throughout this thesis. We present it here to display the differences between the different implementations. In Chapter 3, the difference between service based identity management and user-centric identity management was noted to be a gain in usability. We also noted that existing implementations reduced the privacy afforded to users due to the identity provider having access to their identities. This trade-off is visible in the *authentication user simplicity* and *authentication privacy* metrics in the table.

The purpose of the further evolution of identity management, stated in Chapter 7, was to improve usability and to reverse this downward trend in privacy. To facilitate this, we can see that the computational overhead has suffered. In the case of the first device based implementation, profile access speed and privacy were also affected. The trust model of the second implementation facilitates the better performance in these areas. Despite this, the trust afforded to the identity provider is much less than in existing IdPB mechanisms. The second implementation is even backwards compatible with existing IdM architectures due to its centralised profile storage.

These results have shown that the evaluation has provided motivation for a better architecture. It has also verified the areas in which improvements have been made and the high-level overview has provided insight into the differences in trust afforded to the components in the system.

CHAPTER 9

CONCLUSIONS

Intellectual growth should commence at birth,

and cease only at death.

— Albert Einstein

9.1 What have we learnt?

In the introduction the problem of identity management was posed - specifically, the issue of usability from the point of view of the user. This was prompted by the lack of development in the area. In order to address this issue an approach to evaluate existing developments was taken. This is due to the lack of focus and direction in the development of new ideas in the area of identity management architectures and implementation.

For the evaluation, a search was performed in order to locate existing research which evaluates identity management architectures and implementations. It was found during this review of existing literature, that only small parts of the problem of identity management are considered by each of the approaches. To supplement this work, a more generic evaluation approach was taken to attempt to determine whether a holistic evaluation may be provided by them. It was found that the generic evaluation did not perform optimally due to the lack of expert knowledge in all fields, and the different perspectives, such as those of the user and service provider.

Combining the results of the literature review and the generic trade-off method, our first contribution is described, a new methodology for evaluating identity management architectures and implementations was derived. The new methodology set a number of weighted requirements for identity management systems. Despite utilising a different form of evaluation technique, it managed to encompass most of the evaluation criteria of existing techniques. The purpose of this evaluation is not to state that one architecture or implementation is better than another. Rather, it provides an overview of the improvements made in certain areas versus the trade-offs made in others. Thanks to a high level overview, it is also possible to make educated guesses as to which components cause improvements (and indeed detriment) in specific metrics.

The methodology developed provides a simple and intuitive way to evaluate IdM architectures. It many be expanded with new features without invalidating existing evaluations. The grouping and scoring system facilitates these extensions, therefore ensuring completeness may be maintained with further developments in the field.

The second contribution of this thesis utilised the metrics developed to evaluate two existing identity management architectures. Specifically, “traditional” SB IdM and “single sign-on” IdPB IdM. The improvement in usability is visibile, although it is clear that it could be improved further.

Through the evaluation of existing architectures, our third contibution of DB IdM was proposed. The purpose of this new architecture is to improve the usability of authentication over existing developments, especially for mobile users. Also, it aims to provide support for equally more usable profile management, an area which had been mostly ignored by previous developments. Evaluation of this architecture before an attempt at implementing it was made showed that usability was indeed increased. However, a possible trade-off was bandwidth and computation. With increasing Internet and processor speeds, this seems reasonable.

The fourth contribution is the implementation of the DB architecture as conceived. This proved rather tricky - the issue lies with the credentials residing at the identity provider but the authentication residing at the user device (of which there may be many). This centralised identity with decentralised users is reminiscent of group signature schemes, and therefore the use of one was proposed. Unfortunately, the management of the public key became an issue, requiring the inclusion of a fourth type of system in the architecture. A trusted node which provided a count in the public key ensured that the identity provider could not make changes to the identity without the users knowledge. The simplicity of this trusted node meant that it could be implemented using trusted hardware, or an external entity. Also, the generic nature of the signature scheme permitted it to be used for signing messages other than authentication. This permitted the idea of profile attribute transfer between services in order to make use of the decentralised profile, the benefit to the service providers being that profile attributes are more likely to be collected and maintained, if centrally tracked.

Following on from this implementation, a further contribution of a second novel implementation is proposed. Instead of utilising an external trusted entity, the identity provider is awarded a certain amount of trust. This model trusts the identity provider not to release credentials or attributes to untrusted devices. However, it does not trust it with the attributes directly. Instead it uses encrypted attributes. Through this trust model the system remains secure so long as the identity provider does not steal a user's device. Due to the generic nature of the centralised storage, attributes and credentials may be stored and retrieved by any device. This has many different uses, but the centralisation of the well known utility of a browser based form filler works very well. With some additional HTML tags or attributes on existing tags for websites, automated authentication is also possible.

The evaluation of the two implementations differed slightly, due to quirks of the implementation. With the group signature version, another system is introduced which alters the result. In the second implementation, the trust model is modified in order to implement the architecture. This shows that the evaluation mechanism is able to indicate how the state-of-the-art may be progressed. However, the restrictions of implementation cannot be considered at such a high level. Instead, prototypes are required in order to provide these insights.

9.2 The future of identity management

DB IdM is not the ultimate answer to the problems faced by users authenticating and managing profiles. Also, the evaluation metrics are likely to be missing criteria, or other metrics may be missing entirely.

To derive the metrics, this thesis was motivated by lessons learnt from a classroom experiment, expert judgement and a systematic literature review into identity management architecture evaluation. As identity management systems evolve, so will the results of the expert judgement. These may be added into the metrics, altering the weighting of existing requirements. Additionally, the systematic literature review could be performed for each of the metrics identified. Undoubtedly, this will uncover some existing literature which was overlooked due to the search terms utilised. The advantage to the methodology of evaluation utilised in this thesis, is that it may be developed and expanded upon without invalidating previous results.

As for future developments in identity management, there will likely be developments which are able to increase usability, security or other metric within most of the architectures. However, the use of a user device rather than an identity provider, to be the core of authentication, will always perform better than an identity provider or SB solution. People are likely to prefer relying on their possessions, rather than trust third parties. Providing that vendors do not take away the freedom of user devices[1] this will always be a more private and secure solution over a distributed set of devices.
In summary, there are a number of future research questions:

- Have all metrics relating to identity management been identified?
- How may one be sure that all statements relating to a metric have been identified?
- Is there a systematic manner to link changes in the metrics to changes in the architectural component distribution?

DB identity is not the ultimate solution however. There is the possibility of utilising personal identifiers (biometrics) to authenticate users without reasonable doubt. However, this still requires a method of centrally storing or accessing profile data to solve the problems of identity management. Currently though, the use of biometric authentication is underdeveloped and has major issues. For one, there is no way to revoke a biological component, for example if someone is able to replicate your fingerprint you cannot change it. Additionally, most biological methods are not private; for example it is easy to gain a fingerprint of a glass or DNA from a dropped hair. For now, utilising devices that you own is the best way to revocably authorise yourself.

[1]as has occurred in a number of instances (the iPad for example)

LIST OF REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000. 41

[2] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer Berlin / Heidelberg, 2000. 94, 110

[3] Mark Atwood, Richard M. Conlan, Blaine Cook, Leah Culver, Kellan Elliott-McCrea, Larry Halff, Eran Hammer-Lahav, Ben Laurie, Chris Messina, John Panzer, Sam Quigley, David Recordon, Eran Sandler, Jonathan Sergent, Todd Sieling, Brian Slesinsky, and Andy Smith. OAuth Core 1.0. Technical report, Technical report, 2007. 41

[4] Carolyn Axtell, Donald Hislop, and Steve Whittaker. Mobile technologies in mobile spaces: Findings from the context of train travel. *International Journal of Human-Computer Studies*, 66(12):902 – 915, 2008. Mobile human-computer interaction. 69

[5] Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 40–46, New York, NY, USA, 2005. ACM. 39

[6] Endre Bangerter, Jan Camenisch, and Anna Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *Security Protocols*, Lecture Notes in Computer Science, pages 20–42. Springer Berlin / Heidelberg, 2006. 38, 42, 75

[7] Marc Barisch. Design and evaluation of an architecture for ubiquitous user authentication based on identity management systems. In *10th IEEE Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 863–872. IEEE, 2011. 61, 63, 64, 72, 74

[8] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW-14)*, volume 96, 2001. 131

[9] Marina Blanton. Online subscriptions with anonymous access. In *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 217–227, New York, NY, USA, 2008. ACM. 39

[10] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer Berlin / Heidelberg, 2004. 95

[11] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer Berlin / Heidelberg, 2004. 94

[12] Duncan A. Buell and Ravi Sandhu. Identity management. *Internet Computing, IEEE*, 7(6):26–28, Nov.-Dec. 2003. 37

[13] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In *Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 120–133. Springer Berlin / Heidelberg, 2005. 94

[14] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *Public Key Cryptography - PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 481–500. Springer Berlin / Heidelberg, 2009. 96

[15] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 101–120. Springer Berlin / Heidelberg, 2002. 94, 96, 110

[16] Jan Camenisch, Abhi Shelat, Dieter Sommer, Simone Fischer-Hübner, Marit Hansen, Henry Krasemann, Gérard Lacoste, Ronold Leenes, and Jimmy Tseng. Privacy and identity management for everyone. In *DIM '05: Proceedings of the 2005 workshop on Digital identity management*, pages 20–27, New York, NY, USA, 2005. ACM. 41

[17] Jean Camp. Digital identity. *Technology and Society Magazine, IEEE*, 23(3):34–41, Fall 2004. 36

[18] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Assertions and protocols for the oasis security assertion markup language (SAML) v2.0, March 2005. 37

[19] David Chaum and Eugène Van Heyst. Group signatures. In *EUROCRYPT'91: Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, pages 257–265, Berlin, Heidelberg, 1991. Springer-Verlag. 94

[20] Winnie Cheng, Jun Li, Keith Moore, and Alan H. Karp. A customer-centric privacy protection framework for mobile service-oriented architectures. In *Services Computing, 2008. SCC '08. IEEE International Conference on*, volume 2, pages 13–20, Honolulu, HI, July 2008. 42

[21] Kari Chopra and William A. Wallace. Trust in electronic environments. page 10, Jan. 2003. 24

[22] Premkumar T. Devanbu and Stuart Stubblebine. Software engineering for security: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 227–239, New York, NY, USA, 2000. ACM. 25

[23] W. Diffie and M. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, 1976. 97

[24] Marlena Erdos and Scott Cantor. Shibboleth architecture DRAFT v05, May 2002. 5, 37

[25] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Berlin / Heidelberg, 1987. 94

[26] Simone Fischer-Hübner, Hans Hedbom, Marit Hansen, and Peter Keller. PRIME Framework v3, March 2008. 41

[27] Shoichirou Fujiwara, Takaaki Komura, and Yasuo Okabe. A privacy oriented extension of attribute exchange in shibboleth. *Applications and the Internet Workshops, IEEE/IPSJ International Symposium on*, page 28, 2007. 38

[28] Uwe Glässer and Mona Vajihollahi. Identity management architecture. In *IEEE International Conference on Intelligence and Security Informatics, 2008, IEEE ISI 2008*, pages 137–144, Taipei, Taiwan, 2008. 56

[29] Joshua T. Goodman, Carl M. Kadie, David M. Chickering, Donald E. Bradford, and Dane A. Glasgow. Intelligent autofill. Patent, 2007. US 7254569. 36

[30] Jens Groth. Fully anonymous group signatures without random oracles. In *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer Berlin / Heidelberg, 2008. 94, 95

[31] Ali N. Haidar and Ali E. Abdallah. Comparison and evaluation of identity management in three architectures for virtual organizations. In *Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security*, pages 21–26, Washington, DC, USA, 2008. IEEE Computer Society. 61, 63, 64, 70, 72

[32] Jonna Häkkilä and Ilkka Känsälä. Role based privacy applied to context-aware mobile

applications. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 6, pages 5467–5472, October 2004. 42

[33] D. Hardt, J. Bufu, and J. Hoyt. Openid atrribute exchange 1.0 - final. `http://openid.net/specs/openid-attribute-exchange-1_0.html`, December 2007. 40

[34] Cormac Herley and Paul C. van Oorschot. A research agenda acknowledging the persistence of passwords. *Security & Privacy, IEEE*, (99). 6

[35] Jason I. Hong and James A. Landay. An architecture for privacy-sensitive ubiquitous computing. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189, New York, NY, USA, 2004. ACM. 42

[36] Toshiyuki Isshiki, Kengo Mori, Kazue Sako, Isamu Teranishi, and Shoko Yonezawa. Using group signatures for identity management and its implementation. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 73–78, New York, NY, USA, 2006. ACM. 107

[37] Tor Anders Johansen, Ivar Jørstad, and Do van Thanh. Identity management in mobile ubiquitous environments. In *ICIMP '08: Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection*, pages 178–183, Washington, DC, USA, 2008. IEEE Computer Society. 40

[38] Audun Jøsang. Prospectives for online trust management. *Submitted to IEEE Transactions on Knowledge and Data Engineering*, 2012. 24

[39] Audun Jøsang and Simon Pope. User centric identity management. In *Asia Pacific Information Technology Security Conference, AusCERT2005*, pages 77–89, 2005. 10, 37, 101

[40] Yogesh Kalyani and Carlisle Adams. Privacy negotiation using a mobile agent. In *Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on*, pages 628–633, Ottawa, Ont., May 2006. 42

[41] Rick Kazman, Mario Barbacci, Mark Klein, S. Jeromy Carrière, and Steven G. Woods. Experience with performing architecture tradeoff analysis. *Software Engineering, International Conference on*, 0:54, 1999. 29, 46

[42] Aram Khalili, Jonathan Katz, and William A. Arbaugh. Toward secure key distribution in truly ad-hoc networks. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*, pages 342–346, Jan. 2003. 62

[43] Barbara Kitchenham. Procedures for performing systematic reviews. Technical Report TR/SE-0401, Keele University Technical Report, July 2004. 57

[44] Michael Koch and Wolfgang Wörndl. Community-support and identity management. In *Proceedings of the European Conference on Computer-Supported Cooperative Work (ECSCW2001), Bonn, Germany*, pages 319–338, 2001. 100

[45] Ronald Leenes, Jan Schallaböck, and Marit Hansen. PRIME White Paper v3, May 2008. 41

[46] Jim Miller. Who are you, part II: More on the trade-off between information utility and privacy. *IEEE Internet Computing*, 12(6):91–93, Nov.-Dec. 2008. 19, 26, 40

[47] Jim Miller. Who are you? The trade-off between information utility and privacy. *IEEE Internet Computing*, 12(4):93–96, July-Aug. 2008. 4, 19, 21, 26

[48] Jim Miller. One more take on identity. *IEEE Internet Computing*, 13(2):99–101, 2009. 70, 100

[49] Andrew C. Myers and Barbara Liskov. Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering Methodologies*, 9(4):410–442, 2000. 36

[50] Hyun-Kyung Oh and Seung-Hun Jin. The security limitations of SSO in OpenID. *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, 3:1608–1611, Feb. 2008. 41

[51] John Soren Pettersson, Simone Fischer-Hübner, Ninni Danielsson, Jenny Nilsson, Mike Bergmann, Sebastian Clauss, Thomas Kriegelstein, and Henry Krasemann. Making PRIME usable. In *Proceedings of the 2005 symposium on Usable privacy and security*, pages 53–64. ACM New York, NY, USA, 2005. 25, 41

[52] Birgit Pfitzmann and Michael Waidner. Privacy in browser-based attribute exchange. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 52–62, New York, NY, USA, 2002. ACM. 38, 61, 63, 64, 73

[53] Zulfikar Ramzan and Matthias Ruhl. Protocols for anonymous subscription services. *Unpublished manuscript*, 2000. 39

[54] David Recordon and Drummond Reed. OpenID 2.0: a platform for user-centric identity management. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 11–16, New York, NY, USA, 2006. ACM. 5, 40

[55] Gruia-Catalin Roman, Gian Pietro Picco, and Amy L. Murphy. Software engineering for mobility: a roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 241–258, New York, NY, USA, 2000. ACM. 25

[56] Thomas L. Saaty. How to make a decision: the analytic hierarchy process. *European journal of operational research*, 48(1):9–26, 1990. 68

[57] Luis F. G. Sarmenta, Marten van Dijk, Charles W. O'Donnell, Jonathan Rhodes, and Srinivas Devadas. Virtual monotonic counters and count-limited objects using a TPM without a trusted OS. In *STC '06: Proceedings of the first ACM workshop on Scalable trusted computing*, pages 27–42, New York, NY, USA, 2006. ACM. 95

[58] Frank Schell, Andreas Schaf, Jochen Dinger, and Hannes Hartenstein. Assessing identity and access management systems based on domain-specific performance evaluation. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, WOSP/SIPEW '10, pages 253–254, New York, NY, USA, 2010. ACM. 62, 69, 71

[59] J. Siddiqi, B. Akhgar, M. Naderi, W. Orth, N. Meyer, M. Tuisku, G. Pipan, M. L. Gallego, J. A. Garcia, M. Cecchi, and J. Colin. Secure ict services for mobile and wireless communications: A federated global identity management framework. *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*, pages 351–357, April 2006. 39, 43

[60] Anna Squicciarini, Marco Casasaa Mont, Abhilasha Bhargav-Spantzel, and Elisa Bertino. Automatic compliance of privacy policies in federated digital identity management. pages 89–92, June 2008. 38

[61] Christopher Staite, Rami Bahsoon, and Stephen Wolak. Fine-grained recommendation systems for service attribute exchange. In *Service-Oriented Computing*, volume 5900 of *Lecture Notes in Computer Science*, pages 352–357. Springer Berlin / Heidelberg, 2009. 24

[62] Suriadi Suriadi, Ernest Foo, and Rong Du. Layered identity infrastructure model for identity meta systems. In *AISC '08: Proceedings of the sixth Australasian conference on Information security*, pages 83–92, Darlinghurst, Australia, 2008. Australian Computer Society, Inc. 61, 63, 64, 72

[63] Suriadi Suriadi, Ernest Foo, and Audun Jøsang. A user-centric federated single sign-on system. *Journal of Network and Computer Applications*, 32(2):388 – 401, 2009. 56

[64] Toshihiro Takagi, Takaaki Komura, Shuichi Miyazaki, and Yasuo Okabe. Privacy oriented attribute exchange in shibboleth using magic protocols. pages 293–296, 28 2008-Aug. 1 2008. 38

[65] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11, 1991. 2

[66] Peter White. Identity management architecture: A new direction. In *2008 IEEE 8th International Conference on Computer and Information Technology, CIT 2008*, pages 408–413, Sydney, NSW, Australia, 2008. 6

[67] Juan C. Yelmo, Rubén Trapero, and José M. del Alam. Identity management and web services as service ecosystem drivers in converged networks. 47(3):174–80, March 2009. 43

[68] Seung Yi and Robin Kravets. Composite key management for ad hoc networks. In *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 52–61, 2004. 35

[69] Cai-Nicolas Ziegler and Georg Lausen. Analyzing correlation between trust and user similarity in online communities. In *Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 251–265. Springer Berlin / Heidelberg, 2004. 24