ETH zürich

ETH Library

A Comparative Analysis of Identity Standards for the Internet

Master Thesis

Author(s):
Etter, Kaspar

Publication date:
2013

Permanent link:
https://doi.org/10.3929/ethz-a-010056870

Rights / license:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection.
For more information, please consult the Terms of use.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A Comparative Analysis of Identity Standards for the Internet

Master Thesis

Kaspar Etter
etterka@ethz.ch

Global Information Systems Group, Department of Computer Science
Prof. Dr. Moira C. Norrie, Supervised by Matthias Geel, 19 May 2013

1.0

Contents

1. Preface 3

1.1. Abstract 3
1.2. Acknowledgments 3

2. Introduction 4

2.1. Overview 4
2.2. Objectives 4
2.3. Outline 4
2.4. Related Work 5

3. Background 6

3.1. History 6
3.2. Terminology 7

4. Characterization 8

4.1. Identification 8
4.2. Authentication 8
4.3. Authorization 9
4.4. Accountability 9
4.5. Attributes 9
4.6. Contacts 10
4.7. Classification 10

5. Comparative Study 11

5.1. Standards 11
5.2. Scenario 12
5.3. Criteria 13
5.4. SAML 14
5.5. OpenID 15
5.6. WebID 16
5.7. Virtual ID 18
5.8. Evaluation 19

6. Conclusions 20

6.1. Contributions 20
6.2. Future Work 20
6.3. Semantic Web 21

A. Technologies 22

A.1. Internet 22
A.2. World Wide Web 22
A.3. Cryptography 23
A.4. Vulnerabilities 23

B. Standards 24

B.1. Comparison 24

C. References 27

«Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.»

Antoine de Saint-Exupéry

1. Preface

1.1. Abstract

Many *identity management systems* have been proposed over the last decade to solve the problem of users having to remember too many passwords for online authentication. The present thesis introduces a *characterization* to capture the important aspects of these systems and suggests a new *classification* based on two major characteristics. The acquired features are then utilized to design a *comparative study*, in which I implemented a *decentralized file service* with *SAML*, *OpenID*, *WebID* and my own *Virtual ID*. The *criteria* for the evaluation were grouped into four categories, namely *suitability*, *usability*, *security* and *privacy*. After giving a few *recommendations*, I finally argue why the realization of the *Semantic Web* requires identities and why current single sign-on solutions are not adequate for building the next generation of distributed online services.

1.2. Acknowledgments

I thank **Matthias Geel** for supervising this work and, together with Prof. Dr. **Moira C. Norrie**, for giving me the opportunity to write the master thesis on my own topic. I have appreciated the received freedom and the many fruitful discussions we had so far a lot!

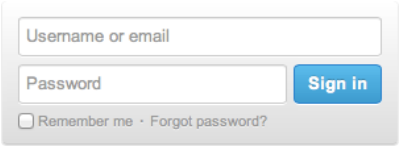
2. Introduction

2.1. Overview

Problem

In today's *Web*, most platforms **authenticate** their users by requiring them to enter their **username** and **password**. This prevalent authentication mechanism has several problems:

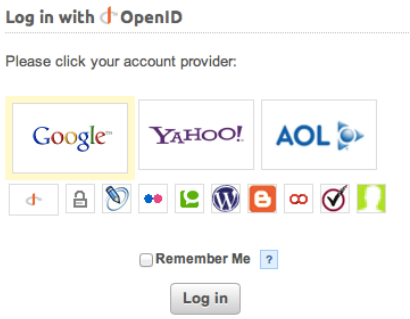
- **Fatigue**: Having to remember a distinct password for each platform imposes an increasingly heavy burden on users.
- **Security**: The randomness of human-chosen secrets is so small that they are vulnerable to guessing and brute force attacks. This situation is further aggravated by the above circumstance, making users choose weak or even similar passwords at different sites with the risk of identity theft.
- **Overhead**: Creating an account and providing the same information at every site cost the users time and nerves.



Login screen of *twitter.com*.

Solution

In order to address these issues, numerous **standards** have been designed that allow to reuse the **identity** respectively the authentication of users. Each of them has its strengths and weaknesses and only the current use case determines the exact requirements for the applicable identity standard. For the sake of an **open** *Web*, however, only non-proprietary standards are considered in the scope of this master thesis.



Login screen of *sourceforge.net*.

2.2. Objectives

Characterize

The first goal of this thesis is to develop a *characterization* of identity standards in order to devise potential *requirements*.

Compare

The second goal is to compare selected identity standards based on concrete *criteria* derived from the requirements.

2.3. Outline

Background

The next chapter covers the **history** of the *Web* and *identity*, and introduces a **terminology** to analyze identity standards.

Characterization

This terminology is then used to explain the **characteristics** of identity standards and to introduce a new **classification**.

Comparative Study

Based on the understandings from the characterization, a comparative study is **designed**, **executed** and **evaluated**.

Conclusions

The main part of this thesis is finished by stating the made **contributions**, possible **future work** and the **overall vision**.

2.4. Related Work

Terminology

When discussing *identity management systems*, it is essential that everyone uses the same name to refer to the same or at least very similar concept. Depending on the author and the context, however, already the word **identity** is used in many different ways. *Camp* [Cam04] has tried to give the various terms around digital identity a uniform meaning, but lacks conceptual clarity and consistency herself. Induced by the English language, I still use the words in similar ways (3.2.). *Clauss* and *Köhntopp* [CK01] introduced the notion of **partial identities** that can be used in different contexts with distinct pseudonyms to prevent linkability among service providers. *Pfitzmann* and *Hansen* [PH08] provide precise definitions for the terms **anonymity**, **unlinkability** and **pseudonymity** (as well as some others) and show how they relate to each other. As these concepts were only used for a single criterion in the study, the explanations given in section 4.1. should suffice. *Bhargav-Spantzel et al.* [BCG06] also propose a **taxonomy** for user-centric identity management and distinguish between *relationship-focused* and *credential-focused* systems for their classification, depending on the longevity of issued tokens.

Requirements

The most famous set of requirements for any digital identity system are the **Laws of Identity** by *Kim Cameron* [Cam05]. As a single identity solution seems unrealistic, he proposed the creation of an *identity metasystem* that can combine multiple technologies in a single framework. The laws (interpreted as scientific observations about the world) mainly focus on the *privacy* of users and their *experience* across various contexts. The most important one for this thesis is the idea of *directed identity*, which is a way of realizing *partial identities* by using a *distinct* identifier for *each* service provider in order to avoid the unnecessary release of so-called correlation handles.

Most other requirement elicitations are conducted in order to show the deficiencies of existent solutions and to illustrate the qualities of the author's own proposal. *Kölsch et al.* [KZR11] describe many soft factors that are important for users, service providers, network operators and developers with regard to *PRIME* (see below). Their emphasis is on the sovereignty of the various stakeholders, which ranges from privacy and trust to open standards. Due to their vague nature though, the criteria are not suited for a comparative analysis. More interesting are the **design goals** by *Perlam* and *Kaufman* [PK08] that have a strong focus on security and usability. Many of their requirements concern the interaction between a user and a service provider as they take the identity provider out of the equation in their suggested user-centric public key infrastructure. They also analyze existing schemes for their practical weaknesses without considering newer standards though. Very detailed metrics for evaluating identity management architectures are presented by *Saite* and *Bahsoon* [SB12], which have executed a systematic literature review and derived a list of profile and authentication features. They have used the metrics to calculate a weighted score for a service-based system and *OpenID*.

Analyses

As in-depth analyses of identity standards and their **security** are not the topic of this master thesis, I just want to mention two worrisome papers from last year: Researchers exploited flaws in the *XML Signature* standard to break eleven of the 14 examined *SAML* frameworks (including *Shibboleth*) [SMS12] and showed vulnerabilities in the signature verification of a lot of *OpenID* implementations (also *OpenID4Java*) [WCW12].

Comparisons

Due to the fact that the identity landscape is changing so fast, many of the papers that compare existing standards are out-of-date very soon. Moreover, a lot of papers just present some standards without actually reviewing or comparing them.

The oldest paper of interest compares *OpenID*, *Shibboleth*, *Liberty* and *CardSpace* on the basis of the *Laws of Identity* and some further criteria regarding their openness and flexibility [HJS09]. In my judgement, the considered **requirements** are either too vague or too technical and do not really cover the possibilities (or lack thereof) of these four identity systems. In addition, many of the explored aspects are indiscriminative.

The paper most closely related to this work was written by *Ferdous* and *Poet* [FP12], who had chosen *CardSpace*, *OpenID*, *Shibboleth*, *Liberty Alliance*, *PRIME* and *OAuth* for their analysis. They group the selected **criteria** into *functional requirements*, *security*, *privacy*, *interoperability*, *trustworthiness*, *liability* and *usability*. Quite a few of their metrics can also be found in my characterization of *identity management systems* (chapter 4.), while others are rather strange and were not even included in their own tabular comparison, which is hardly readable.

Hacket and *Hawkey* [HH12] present an extensive **analysis** of *BrowserID* and *WebID* with a focus on *security*, *privacy* and *usability* requirements. They cover the following six aspects: *compromise*, *phishing*, *recycling*, *tracking*, *robustness* and also *user experience*. Three of them are also included in my study.

Several papers inspect the **privacy** of *identity management systems*. *Birrell* and *Schneider* [BS12] distinguish between three kinds of authentication: *interactive* (direct communication of identity and service providers), *active-client* (all messages are relayed through the client) and *credential-based* (issuance and presentation of credentials are independent). They name this characteristic, when the *identity provider* does not learn about the *service provider*, *undetectability* (I will label it *calling home*) and also examine forms of *unlinkability* and *confidentiality*. (A more detailed analysis of *undetectability* and *unlinkability* can be found in a similar paper by *Veeningen et al.* [VWZ12].)

Initiatives

The *European Union* has funded many research programs to promote **privacy-enhancing technologies**: *Future of Identity in the Information Society* (*FIDIS*), *Privacy and Identity Mgmt. for Europe* (*PRIME*), *PrimeLife*, *Privacy and Identity Management for Community Services* (*PICOS*) and *Attribute-Based Credentials for Trust* (*ABC4Trust*). Likewise, the *American government* started the *National Strategy for Trusted Identities in Cyberspace* (*NSTIC*).

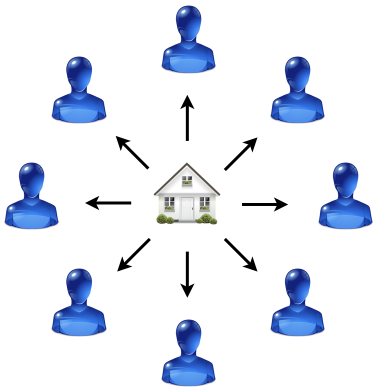
3. Background

3.1. History

Web 1.0

The **World Wide Web** is the popular name for a number of standards, which were created by Tim Berners-Lee at *CERN* in the early 1990s. Among these are the *Hypertext Transfer Protocol* (*HTTP*) to **exchange data**, the *Hypertext Markup Language* (*HTML*) to **structure documents** and the *Uniform Resource Locator* (*URL*) to **reference files** (see A.2. for more information). Together, they form a web of linked pages.

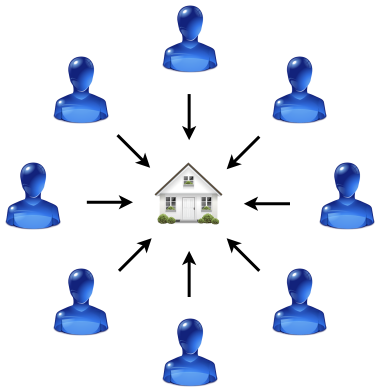
During the first ten years, the *World Wide Web* was mainly used to **publish** documents edited by a webmaster in a **uni-directional** way. This was the golden age of the *homepage*.



Identity 1.0

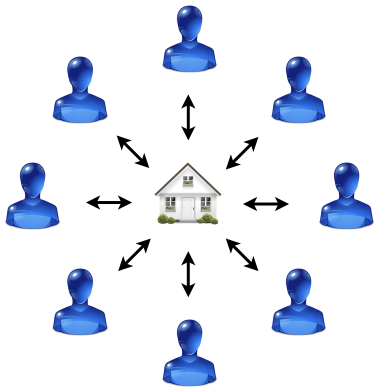
In order to restrict access to certain resources, a mechanism was needed to *authenticate* the users of a site. Due to a lack of better alternatives, the approach of personal computers was taken to the *Web* with catastrophic consequences: Each person had to choose a unique username and password at every platform for **identification** and **authentication** (3.2.).

Such a **platform-centric identity** cannot be transferred from one site to another, which also prevents the exchange of information (like trust) *across platforms*. This requirement gained in importance with the *Web 2.0* and its big diversity.



Web 2.0

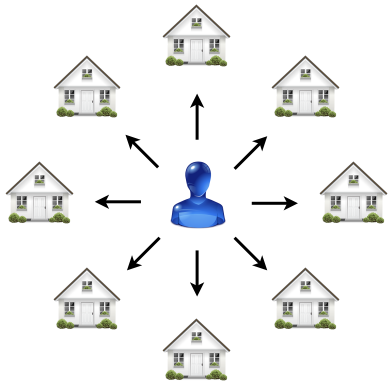
Around the turn of the millennium, a new trend emerged: *Platforms* were initiated to store and share **user-generated content**. People started to unite in virtual communities for various topics and purposes, where they no longer just all consumed information but also provided it. The term *Web 2.0* refers to a different use but not a change of technology.



Identity 2.0

An improvement to the above situation is the **federation** of various proprietary **identity management systems** so that users from *one* domain can access an application in *another* domain without reauthentication (so-called *single sign-on*).

True **user-centric identity**, however, is only achieved if the identity and service providers need not to be known before.



3.2. Terminology

User
The term *user* refers to all **human beings** interacting with a particular (computer) system. In identity management, they are usually the subjects about whom statements are made.

Client
A *client* (also called user agent or browser) is some **program** that runs on the user's device and accesses a remote service.

Server
A *server* is a machine that provides some **services** for clients and other servers. For this, they have to be always *reachable*.

Entity
An *entity* is a concrete or abstract **object** that exists by itself. This can be a natural or artificial person, a client, a server, etc.

Identity (ID)
Identity is the **transitive**, **symmetric and reflexive relation**, which makes two entities the same (that is to say *identical*). For the purpose of access control (see below), one needs to specify a procedure for the identification and authentication of entities. It is exactly this procedure that defines the notion of identity in the current system. However, this only imitates the identity of an entity in the real world and, as it is not an inherent part of it, identities can be *created*, *lost* and *stolen*.

Attribute
An *attribute* is a piece of **information** that is associated with an entity. Unless the author of an attribute has authoritative power, it is just a **claim** about the *subject* of the identity. So-called *ontologies* define the vocabulary used for attributes.

Certification
In order to increase the **trustworthiness** of attributes, their validity can be confirmed by special *certification authorities*. Which **organizations** are allowed to *issue* which certificates and whether these can be *revoked* depends on the standard.

Contact
Some standards do not only support the identification and authentication of its users, but also enable them to **model** their **social network** once for the reuse by various services.

Role
Most people have *different* roles in *different* **contexts**, which is facilitated by allowing them to assume *different* identities.

Access Control
Access control ensures the appropriate admission of people to resources according to some *access policy*. The procedure consists of the following steps: **Identification** of the subject, **authentication** of its claims and **authorization** for access.

The access policy can be based on several aspects of the users: their *identity*, their *attributes* and their *roles*, where the last is in practice just realized by one of the former two.

Furthermore, the rules of the system determine whether access control is *discretionary* (admission specified by the resource owner) or *mandatory* (enforced secrecy of data).

Identification
Whenever a resource is protected by some access policy, the requester needs to make claims about why he or she should be granted access. Corresponding to the above section, the user can either claim an *identity*, certain *attributes* or certain *roles*. An identity is usually declared with a *locally* or *globally* unique **identifier**. If no strict identification is required, some standards allow to disclose attributes *pseudonymously* or in a few cases even *anonymously* to the resource provider (the notion of *(un-)linkability* is explained in more detail in 4.1.).

Authentication
Whatever claims are raised, the requester needs to *convince* the resource provider of their *authenticity*. The verification of the presented **evidence** is called authentication and can be based on three **factors**: *property* (e.g. fingerprint, retinal pattern, signature), *possession* (key, badge, smart card) and *knowledge* (password, PIN, response to challenge). In order to increase security, multi-factor authentication is feasible.

Authorization
Once the user's identity, attributes or roles are established, it is determined whether he or she is indeed *authorized* for the intended action. The term comprises *having* the **privileges** as well as *granting* them. This thesis focuses mainly on the second aspect and distinguishes between permissions that are *authoritative* (act on behalf of the user) and those that are *non-authoritative* (allow others access to own assets).

Accreditation
The act of giving another entity *authoritative* **power** is called *accreditation*. This process results in the other entity gaining a *credential*, which can be used to act within a certain scope on behalf of the *issuer*. Certification authorities can accredit *intermediate authorities* and users can accredit their *clients*.

Accountability
Accountability means that an entity can be held **responsible** (and thus potentially also *liable*) for its actions and omissions.

Identity Provider (IP)
An *identity provider* **authenticates** an entity on behalf of the *service provider*. (The only alternative to delegated identity management is to establish a *web of trust* on the user level.)

Service Provider (SP)
A *service provider* (sometimes erroneously called *relying party*) **consumes** the verifications of *identity providers*.

4. Characterization

4.1. Identification

Identity

Comparing the wide variety of *identity standards* is not easy and needs to be based on a set of objectively determinable **characteristics**, which I describe in the following sections.

A first important consideration is the meaning of identity in a standard. Is the **real identity** of the user *ensured* or is it possible to create *arbitrary* identities? In the former case, is the virtual identity also *unique*? Other aspects of ID include:

- **Scope**: Is the identity valid in a *local*, *global* or *federated* context (where the latter is a mixture of the former two)?
- **Creation**: Which entity creates the identity for the user? The *user* by hand, an automated *client* or some *authority*?
- **Deletion**: Is the deletion of an identity *supported* and if so, what exactly does it mean? (No means just stop using it.)
- **Merging**: Is it *possible* to merge several identities into one, which is very useful for mergers and acquisitions but also for users who started with several identities and want to reduce them? Splitting an identity makes no sense, as it is unclear which successor shall assume its rights and duties.

Identifier

For the identification of an entity, it is essential to be able to reference its identity by means of an identifier. Besides the question what **form** it takes, the criteria below are of interest:

- **Relocation**: *Can* an identity be moved to a new provider (by changing its identifier) without affecting the system?
- **Reusability**: Is it explicitly *forbidden* to reuse an identifier for a different identity (by another user) after relocation?
- **Interaction**: How do users provide their identifiers during identification? Is the interaction with the service provider *implicit*, *explicit* (e.g. with a prompt) or *unspecified*? There is a tradeoff between usability and privacy through clear consent among the first two options. If the procedure for identification is not specified, it cannot be to automated.
- **Frequency**: How often is identification performed? Once per *browser session*, per *service provider* or just per *client*?
- **Submission**: *How* are identifiers finally submitted to SPs?

Attributes

For access control, it is often sufficient to receive just certain attributes of the requester without having to know his or her full identity. Another advantage is that services can easily be customized without any overhead. Given that this feature is supported, the question is whether the attributes to convey are **selected** *implicitly* or *explicitly* and by *whom*. More points:

- **Linkability**: Can several requests to a service provider be linked to the same entity? Attribute-based identification can either be *identifiable* (the identifier is disclosed to SP), *pseudonymous* (a distinct but persistent identifier created for each SP) or *anonymous* (even subsequent requests to the same SP cannot be associated with each other).
- **Revocation**: *Who* revokes anonymity in case of misuse?

4.2. Authentication

Mechanism

The claims made during the identification phase are verified by the service provider in the authentication stage. Usually, the **mechanism** to do so either involves *browser redirections* or *cryptographic signatures*. Further important aspects are:

- **Performance**: How much effort is required to validate the claims? This is best estimated by the number of *messages* sent between the parties and the amount of *cryptography* involved when computing and verifying the signatures.
- **Compatibility**: Are assertions in other standards equally accepted after a proper *conversion* into the current one?
- **Requirements**: *What* changes are needed to deploy it?
- **Vulnerabilities**: What *attacks* are the most likely (see A.4.)?
- **Privacy Issues**: Are there any privacy issues like the *calling home problem*, where the IP learns of all authentications?

Identity Provider

In authentication, the user's identity provider plays a crucial role. A major distinction is whether the identity provider has to be *online* during authentication (because it is part of the **communication flow**) or can also be *offline* (since it is never contacted). The risk of being temporarily locked out and denied access to a service is higher in the former case.

- **Discovery**: *How* can the user's IP be detected?

- **Relocation**: Can users change their identity provider and if yes, *how*? (Some standards allow to keep the identifier.)
- **Verification**: *How* is the authenticity of an IP examined?
- **Rotation**: How *often* do IPs have to change their keys?
- **Regulation**: Are there any *restrictions* to become an IP?

Service Provider

Does a service provider need to be **registered** *before* being able to consume the authentications of identity providers?

Evidence

In order to **authenticate** an entity, an identity provider asks for some *evidence*. Such evidence has several characteristics:

- **Factor**: Evidence is either based on *properties*, *possessions* or *knowledge* of the user. Each of them has pros and cons:
  - *Property*: Hard to lose and copy but difficult to replace.
  - *Possession*: Easy to replace but also easily lost and stolen.
  - *Knowledge*: Easy to forget and often easy to be guessed.
- **Control**: Who manages the evidence, the *client* or the *user*? If it is the user, it is not reasonable to authenticate a client in absence of the user, which prevents any automation.
- **Rotation**: Does the evidence need to be *changed* at times?
- **Interaction**: *How* do users provide the required evidence?
- **Frequency**: How often is the evidence to required? Once per *browser session*, per *service provider* or per *ID provider*?
- **Submission**: *How* is the evidence submitted to the SP?
- **Compromise**: If an attacker obtains the evidence, is only the concerned *client* or the complete *identity* at risk? This question usually depends on the authorization (see 4.3.).
- **Revocation**: When lost, *how* is the evidence invalidated?

Session

Once the authentication was successful, *how* is it **retained**?

- **Logout**: *How* do you logout from the numerous services? Single sign-on does not necessarily mean single log-out!
- **Encryption**: Is encrypting the communication *mandatory*?
- **Forward Secrecy**: Is the communication still *confidential* if an attacker learns one of the private keys in the future?

4.3. Authorization

Clients

Is it *possible* to give clients **authoritative power** over an ID?

- **Rights**: If yes, *what* is the client authorized to do at the IP?
- **Permissions**: With which *privileges* can the client also act on behalf of the user towards others? (The distinction to rights is that permissions also depend on others; see 4.7.)
- **Restrictions**: Can the in- and external privileges be *limited*?
- **Management**: *Can* a client add and remove other clients?
- **Accreditation**: *How* can a client request to be authorized?
- **Deaccreditation**: *How* is a client's authorization removed?

Users

Is it also *possible* to **authorize** other users respectively their identities with similar rights and permissions as for clients?

4.4. Accountability

Signatures

Can users respectively their clients **sign** a message so that its origin cannot be denied (which is called non-repudiation)? If yes, *where* are such signatures used in the protocol (if at all)?

Liability

What is the **legal status** of these *signatures*? Furthermore:

- **Duration**: For what *time* can the issuer be held liable?
- **Revocation**: Is it *possible* to invalidate a client's key?

Framing

Is it *possible* for an identity provider to **assume** the identity of its users? Such a misbehavior can be prevented or at least detected if the identity provider is required to keep a record of each modification to an identity. Whenever some change cannot be traced back to the original client by verifying and comparing the signatures, we can be sure that the identity provider was cheating. (Depending on the desired scenario, this property might seem less relevant than it actually is.)

4.5. Attributes

Storage

Though a lot of standards allow the association of attributes to an identity, the possibilities vary widely. Besides the major question, whether the **set** of attributes is *predefined* or freely *extensible* by everyone, the following aspects are of interest:

- **Format**: Are attributes restricted to *strings* or can *arbitrary* values be used? How are the *syntax* and *semantics* defined?
- **Regulation**: *Who* specifies the rules and policies regarding the usage of attributes by third parties (if specified at all)?
- **Transmission**: Can attributes be conveyed to a SP during *authentication* for ease or attribute-based access control?

Access

Can service providers or third parties retrieve attributes of an identity **asynchronously** (independently of authentication)? If so, can the access rules be *specified* on the attribute-level?

- **Policy**: *How* are these rules defined; *what* are the options?
- **Caching**: How long may attributes be stored before they are considered *stale* (expired) and need to be *refetched*?

Certification

Can attributes be **certified** in order to increase the reliability? (Trust is often implicitly given in federated identity systems.)

- **Authority**: Which entities are the certification authorities? Only the respective *IP*; *arbitrary* or *hierarchical* authorities?
- **Revocation**: Is it *feasible* to invalidate issued certificates?
- **Liability**: Are authorities *liable* for incorrect certificates?

4.6. Contacts

Format

Can you add other identities as contacts to your identity so that service providers can make use of your **social graph**? If this feature is supported, *how* are these contacts stored?

Structure

Is there a way to **group** the contacts for easier *management*?

Disclosure

Is it *possible* to give third parties like friends and companies also **access** to your contacts? If so, *how* is this accomplished?

Notification

Though the concept of contacts is usually asymmetric (not mutual), *can* you **inform** other users about your existence?

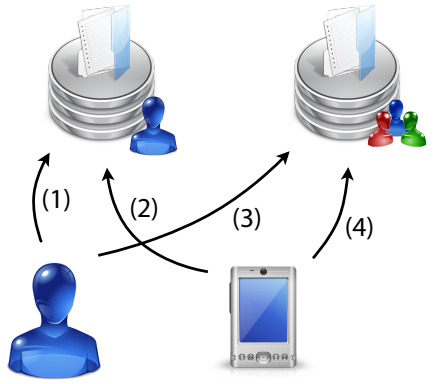
4.7. Classification

Crucial Characteristics

When looking at the characteristics outlined above, it can be seen that one of the major distinctions is whether a standard is designed to **interact** with the **user** or just with their **client**. This manifests itself in the kind of *evidence* that is requested for authentication as well as in the *automation* of procedures.

A key feature of most standards is the ability to prove one's identity towards other entities, which requires a *global* (or at least *federated*) *identifier*. In order to be considered the same identity, there has to be an identity provider so that a single change (like new *evidence* or a modified *attribute*) affects all service providers. In other words, there is also a difference whether **access** can only be given to **internal** resources (at the own provider) or also **external** ones (at other providers).

I suggest the following *terminology* for this **classification**:



Standard	User	Client
Internal	(1) Authentication	(2) Authorization
External	(3) Federation or Sign-On	(4) Identity

(1) Authentication Standard

The main purpose of an *authentication standard* is to **unify** the user experience and **simplify** the development of apps. Typically, the user has a *local identifier* and authentication is straightforward. Example: *HTTP Authentication* [RFC2617].

(2) Authorization Standard

The goal of *authorization standards* is to fight the so-called **password anti-pattern**. (An anti-pattern is a procedure that was originally considered a good idea and became popular but later turned out to be counterproductive in practice.)

The most common use case is to give applications **access** to a **service** without entering the master password, which is more secure as it limits the possible damage when the *client* gets compromised. Example: *OAuth* [OAuth] (to protect *APIs*).

(3a) Federation Standard

A *federation standard* allows to use an identity throughout a federation. You need to be accepted by the federation to run your *own* identity or service provider though, which is often not a problem among **corporate** and **academic** institutions. Example: The *Security Assertion Markup Language* [SAML].

(3b) Sign-On Standard

A *sign-on standard* provides for each user a **global identifier** that can be used to prove one's identity towards any service provider through authentication. Example: *OpenID* [OpenID].

(4) Identity Standard

In order to qualify as an *identity standard*, **global identifiers** and full **automation** with *attributes* and *contacts* are needed. *Client accountability* is also desired. Example: *WebID* [WebID].

5. Comparative Study

5.1. Standards

Selection

After having seen numerous *characteristics* of **access control standards**, it is time to examine their *consequences* based on representatives from each class in practice. The aim is to get an idea of their respective *strengths* and *weaknesses*, where only **non-commercial** standards were considered.

Another requirement for standards was to have a publicly available specification of their underlying **protocol**, thereby intentionally excluding mere *technologies* and *initiatives*. As pure **authentication** and **authorization standards** were of no interest for the *scenario* described in 5.2., those were also discarded. The following standards have then been chosen for a mixture of *popularity* and *diversity* (see appendix B. for an in-depth analysis based on the above *characterization*):

- **SAML**: The *Security Assertion Markup Language* is an XML-based data format for the exchange of authentication and authorization information between identity providers and service providers. It has been devised by the *Organization for the Advancement of Structured Information Standards* (*OASIS*) in 2001 with its second version being released in 2005. *SAML* has gained wide-spread deployment among governmental, academic and commercial organizations. As *ETH Zurich* bases the authentication of its users also on it, *SAML* is the perfect example of a *federation standard*.
- **OpenID**: The most well-known standard for *authenticating* users on the *Web* is *OpenID*, which was developed in 2005. After having gained tremendous momentum in 2008, its popularity is steadily declining ever since. Many service providers stopped their *OpenID* support due to its lack of usability as people *forget* and *mix* their *OpenIDs* and can no longer *recover* their accounts. (I discuss the difficulties with *Google* as the largest *OpenID* provider in 5.5.) Though many service providers are turning away from it, *OpenID* is still the *sign-on standard* of preference for this study.
- **WebID**: *WebID* is the first real *identity standard* according to the above classification and uses *SSL client certificates* to authenticate its users. Each of them has a profile page in the machine-readable *Friend Of A Friend* (*FOAF*) syntax, where the public key of their certificates are published. Originally proposed at a *W3C Workshop* in 2009, it is now elaborated in a *W3C Community Group* but its adoption is quite limited so far. Its approach is still interesting though!
- **Virtual ID**: After having developed the main concepts of *Virtual ID* in 2010, I worked out its cryptographic protocol in my bachelor thesis. During the *Distributed Systems Lab* at *ETH Zurich*, I have built a first prototype together with *Dominik Blunschy*. *Virtual ID* belongs also to the class of *identity standards* and is this thesis's actual motivation.

Rejection

Due to limited time, many standards that would have been interesting to study had to be omitted. The following list of such standards indicates some reasons for their exclusion:

- **Kerberos**: Developed in 1983 as an authentication solution for the *Massachusetts Institute of Technology* (*MIT*), *Kerberos* became an *IETF standard* ten years later. It uses *symmetric key cryptography* to distribute keys to users and services for *mutual authentication* and *confidential communication*. As a trusted third party is required, it does not scale to the size of the Internet and is just another *federation standard*.
- **OpenPGP**: *Pretty Good Privacy* (*PGP*) had been created by Phil Zimmermann in 1991 and standardized as *OpenPGP* by the *Internet Engineering Task Force* (*IETF*), see [RFC4880]. It is mainly used to encrypt and sign e-mails among peers without relying on central authorities for key certificates (thanks to its *web of trust*). To my knowledge, *OpenPGP* has never been used to authenticate users at services so far.
- **WS-Federation**: This standard belongs to the *Web Services* protocol suite and enables the *federation* of identities very similar to *SAML*. Its main proponents are *Microsoft* and *IBM*, which use it as part of their *Identity Metasystem* (see 2.4.) in the *Higgins Project* or by the notion of *Information Cards*.
- **BrowserID**: This protocol (also known as *Mozilla Persona*) was initiated in July 2011 and is from a user's perspective quite close to *OpenID*. The main *differences* are that it uses email addresses instead of URLs as *identifiers*, improves on *privacy* by solving the *calling home problem* and is meant to be fully *integrated* into the next generation of browsers.
- **SuisseID**: This is the first digital identity whose signatures are legally approved in Switzerland and this initiative was launched by the *Swiss government* in May 2010. Users get a certificate issued by an accredited private organization on a smart card or USB token to authenticate themselves and sign. It is, therefore, only an *authentication standard*.

5.2. Scenario

File Sharing

As we have seen in the classification of standards (see 4.7.), identity management is all about **access control** to *internal* and *external* resources. A very basic but quite generic form of resources are files. As the sharing of files is a common use case of the *Global Information Systems Group* at *ETH Zurich*, I have also adopted it as the scenario for my study. The goal is to implement a **distributed file service** based on the chosen standards in order to provide a decentralized alternative to platforms like *Dropbox*, *Google Drive* and *Microsoft's SkyDrive*.

Requirements

The beauty of this scenario is that only a few characteristics (see 4.) are essential but many aspects of the standards like attributes and contacts affect the criteria described below.

The minimal set of **features** that need to be supported are:

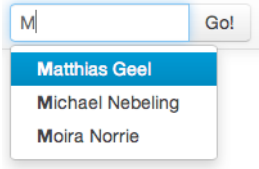
- **Internal Files**: Users have to be able to *upload*, *download* and *delete* their own files from the file service provider of their choice. Additionally, they may specify for each file an access policy that determines its *visibility* towards others.
- **Contacts**: For a simplified access to the files of their friends, users *add* and *remove* contacts that they can click to *visit*.
- **External Files**: When visiting the files of a friend, users can *see* and *download* the ones they are indeed authorized for.

User Interface

For the first three standards *SAML*, *OpenID* and *WebID*, I have implemented a single **Web application** and protected it with the various authentication mechanisms. Each requirement is covered by a dedicated page, which are printed below.

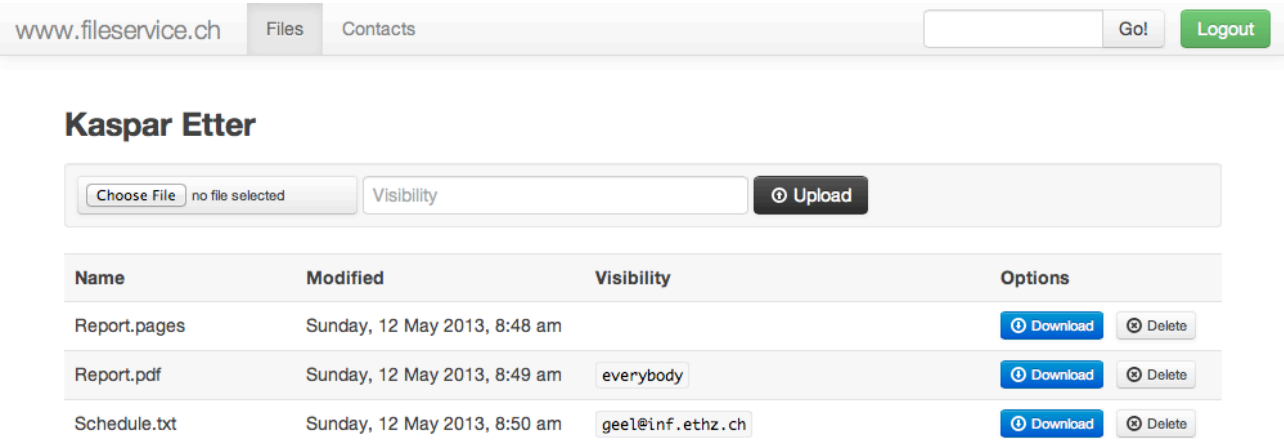
The prototypes are **available** on *www.fileservice.ch* and on the separate host *www.xdf.ch*. (*XDF* is Virtual ID's data format and the domain is just reused for this new purpose.) **Please do not rely on this file service!** It is only a proof of concept and the database may be reset at any time! For this reason, no support will be provided to the users of this platform.

The intention of this *implementation* is to demonstrate the **advantages** and **disadvantages** of different approaches to access control. In particular, **usability and security were of no concern** and could easily be improved! For example, all users share the same table for their identities and, therefore, hijacking another name is trivial, especially when searching:

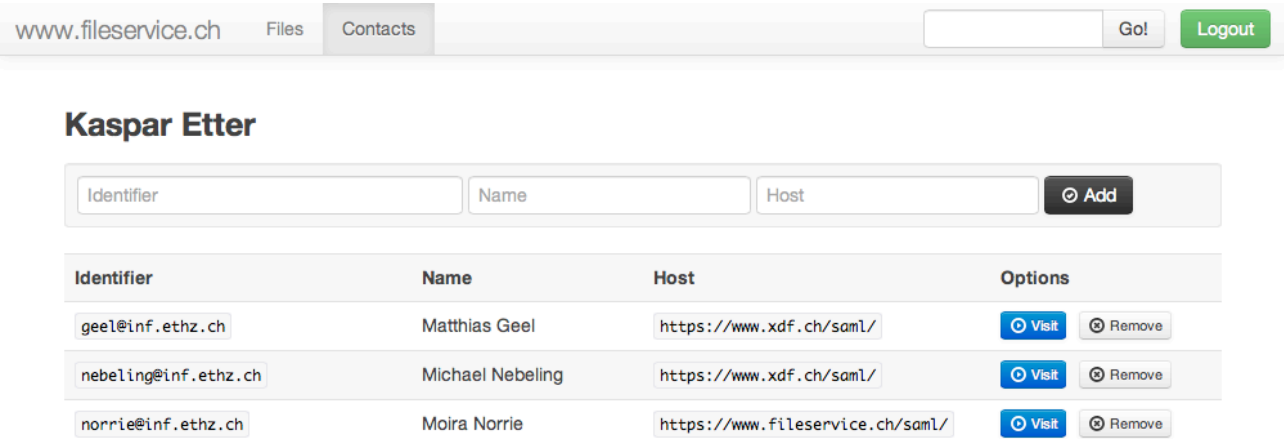


Search box for contacts.

List of my **files** with the options to *upload*, *download* or *delete* a file:



List of my **contacts** with the options to *add*, *visit* or *remove* a contact:



List of my **files** as seen by *Matthias Geel* with the option to *download* them:

www.fileservice.ch Files Return

Kaspar Etter

Name	Modified	Options
Report.pdf	Sunday, 12 May 2013, 8:49 am	Download
Schedule.txt	Sunday, 12 May 2013, 8:50 am	Download

5.3. Criteria

Categories

For being able to compare the chosen *standards* based on the presented *scenario* (see 5.1. and 5.2.), we need a set of objective *criteria*. From the numerous **characteristics** that were discussed earlier (see 4.), I have selected those which are really *relevant* for decentralized file sharing and have grouped them into the following four categories:

- **Suitability**: Is the standard appropriate for the scenario?
- **Usability**: How well does the solution support the user?
- **Security**: How vulnerable is the application to attacks?
- **Privacy**: Is the dissemination of information restricted?

Focus

Most of the criteria described below concentrate on aspects of the *standard* and not their respective *frameworks* or usual *practices*. As an example, the ease of deployment is not part of the study as it is difficult to assess **objectively** and tells us more about the available tools than the inherent concepts. If a standard is not so clear about the right *practice*, however, it is the *mainstream* that counts (also in case of divergence).

Grading

Each category consists of **four aspects** on which a standard can score. Points are either awarded *completely* or *not at all*, with four points being the *maximum* and zero the *minimum*. The following **ranking** is used: –– (0), – (1), = (2), + (3), ++ (4).

Exclusion

Covering only sixteen *characteristics* of identity management systems means that many features were not considered. This includes most facets of identity administration, performance, attribute certification, caching, liability and regulation, which were not regarded as being crucial for the outlined scenario.

Suitability

- **Scope of Identity**: If everyone should be able to host their files themselves, it is essential that identities are accepted *globally* without requiring a central registration in advance.
- **Attribute Access**: In order to *look up* the name and host of other users (see the screenshot on the previous page), it is desirable to be able to retrieve attributes asynchronously. (Attribute transmission during authentication is deficient.)
- **Authorization**: Especially when dealing with files, you are interested to give native clients access to your resources but also to the ones of your friends. If the procedures can be *automated*, such a client is able to import external files.
- **Accountability**: As soon as you start to delegate tasks (be it to clients or other users), being capable of holding each entity *responsible* is beneficial (e.g. when modifying files).

Usability

- **Requirements**: Does the usage of a standard demand any additional *hardware* or *software* (being bad for adoption)?
- **Attribute-Based Access Control (ABAC)**: Many times, it is more convenient to restrict the admission to resources on the basis of trusted attributes instead of listing identifiers.
- **Contacts**: Automatically synchronizing his or her contacts with a new service improves the user's efficiency notably.
- **Authentication**: If each contact uses a different host, the authentication at every service provider is cumbersome. Either trusting your *contacts* or all *SPs* does simplify a lot!

Security

- **Phishing**: Whenever a browser *redirection* is initiated by a service provider, users could be sent to a fake ID provider.
- **Evidence**: Since most users choose too weak passwords, a point is only granted if authentication relies on *possession*.
- **Compromise**: In case some evidence gets compromised, it is preferable to lose only control of a *client*, not your *ID*!
- **Logout**: After having downloaded all the sought files, how can you know that you logged out at *all* service providers?

Privacy

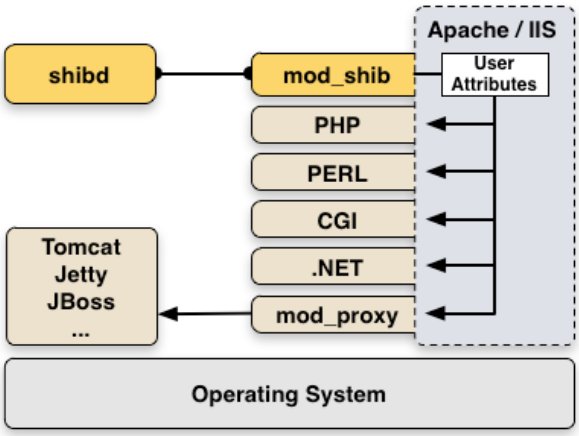
- **Calling Home**: If an identity provider learns about every authentication, it can *track* its users across the Internet.
- **Wrong Consent**: Similar to *phishing*, not only the identity provider can be forged, but also a service provider. Such a situation might lead to an involuntary leakage of identity.
- **Attribute Selection**: Though a lot of standards allow the transmission of attributes during *authentication*, it is not always up to the user to select respectively reject them. (In distributed services, users no longer choose all SPs!)
- **Pseudonymity**: Allowing users to *mask* their real identity for attribute-based access control improves their *privacy*.

5.4. SAML

Execution

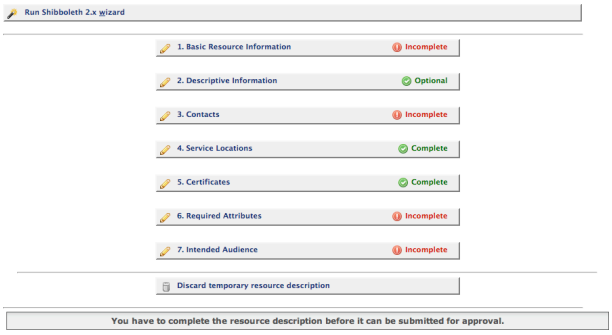
In order to reuse the credentials of users at *ETH Zurich* (or any other Swiss university), I have chosen the *Authentication and Authorization Infrastructure* of *SWITCH* (short **SWITCHaai**) for the implementation of the first standard. *SWITCHaai* runs on the open source software *Shibboleth*, which uses the *Security Assertion Markup Language* to provide single sign-on among federation partners. (See *www.switch.ch/aai/demo/* for how *Shibboleth* works and appendix C. for all the references).

On *www.switch.ch/aai/support/serviceproviders/*, *SWITCH* has excellent **guides** for installing and configuring *Shibboleth* in the *SWITCHaai* federation. Having the file service realized with *Java Server Pages* running on *Tomcat* (see 5.2.), I needed to switch to *Apache* for being able to install the *Shibboleth* module (called *mod_shib*) that communicates with a special process named *shibd*. Only three directives were required to **protect a location** with *Shibboleth* in the configuration files of *Apache*. The requested attributes of users (see below) are passed through the *Apache JServ Protocol* (*AJP*) to the login script, which establishes a cookie-based user **session**. (This step is not necessary as *Shibboleth* would handle the session for you, but I kept it in conformity with the other standards.)



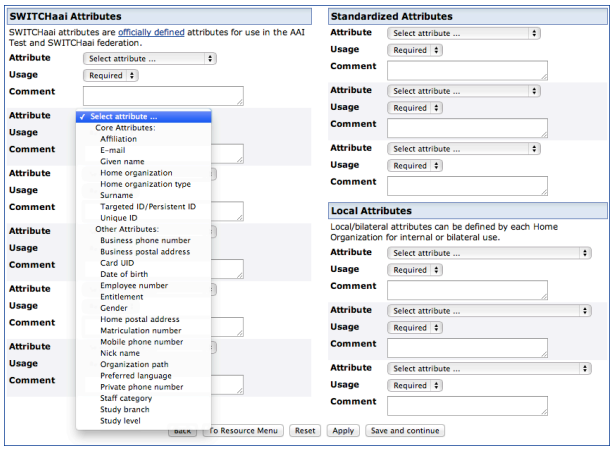
The **server setup** (chart taken from *www.switch.ch*).

After installation and configuration, you need to register as a service provider at the *AAI Resource Registry* (*rr.aai.switch.ch*):



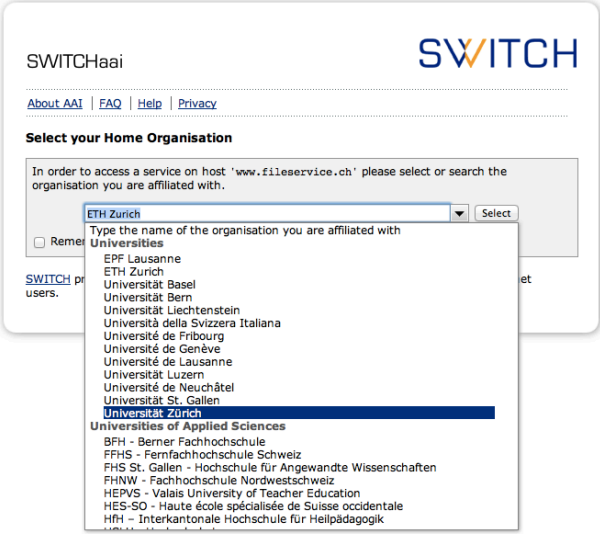
Completing your data at the **resource registry**.

All technical information is directly taken from the *metadata*, you still have to enter many details manually though. Before you are done, you also specify the *attributes* that you require from the identity providers as well as the *intended audience*:



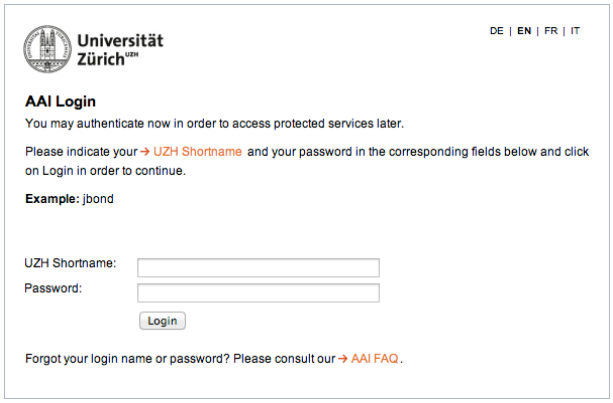
Selecting the *required* or *desired* **attributes**.

Once submitted, your application has to be approved by the Resource Registration Authority of your organization, which was in my case the *IT Services* of *ETH Zurich*. As a last step, you confirm the fingerprint of the **self-issued certificate** used by *Shibboleth* to encrypt and authenticate *SAML* messages on a second channel like e-mail. Afterwards, users can select their home organization and authenticate themselves to **log in**:



Redirection to the so-called *where-are-you-from* page.

User **authentication** at and by *ETH Zurich*.



SWITCHaai **authentication** at the *University of Zurich*.

Overall, the deployment process was relatively smooth as a lot of things happen under the hood. However, if the default configuration is no longer sufficient because, e.g., you want to activate a second domain (as in my case with *www.xdf.ch*), the hassle begins and *Shibboleth* becomes overly complex.

Suitability

- **Scope of Identity**: Considering the on both sides largely manual registration procedure for service providers, self-hosting is not practical for most users in this federation.
- **Attribute Access**: Shibboleth only allows the transmission of attributes during authentication. If you want to add an external user as a contact, no data lookups can be made.
- **Authorization**: *SAML* does not know the notion of a client as presented here and thus no authorizations are possible. Since the authentication towards the identity provider is not further specified, file access cannot be automated.
- **Accountability**: Users are not able to sign their requests.

Usability

- **Requirements**: *Shibboleth* uses standard technologies of the *Web* and therefore functions in all available browsers. It relies on cookies to store the user session and, without *JavaScript*, one more click is needed after authentication.
- **Attribute-Based Access Control**: The big advantage of an identity federation is that you know all identity providers. As they guarantee the quality of its users' attributes, you can rely on these for access control. The only problem is that you need to know the desired attributes in advance, as they are already specified in the registration procedure.
- **Contacts**: The only information you can get about a user's social environment is his or her organizational unit. When combined with the *Lightweight Directory Access Protocol* to get useful contact data (separately for each organization), the search functionality can still significantly be improved.
- **Authentication**: Since the service providers of a federation are manually approved, all of them are trusted to respect the privacy of users, who are thus implicitly signed on.

Security

- **Phishing**: The fact that unsuspecting users can be sent to an arbitrary page for selecting their organization and then another one for authenticating themselves is a very severe issue of *Shibboleth* and knowing all IPs does not solve this!
- **Evidence**: How users authenticate themselves at their IPs is not prescribed by the standard but, as you can certainly guess from the screenshots, most IPs employ passwords.
- **Compromise**: Since such passwords usually allow you to control the whole identity, which in case of *SWITCHaai* is often a very critical one, attacks can be totally disastrous.
- **Logout**: What probably most users are unaware of is the circumstance that they will remain logged in for a specific service provider until they *quit* their browser. Just *logging out* at a service provider and *closing* the open windows is not sufficient (though for example *myStudies* tells you so)!

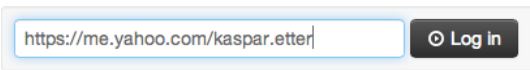
Privacy

- **Calling Home**: Your home organization learns about the time and place of every authentication that you demand.
- **Wrong Consent**: If users tick the checkbox so that *SWITCH* remembers their selection of home organization, they will no longer be notified about future authentication requests during the same browser session! This means that as soon as you are approved as a service provider, you can identify visitors on your site without their knowledge or consent.
- **Attribute Selection**: Another drawback of *SWITCHaai* is the fact that users are never informed about which attributes are transmitted to a service provider (see the screenshots above)! (A list of the requested attributes by each service provider is available on *www.switch.ch/aai/participants/*.)
- **Pseudonymity**: There is a dedicated attribute to identify users pseudonymously (in order to confuse developers it is both called *targeted ID* and *persistent ID*) but as they are unlikely to learn about it, such privacy is totally voluntary.

5.5. OpenID

Execution

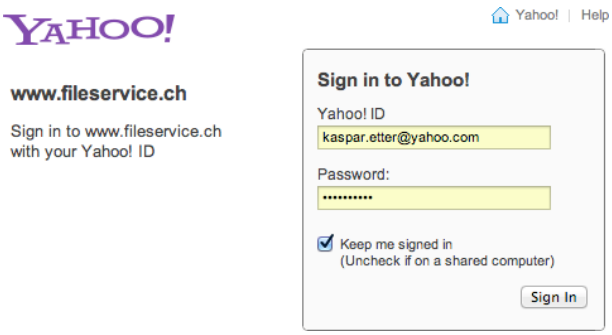
Deploying *OpenID* for user authentication is fairly simple: Import the libraries from *code.google.com/p/openid4java/* and add a few lines to your code. Users can then log in:



Identification at the service provider.

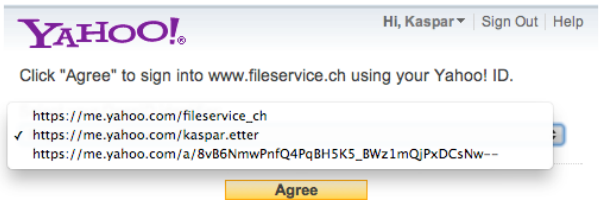
After entering their *OpenID* identifier, which is usually just an *URL*, users need to **prove** that they indeed **control** the given identifier. For this purpose, a service provider (called *Relying Party* in the specification) performs a so-called **discovery** on the identifier, looking for metadata in the header of the page referenced by the *URL* which indicates their *OpenID Provider*. Users are then redirected to their *OpenID Provider* with a few request parameters encoded into the query string (which is

the part after the question mark in a *URL*), where they enter their credentials to authenticate themselves the first time:



Authentication at the identity provider.

In a final step, users need to **approve** that they indeed want to disclose their identity and possibly some more attributes to the stated service provider. Certain *OpenID Provider* allow their users to choose between various identifiers and related attributes (a feature which is often called *personas*) to return to service providers. The specification calls this functionality *OpenID Provider driven identifier selection* (section 10 of *spec*):



Authorization of the SP at the IP with ID selection.

In order to protect their users' privacy, *Google* goes even one step further and generates automatically a *distinct* identifier for *every* service provider. This concept is known as **directed identity** (see 2.4.) and can to my knowledge not be turned off.

Suitability

- **Scope of Identity**: *OpenIDs* are accepted *globally* but, due to the above-mentioned feature, users are often not aware of their true but *opaque* (non human-readable) identifier.
- **Attribute Access**: An extension to the core protocol which is called *OpenID Attribute Exchange* allows service providers to *read* or *store* personal information *during* authentication. However, this does not enable service providers which are not involved in the authentication to look up attributes of contacts. As the big players *Yahoo!* and *Google* do have no public profile for their users, it is also hard to use the *URL*.
- **Authorization**: *OpenID* is a standard to authenticate users that interact with a browser in the *World Wide Web*. Since giving clients or third parties access to one's resources is not incorporated, a complementary framework evolved under the name *OAuth*, providing a standardized method to authorize clients for *internal* (but not *external*) access.
- **Accountability**: Though *OpenID* has been designed with the aim of allowing people to comment on blog posts by others, it provides no way to check a statement's validity.

Usability

- **Requirements**: None, and you do not even need to have root access to a *Web* server to run your own SP or also IP!
- **Attribute-Based Access Control (ABAC)**: *OpenID* states explicitly that it is not a trust system and that you cannot rely on identity providers in order to prevent spamming.
- **Contacts**: *OpenID* does not support exchanging contacts.
- **Authentication**: For limiting the risk of accidental identity disclosure, service providers have to be approved initially.

Security

- **Phishing**: User can be spoofed easily by redirecting them to a fake copy of their identity provider or performing so-called cross-site scripting if no appropriate measures are in place. The most reasonable way to prevent this is to use client certificates for the authentication of users at identity providers, but in that case you could as well rely on *WebID*.
- **Evidence**: It is the responsibility of every identity provider to choose an adequate authentication mechanism, but as most of them use passwords, no point can be given here.
- **Compromise**: Your *OpenID* is your single point of failure. If your credentials get stolen, the attacker can access all SPs.
- **Logout**: *OpenID* is just about authenticating users towards service providers and supports no session management. As a consequence, there is no *single log-out* functionality.

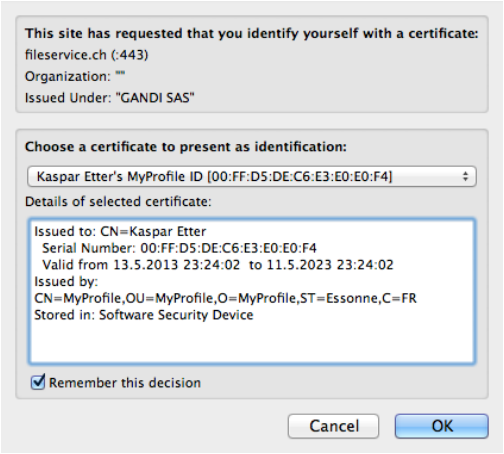
Privacy

- **Calling Home**: Being redirected to your identity provider for each authentication means that it can totally track you.
- **Wrong Consent**: Since you need to approve each service provider the first time you sign on, you should recognize an unsolicited attempt to establish your identity. For being notified every time you log in at a certain service provider, you can usually just uncheck a corresponding checkbox.
- **Attribute Selection**: Though the *OpenID Attribute Exchange* specification explicitly states that requirements should not be enforced by the *OpenID Provider*, a lot of providers fail to provide a reasonable user experience for deselecting a few attributes. This is not a problem of *OpenID*, however.
- **Pseudonymity**: By allowing the issuance of *random* but *persistent* identifiers per service provider (also known as *directed identities*), pseudonymity can be achieved but the original purpose of *OpenID* – to *identify* users – is defeated.

5.6. WebID

Execution

The beauty of *WebID* is its simplicity: Users create a certificate in their browser for client-side **authentication** in the secured *Web* protocol (known as *HTTPS*) and publish their public key on their **profile page**, which can be hosted anywhere. When a server demands a certificate, users are asked to select one:



Selecting the desired **client certificate**.

These certificates do not have to be signed by a well-known certification authority, the software on the server just needs to **verify** that the public key is indeed published on the page stated as a *Subject Alternative Name* (see image on the right).

The only difficulty in deployment was to configure a server in such a way that users were indeed prompted to provide a client certificate. With the little information that is available on the *Web* about *WebID*, we did not manage to **set up** our *Tomcat* server correctly. In case of *Apache*, however, a single directive ("SSLVerifyClient optional_no_ca") did the trick.

Suitability

- **Scope of Identity**: Based on *URLs*, identifiers are global.
- **Attribute Access**: Using vocabularies such as *FOAF* (*Friend of a Friend*), users can add personal details to their profile. As long as these are public, everyone can retrieve them.
- **Authorization**: The certificates can also be used by other clients than browsers (and even for other protocols than *HTTPS*), allowing the automated import of external files.
- **Accountability**: *Transport Layer Security* (*TLS*), on which *WebID* builds, provides *authenticity* and *confidentiality* but no *non-repudiation* among the communication partners.

Usability

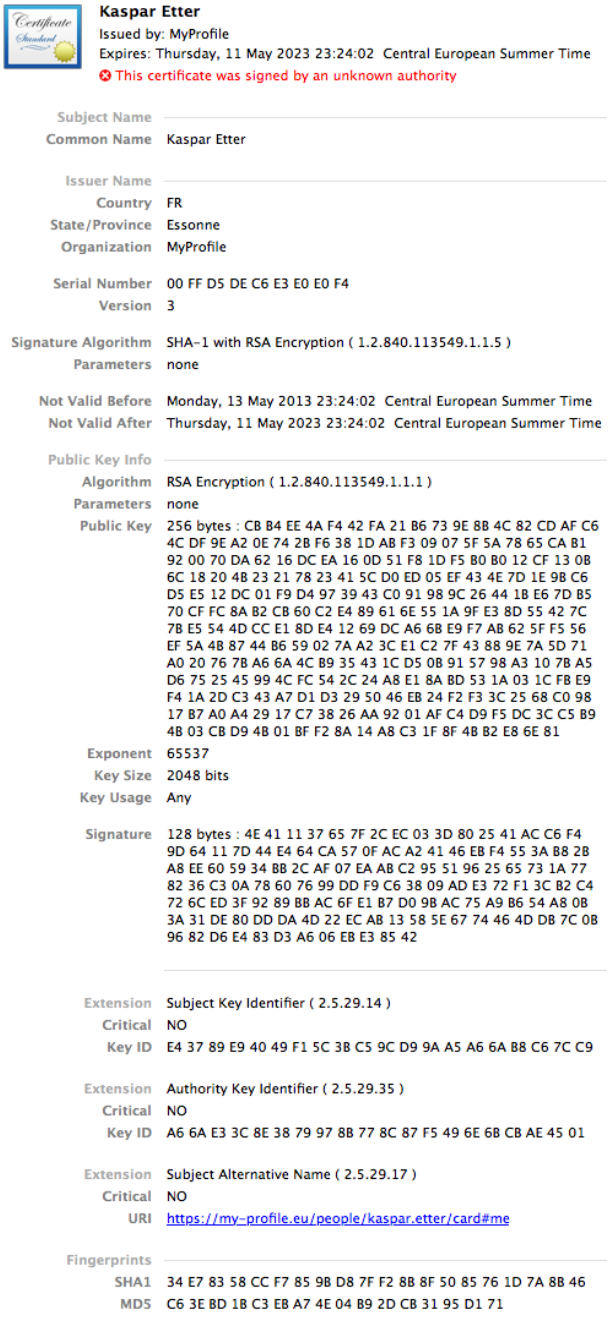
- **Requirements**: *WebID* builds on common *Web* standards and works in most browsers but its usability varies widely.
- **Attribute-Based Access Control**: The various statements on a user's profile can generally not be trusted and are not intended to be certified, but thanks to the *Semantic Web* (respectively the *Resource Description Framework* (*RDF*)), claims can be verified by consulting the primary source.
- **Contacts**: As the name *FOAF* suggests, its goal is to link people on the *Web* and SPs can easily consume this data.
- **Authentication**: Depending on your browser, you can set it to present a client certificate without previously asking you, but this is usually not what you want. Consequently, you need to select a certificate for every service provider.

Security

- **Phishing**: As no redirections are used and your private key is never revealed (not even to your IP), this is not an issue.
- **Evidence**: Since the corresponding private key is required to misuse certificates, your ID is protected with *possession*.
- **Compromise**: *WebID* does not specify any restrictions of authorization, so unless your identity provider does anything special, you risk to lose control over your entire ID!
- **Logout**: As soon as browser manufacturers start to offer better support for *WebID*, a single logout is no problem.

Privacy

- **Calling Home**: Since service providers could fetch profile data anonymously, identity providers cannot track users.
- **Wrong Consent**: With current browser implementations, it could easily happen that you approve identification to undesired SPs, since these are not stored across sessions.
- **Attribute Selection**: Though *WebID* does not support the transmission of attributes during authentication, a user can still configure its IP to deliver them only selectively.
- **Pseudonymity**: Since the identifier in a certificate cannot be modified, the user is always identified (and creating a separate certificate per SP renders *WebID* totally useless).



Kaspar Etter
Issued by: MyProfile
Expires: Thursday, 11 May 2023 23:24:02 Central European Summer Time
This certificate was signed by an unknown authority

Subject Name	
Common Name	Kaspar Etter
Issuer Name	
Country	FR
State/Province	Essonne
Organization	MyProfile
Serial Number	00 FF D5 DE C6 E3 E0 E0 F4
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Monday, 13 May 2013 23:24:02 Central European Summer Time
Not Valid After	Thursday, 11 May 2023 23:24:02 Central European Summer Time
Public Key Info	
Algorithm	RSA Encryption (1.2.840.113549.1.1.1)
Parameters	none
Public Key	256 bytes : CB B4 EE 4A F4 42 FA 21 B6 73 9E 8B 4C 82 CD AF C6 4C DF 9E A2 0E 74 2B F6 38 1D AB F3 09 07 5F 5A 78 65 CA B1 92 00 70 DA 62 16 DC EA 16 0D 51 F8 1D F5 B0 B0 12 CF 13 0B 6C 18 20 4B 23 21 78 23 41 5C D0 ED 05 EF 43 4E 7D 1E 9B C6 D5 E5 12 DC 01 F9 D4 97 39 43 C0 91 98 9C 26 44 1B E6 7D B5 70 CF FC 8A B2 CB 60 C2 E4 89 61 6E 55 1A 9F E3 8D 55 42 7C 7B E5 54 4D CC E1 8D E4 12 69 DC A6 6B E9 F7 AB 62 5F F5 56 EF 5A 4B 87 44 B6 59 02 7A A2 3C E1 C2 7F 43 88 9E 7A 5D 71 A0 20 76 7B A6 6A 4C B9 35 43 1C D5 0B 91 57 98 A3 10 7B A5 D6 75 25 45 99 4C FC 54 2C 24 A8 E1 8A BD 53 1A 03 1C FB E9 F4 1A 2D C3 43 A7 D1 D3 29 50 46 EB 24 F2 F3 3C 25 68 C0 98 17 B7 A0 A4 29 17 C7 38 26 AA 92 01 AF C4 D9 F5 DC 3C C5 B9 4B 03 CB D9 4B 01 BF F2 8A 14 A8 C3 1F 8F 4B B2 E8 6E 81
Exponent	65537
Key Size	2048 bits
Key Usage	Any
Signature	128 bytes : 4E 41 11 37 65 7F 2C EC 03 3D 80 25 41 AC C6 F4 9D 64 11 7D 44 E4 64 CA 57 0F AC A2 41 46 EB F4 55 3A B8 2B A8 EE 60 59 34 BB 2C AF 07 EA AB C2 95 51 96 25 65 73 1A 77 82 36 C3 0A 78 60 76 99 DD F9 C6 38 09 AD E3 72 F1 3C B2 C4 72 6C ED 3F 92 89 BB AC 6F E1 B7 D0 9B AC 75 A9 B6 54 A8 0B 3A 31 DE 80 DD DA 4D 22 EC AB 13 58 5E 67 74 46 4D DB 7C 0B 96 82 D6 E4 83 D3 A6 06 EB E3 85 42
Extension	Subject Key Identifier (2.5.29.14)
Critical	NO
Key ID	E4 37 89 E9 40 49 F1 5C 3B C5 9C D9 9A A5 A6 6A B8 C6 7C C9
Extension	Authority Key Identifier (2.5.29.35)
Critical	NO
Key ID	A6 6A E3 3C 8E 38 79 97 8B 77 8C 87 F5 49 6E 6B CB AE 45 01
Extension	Subject Alternative Name (2.5.29.17)
Critical	NO
URI	https://my-profile.eu/people/kaspar.etter/card#me
Fingerprints	
SHA1	34 E7 83 58 CC F7 85 9B D8 7F F2 8B 8F 50 85 76 1D 7A 8B 46
MD5	C6 3E BD 1B C3 EB A7 4E 04 B9 2D CB 31 95 D1 71

Certificate with the **identifier** in *Subject Alternative Name*.

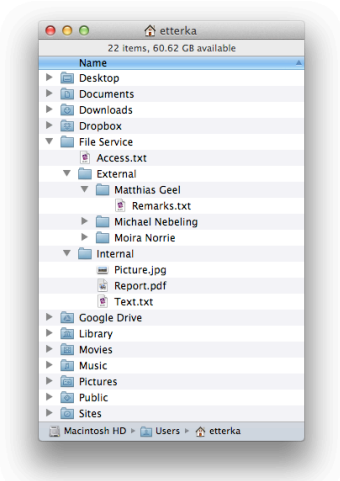
5.7. Virtual ID

Execution

A big part of this master thesis consisted in bringing the not fully existing implementation of *Virtual ID* to a level that can be used for a **decentralized file service** as described above. The current version builds on a prototype that I have written together with *Dominik Blunschy* in the *Distributed Systems Lab* at *ETH Zurich* last spring. Besides general **improvements**, the main advancements have been in managing clients with the granular permissions and restrictions as outlined in a thesis for my bachelor's degree (client authorization has just been binary before), in adding support for contacts and enabling access policies for attributes and other resources. Moreover, certified attributes are now also supported, which was mainly needed to verify the public key of hosts. (Clients are delivered with the public key of *virtualid.ch* and can therewith validate the public key of *xdf.ch*.) Unfortunately, a general mechanism to issue certificates is not yet in place and therefore attribute-based access control is not possible so far. The **cryptography** required for *anonymous credentials* (see A.3.), however, is fully implemented and clients use it to authenticate themselves at other hosts (and can thereby no longer be distinguished from each other). This lack of certification is also the reason why it is not yet possible to run one's own hosts, since clients would not be able to verify their authenticity. (The public key of the host *xdf.ch* has been signed with an internal procedure.)

Thanks to the fact that *Virtual ID* and its type system, which is called *Extensible Data Format* or short *XDF*, are so simple to extend, the development of the desired **file service** required only relatively little effort. The functionality that is related to the actual sharing of files was realized with just one database table and four methods on the client- and the server-side to issue respectively handle the necessary requests. A little bit more work was to write the thread that loops every second through the synchronized files to check if anyone of them has changed and needs to be uploaded or downloaded.

Unlike the other standards, *Virtual ID* synchronizes the files directly in your **file system**. On startup, the client creates the folder *File Service* in your home directory with two subfolders called *Internal* and *External* for your own files respectively the ones of your friends. The file structure should look like this:



The file structure for **synchronization**.

All the files stored in *Internal* are actively **synchronized** with the host which you have specified when starting a client for the first time. The folder *External* though just imports all the files that your contacts have shared with you to folders that are labelled with their respective names. These files are only to read: Whatever you change there will not be reflected on any server! Furthermore, the synchronization process does not keep track of changes, it just mirrors added or modified files for now. As a consequence, a file that you delete will be immediately added back again (unless you take the option dedicated for this in the client's command line interface). If you want to see external files synchronization, you can add *kaspar.etter@virtualid.ch* as a contact to get my public files.

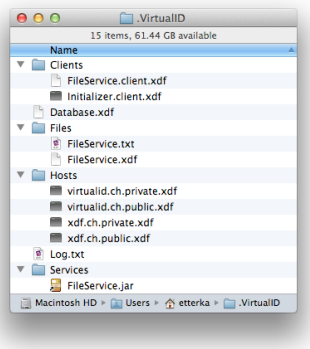
The **visibility** of your files can be specified in the implicitly created and automatically updated file *Access.txt*. The syntax for each line is first the file name and a colon followed by a so-called *passive expression* that determines the access policy. Basically, you can separate several identifiers with a plus sign, use the keyword "everybody" to make a file public or enter a zero to grant access only to your contacts. As *expressions* are not yet checked on the client-side, you will receive ugly error messages if you make a mistake. (Just delete that line then.)

In order to **execute** the client, you need to have access to a (preferably local) *MySQL* installation. On initial startup, the client guides you through the configuration of the database and the creation of your *virtual identity* (abbreviated as *VID*).

The client creates a hidden folder named *.VirtualID* in your home directory to store the database **configuration** and the client's secret, which is used for authentication. If you delete this folder, you will no longer be able to access your *VID*! You can just create a new one though. The subfolder *Files* contains the log of the file service as well as the client's name, which is important in case you want to authorize a second client on a different machine. (This setup synchronizes writable files!)

The other folders and files in *.VirtualID* are only needed if the code is run as a **server**, which will create a private and a public key for each host and log all incoming requests in the file *Log.txt*. As services are developed independently of the core server, their implementations can be placed as a *Java Archive* (*JAR*) in the *Services* folder, from where they will be loaded dynamically, i.e. during the runtime of the server.

This hidden folder has the following content, where normal users do not have the files with the black and brownish icons:



The file structure for **metadata**.

Suitability

- **Scope of Identity**: *Virtual ID's* identifiers are valid *globally*.
- **Attribute Access**: A client can retrieve attributes of other users if its own *virtual identity* is authorized to access them.
- **Authorization**: Since authorizing clients is at the heart of *Virtual ID*, these can manage files on behalf of their users.
- **Accountability**: Each request that affects a server's state has to be signed and cannot be repudiated by the user.

Usability

- **Requirements**: To take full advantage of its features, new clients are needed for *Virtual ID*. Like in the case of e-mail, however, the protocol might also be accessed via the *Web*.
- **Attribute-Based Access Control**: Though not yet realized, *Virtual ID* is designed to support this functionality later on.
- **Contacts**: Users can give a client access to their contacts so that it can look up their attributes and fetch their files.
- **Authentication**: If you want to access the files of your co-workers and friends, you trust them to choose a reputable host. For this reason, clients authenticate as you implicitly.

Security

- **Phishing**: A client never reveals its secret key and can in the worst case be accredited at the wrong *virtual identity*, which is at most a privacy and never a security problem.
- **Evidence**: Client authentication is based on cryptography.
- **Compromise**: In *Virtual ID*, clients are authorized according to the principle of least privilege, limiting potential harm.
- **Logout**: Clients do not keep sessions and need no logout.

Privacy

- **Calling Home**: Since the public key of hosts can be cached for an year, they are not consulted during authentication.
- **Wrong Consent**: How clients authenticate themselves is specified for each contact so that all requests can be fully automated, since users are never asked to approve them.
- **Attribute Selection**: Those attributes which are part of an authentication are also specified per contact. Additionally, each attribute has some visibility for asynchronous access.
- **Pseudonymity**: The credentials of *Virtual ID* enable users to access attribute-restricted resources fully anonymous.

5.8. Evaluation

Comparison

The above **analysis** of the four standards based on the criteria presented in 5.3. resulted in the following table:

Name	SAML	OpenID	WebID	Virtual ID
Suitability	**––**	**–**	**+**	**++**
– Scope of Identity	–	+	+	+
– Attribute Access	–	–	+	+
– Authorization	–	–	+	+
– Accountability	–	–	–	+
Usability	**+**	**–**	**+**	**+**
– Requirements	+	+	+	–
– ABAC	+	–	+	+
– Contacts	–	–	+	+
– Authentication	+	–	–	+
Security	**––**	**––**	**+**	**++**
– Phishing	–	–	+	+
– Evidence	–	–	+	+
– Compromise	–	–	–	+
– Logout	–	–	+	+
Privacy	**–**	**=**	**=**	**++**
– Calling Home	–	–	+	+
– Wrong Consent	–	+	–	+
– Attribute Selection	–	+	+	+
– Pseudonymity	+	–	–	+

The **differences** between the four standards are remarkable. The next section gives some advice on how to best use them.

Recommendations

The following statements summarize the important findings of the conducted study and give you a good understanding of the standards' respective advantages and disadvantages:

- **SAML**: The bad score on security and privacy is not really the fault of *SAML* but rather due to the largely negligent implementation in the *SWITCHaai* federation. Not a single effort is made against phishing or in favor of an improved consent procedure during authentication. The misleading messages on logout is just the peak of the shortcomings. I would still recommend it though if you mainly target an academic audience in Switzerland or require reliable data from your users. However, it is not suited for this scenario.
- **OpenID**: Although once quite popular, *OpenID* has lost a lot of its shine and shares many of its problems with *SAML*. It is useful if you want a simple method to allow users to log on with their existing *Google* or *Yahoo!* account. Apart from that, nobody knows their *OpenID* identifier and this *OpenID Provider driven identifier selection* mostly destroys the scenario. Conclusion: Also not suited for this use case!
- **WebID**: In my opinion the big winner of this study is *WebID*, despite being quite immature and insufficiently supported by current browsers. The usage of client certificates means a paradigm shift in *Web*-based user authentication and has many benefits. Its only weakness is also its biggest strength: Exclusively relying on existing technologies, *WebID* is not radical enough. Thus it cannot change the cryptography.
- **Virtual ID**: Being designed for optimal support of scenarios like this one, it is not surprising that *Virtual ID* achieves the maximum grade. It has well consolidated but nevertheless extremely powerful concepts. For easier adoption, one can build a *Web* front end with certificate-based authentication.

6. Conclusions

6.1. Contributions

Characterization

For people not familiar with identity management systems, it is tough to get an accurate picture of the numerous standards and even more difficult to compare them in a thorough way. The presented *characterization* is the first holistic attempt to systematically identify all conceptually important **aspects** of single sign-on and identity standards. Its primary purpose is to facilitate and standardize the creation of *standard profiles* that enable the reader to *grasp* and *classify* existing and new proposals in the *identity landscape* in a short amount of time.

Although many of the *characteristics* can be combined in arbitrary ways, there seem to be some **sweet spots** around which standards cluster. The major two distinctions concern whether they focus on the *users* or their *clients* and whether access is granted to *internal* or *external* resources. Based on this, I suggest the following **classification** of standards (4.7.):

Standard	**User**	**Client**
Internal	(1) Authentication	(2) Authorization
External	(3) Federation or Sign-On	(4) Identity

Comparative Study

In order to show the practical relevance of this classification, I designed a study in which I compared four standards on the basis of a concrete **scenario**. Each standard was employed to implement a *decentralized file service*, where users can select a provider of their choice and are still able to share content with their friends and co-workers utilizing other hosts.

The four **standards** were selected by means of the outlined *classification* with a representative from each class which has support for external access control plus my own standard.

The **criteria** were derived from the *characterization* as well. Grouped into four categories, they result in the grades (5.8.):

Name	**SAML**	**OpenID**	**WebID**	**Virtual ID**
Suitability	−−	−	+	++
Usability	+	−	+	+
Security	−−	−−	+	++
Privacy	−	=	=	++

The given scenario clearly demands a *real* **identity standard**!

6.2. Future Work

Characterization

The list of characteristics is not fixed but should evolve with new insights and requirements. Though I was first skeptical about the benefits of an actual implementation, I learned a lot during the deployment of the four standards. This would certainly affect the *characterization* if I started anew with my master thesis. Among the things I would do **differently** are:

- better conceptualization of the *authentication mechanism* (e.g. whether the messages are relayed through the client);
- split *privacy* and *security* concerns according to the study;
- determine *who* controls the submission of the evidence (with proper challenge-response techniques on the client-side, phishing is no issue (but man-in-the-middle attacks));
- and depict *partial* and *directed identities* more accurately.

Comparative Study

In my opinion, the *criteria* of the study are adequate and also quite informative. The **implementation** could be significantly improved from a usability perspective, but the standards are likely to profit equally from a better user interface. Moreover, a native application for *WebID* could demonstrate its ability to span both the *Web* and the desktop. A desirable extension of the *scenario* would be to allow modifications to external files and thereby highlight the importance of accountability. This feature is not supported in the current version since it would complicate the user experience (also of *Virtual ID*) too much.

It is always possible, however, to evaluate more standards and show their respective strengths and weaknesses. Some potential candidates can be found in the chapter 5.1.

6.3. Semantic Web

Vision

The term *Semantic Web* was coined by Tim Berners-Lee, who is known as the inventor of the *World Wide Web*, in an article from 2001, where he outlined his vision of a *Web* that is also **understandable** by machines and not just humans [BHL01]. Until now, browsers can only display the content of *Websites* to users and wait for their reactions but have a hard time to find relevant information on their own. Once computers can process semantic data on behalf of their users, the dream of having a *personal digital assistant* finally comes true.

Requirements

The promise of the *Semantic Web* requires two ingredients:

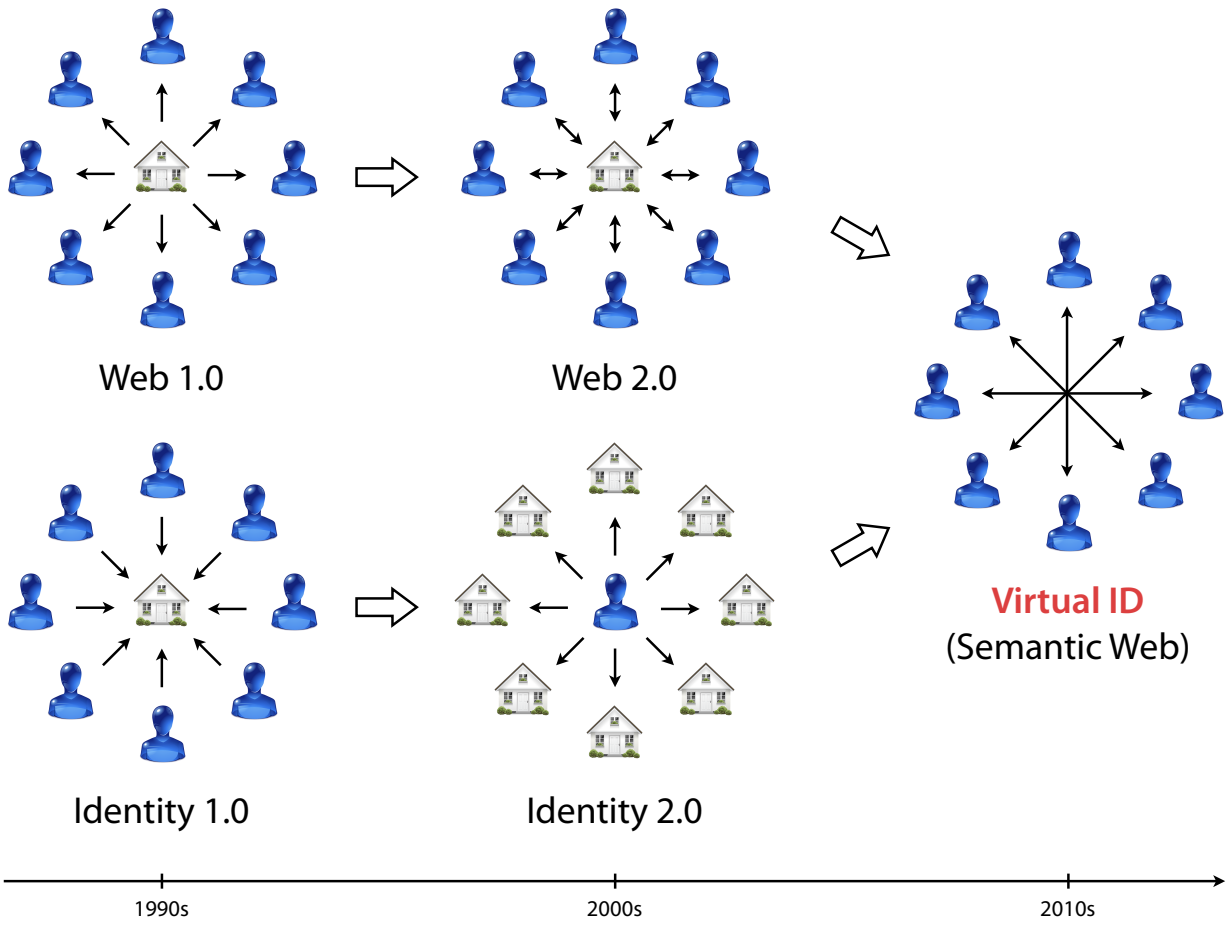
- **Identities**: In order to *trust* some information, you need to know its *source*. On the other hand, whenever your device orders something for you, it enters a *contract* between two legal entities. Furthermore, most data is not meant to be *public* but still contains a lot of semantics, so you would like to have a way to disclose resources only *selectively*.
- **Automation**: For being able to assist you with your tasks, clients need to be *authorized* with the proper permissions, which includes the capability to act *on your behalf* towards others. Only if no more human interaction is required, it is feasible to automate the procedures for *digital agents*.

Realization

What I tried to demonstrate with the present master thesis, is that *federation* and *sign-on standards* do not meet the two requirements for the realization of the *Semantic Web* as they neither allow the user to make **trustworthy statements** nor to authorize clients for automated exchange of information. They just allow service providers to *recognize* the same user.

To the best of my knowledge, there exist currently only two **general-purpose proposals** that are technically qualified to enable the *Semantic Web*: *WebID* and *Virtual ID*. Considering the enormous efforts that are needed to accomplish such a big transition, it is probably worth to rely on the technically and conceptually more mature of the two: *Virtual ID*.

Laying the foundation for the realization of the **Semantic Web**:



Virtual ID is a protocol that constitutes an open **identity layer** for the *Internet* and a **semantic alternative** to the *World Wide Web*. It allows you to prove your identity towards others and to look up attributes of others in a decentralized manner. Being freely extensible with services, *Virtual ID* aims to **supersede proprietary platforms** by establishing a framework of open standards.

A. Technologies

A.1. Internet

Internet Protocol (IP)

The *Internet* originates from the *ARPANET* (*Advanced Research Projects Agency*). In 1974, different networking methods were unified by an internetwork protocol, which became ten years later the only approved protocol on the *ARPANET*. The goal of this *Internet Protocol* (better known as *IP*) is to **route packets** between any two devices that are connected to the *Internet*. Each end point is denoted by an *IP address*, a 32-bit number that reflects the hierarchical organization of the *Internet*.

Transmission Control Protocol (TCP)

As the *Internet Protocol* provides no *reliability*, the *Transmission Control Protocol* (*TCP*) is used to **transport packets between processes** (a *process* is an instantiation of a program), where so-called *port numbers* designate the *process* on each of the communicating devices. When a packet is received, it is the task of the *Operating System* to deliver it to the right *process*. Additionally, *TCP* recognizes lost packets and such which are out of order to provide *reliable* and *in-order* data transfer.

Secure Sockets Layer (SSL)

The *Secure Sockets Layer* (*SSL*) respectively its successor called *Transport Layer Security* (*TLS*) are cryptographic protocols that provide *confidentiality* and *integrity* for *secure* communication over the *Internet*. They use *public-key cryptography* (see A.3.) to authenticate the opposite party and *symmetric encryption* to ensure that nobody can decipher a transmitted message.

Domain Name System (DNS)

Instead of referencing machines by their *IP addresses*, a more human-friendly solution was introduced in 1983: the *Domain Name System* (*DNS*). It acts as a distributed "phone book" to look up the *IP addresses* of *domain names*. A *domain name* is a sequence of labels separated by dots, e.g. *www.ethz.ch*.

Uniform Resource Locator (URL)

A *Uniform Resource Identifier* (*URI*) is a special *string* (a string is a sequence of characters) that uniquely identifies a resource. A *Uniform Resource Locator* (*URL*) is a *URI* that specifies *where* and *how* a resource can be accessed on the *Internet*. *URLs* are written as *scheme://domain:port/path?query_string#fragment*.

A.2. World Wide Web

Hypertext Transfer Protocol (HTTP)

The *Hypertext Transfer Protocol* (*HTTP*), which you know from the scheme of *URLs*, is the so-called *Application Layer* protocol of the *World Wide Web*. A client, usually known as a browser, initiates a *TCP* connection to a server on port 80 to **retrieve the document** as specified by the *path* and the *query string* of the *URL*. Since *HTTP* is a stateless protocol, user sessions are most commonly maintained by *cookies*, which are sent by the browser in every request. As *TCP* provides neither confidentiality nor authenticity, *HTTP* is often used over *SSL/TLS*, which results in the well-known *HTTPS* scheme.

Hypertext Markup Language (HTML)

The *Hypertext Markup Language* (*HTML*) describes the content and structure of *Web* pages. It is text-based and uses *tags* like * * (for opening) and * * (for closing) to mark its content. Browsers interpret these *tags* and display pages accordingly.

Extensible Markup Language (XML)

The *Extensible Markup Language* (*XML*) defines a set of rules to encode information in a machine-readable format. It can be seen as a generalization of *HTML* that is widely used to transfer data between programs and machines. All these standards are specified by the *WWW Consortium* (*W3C*).

JavaScript (JS)

JavaScript (*JS*) is a scripting language that enables dynamic *Web* Pages and was introduced by *Netscape* in 1995. Unlike other scripting languages, it is executed on the client-side by the browser and can *modify* a page's *content* and *layout*.

Resource Description Framework (RDF)

The *Resource Description Framework* (*RDF*) is a *W3C* standard to describe *Web* resources using a variety of notations and formats. It consists of *subject-predicate-object* expressions.

A.3. Cryptography[1]

Symmetric-Key Cryptography

In symmetric-key cryptography, the communicating parties share the **same key** for encryption and decryption. This was the only encryption method publicly known before the year 1976, when Whitfield Diffie and Martin Hellman proposed to use different but related keys for encryption and decryption.

Public-Key Cryptography

The main problem with symmetric-key cryptography is the management of keys since each pair of parties must share a different key or rely on a trusted third party to distribute one on demand. By using a pair of related but **distinct keys**, it is possible to share the one for encryption while keeping the one for decryption secret. The former is known as the *public key*, the latter as the *private key*. For *confidentiality*, it has to be hard to compute the private key from the public key.

Digital Signatures

The essential feature of signatures is that they are easy for the author to produce but hard for others to forge. This is achieved by using the *private key* for **signing** and the *public key* for the **verification** of a signature, which anyone can do.

Public Key Infrastructure

Since it is not possible to authenticate someone you have never met before, you need to trust a third party to confirm their identity. This service is usually provided by **certification authorities** (*CAs*) that issue certificates on the public key of registered users and thereby making them accountable.

Anonymous Credentials

By signing only the *hash*, classical public-key cryptography requires the complete message for the verification of digital signatures. In order to protect one's **privacy**, it is desirable to reveal the content of a certificate more selectively and *prove* nevertheless certain facts about *hidden* statements. This can be used to show certificates *anonymously* as credentials and still provide *evidence* that they were issued to the same user.

The Onion Router (Tor)

As we have seen in the previous section, computers on the *Internet* are uniquely identified by their *IP address*. In order to still achieve *anonymity* on the *Network Layer*, you can join the free volunteer network of nodes that relay each other packets to prevent network surveillance and traffic analysis: *Tor* [Tor].

A.4. Vulnerabilities

Protocol

A first family of vulnerabilities that an attacker might exploit concern the design of a protocol itself. If not properly done, the evil party can *eavesdrop* on the communication channel to learn confidential information, *drop*, *insert* and *replay* any message or be the *man-in-the-middle* that impersonates the respective other party. If randomness is not taken from a big enough range (e.g. by relying on passwords), an attacker can also run a *brute force attack* by simply trying all possible keys or letter combinations. A further option is to trick the victim into voluntarily releasing its secret to a fake *Web* page, which is called *phishing* and is currently the biggest security threat.

Implementation

Many of the vulnerabilities in practice, however, are not due to flawed specifications but due to careless implementations. An attacker can do some *cross-site scripting* (*XSS*) by injecting malicious code that is executed on the client-side (usually in *JavaScript*) or make *cross-site request forgery* (*XSRF*) by posting harmful form data and actions to a *Website* at which the user is currently logged in. If sessions are not handled properly, an attacker could also *fixate the session* of a victim (which was an issue in *OAuth*.) The major problem though is that negligent identity providers can also damage the reputation of service providers, which can deter a company from using a protocol.

[1] This section uses material from a previous paper of mine, which can be found on *www.virtualid.ch*.

B. Standards

B.1. Comparison

The following table compares the standards *SAML*, *OpenID*, *WebID* and *Virtual ID* according to the **characterization** of chapter 4.:

Name	**SAML**	**OpenID**	**WebID**	**Virtual ID**
General				
– Logo	saml xml·org	OpenID	WebID	Virtual ID
– Website	saml.xml.org	openid.net	webid.info	www.virtualid.ch
– Proponent	SSTC of OASIS	OpenID Foundation	WebID Community Group	Virtual ID Foundation
– Intention	Have an open standard for authentication data	Provide a safe, fast and easier login to Websites	Replace the traditional, weak login on the Web	Supersede proprietary platforms of the Web
– Target	Mainly used as SSO in commerce/academia	All users of Websites	All users of Websites	All natural and artificial persons in any services
History				
– Origins	Security Assertion Markup Language in January 2001	By Brad Fitzpatrick for LiveJournal in May 2005	Formerly FOAF+SSL at a W3C Workshop in 2009	Devised by Kaspar Etter at ETH Zurich in 2010
– Development	Active	Active	Active	Active
– Adoption	Industry standard	Decreasing popularity	–	–
– Support	Excellent (SWITCHaai)	Many online forums	Mailing list	Limited
– Specification	Overly complex	Many options but clear	Incomplete but clear	Incomplete but clear
– Community	Big	Huge	Small	None
– Libraries	Apache (Shibboleth)	Many modules/plugins	PHP and Java (?)	Java (not yet available)
– Protocol	Client-server	Client-server	Client-server	Client-server
– Version	2.0	2.0	Unfinished	Unfinished
– Patents	No licenses required	Non-assertion covenants	Disclosure obligations	Guaranteed by IBM
– Extensions	Many proprietary ones	Attribute Exchange, etc.	–	Various services

Name	SAML	OpenID	WebID	Virtual ID
Identification				
– Identity	Unique (if ensured)	Arbitrary	Arbitrary	Arbitrary
– Scope	Federated	Global	Global	Global
– Creation	Authority	User	User	Client
– Deletion	Supported	Identifier remains	Identifier remains	–
– Merging	–	–	–	Supported for persons
– Identifier	Unique ID (opaque)	URL or XRI	Any URI	Identifier like e-mail
– Relocation	–	Delegated or redirected	Possible: HTTP Redirect	Approved redirects
– Reusability	–	Allowed with fragments	Unspecified	Forbidden
– Interaction	Implicit (at ID provider)	Unspecified (manually)	Explicit (prompt)	Implicit
– Frequency	Once per browser session	Per SP and brows. session	Per SP and brows. session	Once per client
– Submission	SAML assertion	Unspecified (form field)	SSL client certificate	Request signature
– Attributes	Implicit by SP (static)	Explicit by the SP (dyn.)	–	Implicit by user (static)
– Linkability	Anonymous (pseudo.)	Identifiable (or pseudo.)	–	Anonymous
– Revocation	? (by identity provider)	(By identity provider)	–	By certifying authority
Authentication				
– Mechanism	Redirections with crypto	Redirections with secret	Cryptographic certificate	Cryptographic signature
– Performance	15 mess. with signatures	11/13 mess. with hashes	SSL handshake (5 mess.)	TCP with crypto (1 mess.)
– Compatibility	Any user authentication	Any user authentication	–	–
– Requirements	Web server module	–	SSL certificate on server	New clients and servers
– Vulnerabilities	Phishing, no logout	Phishing, DNS and XSS	Wrong X.509 certificate	DoS, single root in PKI
– Privacy Issues	Calling home, consent	Calling home problem	ID leaked to wrong SP	–
– Identity Provider	Online	Online	Online	Offline (key cached; < 1 h)
– Discovery	Central page (WAYF)	IP referenced in Meta	Alternative Name in X.509	Part of the identifier
– Relocation	– (not even initial choice)	Delegate to the new IP	Possible: HTTP Redirect	Supported
– Verification	Known list of providers	– (X.509 Certificates)	X.509 Certificates	Centralized PKI
– Rotation	3 years	–	Validity of certificate	2 years
– Regulation	Central registration	–	–	By Virtual ID Foundation
– Service Provider	Central registration	–	–	–
– Evidence	Unspecified (password)	Unspecified (password)	Cryptographic key	Cryptographic key
– Factor	Unspecified (knowledge)	Unspecified (knowledge)	Possession	Possession
– Control	Unspecified (user)	Unspecified (user)	Client	Client
– Rotation	–	–	–	2 years
– Interaction	Unspecified (manually)	Unspecified (manually)	– (during identification)	–
– Frequency	Once per browser session	Once to IP (per session)	– (once per SP and session)	–
– Submission	Unspecified (form field)	Unspecified (form field)	SSL certificate	Request signature
– Compromise	Unspecified (identity)	Unspecified (identity)	Unspecified (identity)	Client
– Revocation	Unspecified (change pw.)	Unspecified (change pw.)	Remove public key	Deaccreditation
– Session	Shibboleth Daemon	By SP (cookies)	SSL Connection	Signatures
– Logout	– (in case of SWITCHaai)	At the IP and all SPs	Browser dependent	Unspecified (client-side)
– Encryption	At least setup encrypted	–	Recommended	Mandatory
– Forward Secrecy	–	–	– (optional in SSL)	–

Name	SAML	OpenID	WebID	Virtual ID
Authorization				
– Clients	–	–	Supported	Supported
– Rights	–	–	Unspecified (all)	Attributes and contacts
– Permissions	–	–	Unspecified (all)	Attributes and services
– Restrictions	–	–	Unspecified (none)	Attr., contacts, history
– Management	–	–	Unspecified (Web UI)	Trust-based client order
– Accreditation	–	–	Unspecified	Commit to secret value
– Deaccreditation	–	–	Removal from profile	Remove (< 1 h effective)
– Users	–	–	–	Supported (with roles)
Accountability				
– Signatures	–	–	Nowhere	Requests and responses
– Liability	–	–	–	Oral (written intended)
– Duration	–	–	–	1 year
– Revocation	–	–	–	–
– Framing	Possible	Possible	Possible	Prevented by auditing
Attributes				
– Storage	Predefined	– (Attribute Exchange)	Extensible	Extensible
– Format	String (XML)	– (UTF-8 string, id. by URI)	String (RDF)	Arbitrary (XDF)
– Regulation	By the federation	–	–	By attribute author
– Transmission	Supported (all attributes)	Supported (all attributes)	–	Only certified attributes
– Access	–	–	Unspecified (possible)	Specified per attribute
– Policy	–	–	Unspecified (possible)	Passive expressions with contexts and attributes
– Caching	–	–	Unspecified (possible)	Specified per attribute
– Certification	(Trusted identity providers)	–	–	Specified per attribute
– Authority	(Identity providers)	–	–	Hierarchical (single root)
– Revocation	–	–	–	–
– Liability	?	–	–	Given (only trust in root)
Contacts				
– Format	–	–	Friend of a Friend (FOAF)	Identifiers in XDF
– Structure	–	–	Unspecified	Hierarchical contexts
– Disclosure	–	–	Public by default	Designated attributes
– Notification	–	–	–	Knock with preferences
Conclusion				
– Classification	Federation Standard	Sign-On Standard	Identity Standard	Identity Standard
– Suitability	Among organizations	Primarily for the Web	Web and Apps	Distributed services
– Summary	Mature standard for businesses/universities	Simple to authenticate millions of Web users	Badly supported by browsers, nice idea	Clean concepts and state-of-the-art cryptography

C. References

Standards and Initiatives

- **ABC4Trust**: *www.abc4trust.net*
- **BrowserID**: *www.persona.org*
- **CardSpace**: (discontinued)
- **FIDIS**: *www.fidis.net*
- **FOAF**: *www.foaf-project.org*
- **Higgins Project**: *www.eclipse.org/higgins/*
- **Kerberos**: *web.mit.edu/kerberos/*
- **Liberty Alliance**: *www.projectliberty.org*
- **NSTIC**: *www.nist.gov/nstic/*
- **OAuth**: *oauth.net*
- **OpenID**: *openid.net*
- **OpenPGP**: *www.openpgp.org*
- **PICOS**: *www.picos-project.eu*
- **PRIME**: *www.prime-project.eu*
- **PrimeLife**: *www.primelife.eu*
- **SAML**: *saml.xml.org*
- **Shibboleth**: *shibboleth.net*
- **SuisseID**: *www.suisseid.ch*
- **Tor**: *www.torproject.org*
- **Virtual ID**: *www.virtualid.ch*
- **WebID**: *webid.info*
- **WS-Federation**: *docs.oasis-open.org/wsfed/federation/*

Request for Comments (RFCs)

- **RFC2616**: *www.ietf.org/rfc/rfc2616* (*HTTP/1.1*)
- **RFC2617**: *www.ietf.org/rfc/rfc2617* (*HTTP Authentication*)
- **RFC4880**: *www.ietf.org/rfc/rfc4880* (*OpenPGP*)

[BCG06] Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Gross, Dieter Sommer; User Centricity: A Taxonomy and Open Issues; Proceedings of the 2006 Workshop on Digital Identity Management, Alexandria, VA, USA, November 3, 2006.

[BHL01] Tim Berners-Lee, James Hendler and Ora Lassila; The Semantic Web; Scientific American Magazine, May 2001.

[BS12] Eleanor Birrell and Fred B. Schneider; Federated Identity Management Systems: A Privacy-based Characterization; 2012.

[Cam04] L. Jean Camp; Digital identity; Technology and Society Magazine, IEEE 23.3, pp. 34–41, 2004.

[Cam05] Kim Cameron; The Laws of Identity; Microsoft Corporation, 2005.

[CK01] Sebastian Clauß and Marit Köhntopp; Identity management and its support of multilateral security; Computer Networks 37.2, pp. 205–219, 2001.

[FP12] Md. Sadek Ferdous and Ron Poet; A comparative analysis of Identity Management Systems; International Conference on High Performance Computing and Simulation (HPCS), pp. 454–461, July 2012.

[HH12] Michael Hackett and Kirstie Hawkey; Security, Privacy and Usability Requirements for Federated Identity, 2012.

[HJS09] Jaap-Henk Hoepman, Rieks Joosten and Johanneke Siljee; Comparing Identity Management Frameworks in a Business Context; The Future of Identity in the Information Society, pp. 184–196, Springer Berlin Heidelberg, 2009.

[KZR11] Tobias Kölsch, Jan Zibuschka and Kai Rannenberg; Privacy and Identity Management Requirements: An Application Prototype Perspective; Digital privacy, pp. 735-749, Springer Berlin Heidelberg, 2011.

[PH08] Andreas Pfitzmann and Marit Hansen; Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology; *dud.inf.tu-dresden.de/Anon_Terminology.shtml*, 2007.

[PK08] Radia Perlman and Charlie Kaufman; User-centric PKI; Proceedings of the 7th symposium on Identity and Trust on the Internet, pp. 59–71, ACM, 2008.

[SB12] Christopher Staite and Rami Bahsoon; Evaluating Identity Management Architectures; Proceedings of the 3rd international ACM SIGSOFT symposium on Architecting Critical Systems, pp. 11–20, ACM, 2012.

[SMS12] Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, and Meiko Jensen; On Breaking SAML: Be Whoever You Want To Be; Proceedings of the 21st USENIX Security Symposium, Bellevue, WA, USA, 2012.

[VWZ12] Meilof Veeningen, Benne de Weger, and Nicola Zannone; A Formal Privacy Analysis of Identity Management Systems; arXiv preprint arXiv:1206.7111, 2012.

[WCW12] Rui Wang, Shuo Chen, and XiaoFeng Wang; Signing Me onto Your Accounts through Facebook and Google: a Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services; IEEE Symposium on Security and Privacy (SP), pp. 365–379, IEEE, 2012.