Identity Management Infrastructure Protocols for Privacy-enabled SOA

Editors:	Sascha Koschinat (GUF)
	Kai Rannenberg (GUF)
	Gökhan Bal (GUF)
Reviewers:	Marc-Michael Bergfeld (GD)
	Gregory Neven (IBM)
	Jan Schallaböck (ULD)

Identifier:	D6.1.1
Type:	Deliverable
Class:	Public
Date:	September 14, 2009

Abstract

In recent years, Identity Management (IdM) evolved into an essential component of service oriented infrastructures, where among other data, an effective exchange of sensitive data between different subsystems is an inherent part of the whole architecture. But it is noticeable that such infrastructures still lack adequate privacy mechanisms. This report uses the example of an electronic CV system and first develops requirements for the establishment of privacy-enabled service oriented architectures. Then, several existing IdM-protocols are investigated with a focus on their applicability as privacy-enhancing mechanisms for IdM infrastructures. Moreover, the challenges that still have to be faced for the deployment of identity management infrastructure protocols (IdMIP) are depicted.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216483 for the project PrimeLife.

Members of the PrimeLife Consortium

1.	IBM Research GmbH	IBM	Switzerland
2.	Unabhängiges Landeszentrum für Datenschutz	ULD	Germany
3.	Technische Universität Dresden	TUD	Germany
4.	Karlstads Universitet	KAU	Sweden
5.	Università degli Studi di Milano	UNIMI	Italy
6.	Johann Wolfgang Goethe-Universität Frankfurt am Main	GUF	Germany
7.	Stichting Katholieke Universiteit Brabant	TILT	Netherlands
8.	GEIE ERCIM	W3C	France
9.	Katholieke Universiteit Leuven	K.U.Leuven	Belgium
10.	Università degli Studi di Bergamo	UNIBG	Italy
11.	Giesecke & Devrient GmbH	GD	Germany
12.	Center for Usability Research & Engineering	CURE	Austria
13.	Europäisches Microsoft Innovations Center GmbH	EMIC	Germany
14.	SAP AG	SAP	Germany
15.	Brown University	UBR	USA

List of Contributors

This deliverable has been jointly authored by multiple PrimeLife partner organisations. The following list presents the contributors for the individual parts of this deliverable.

Chapter	**Author(s)**
Executive Summary	Gökhan Bal (GUF)
Chapter 1: Introduction	Gökhan Bal (GUF)
Chapter 2: Scenarios	Sascha Koschinat (GUF), Andreas Leicher (GUF), Gökhan Bal (GUF)
Chapter 3: Evaluation Criteria and Requirements	Gökhan Bal (GUF), Andreas Leicher (GUF), Sascha Koschinat (GUF)
Chapter 4: Candidate Solutions for IdMIP	Gökhan Bal (GUF), Andreas Leicher (GUF), Uli Pinsdorf (EMIC), Stuart Short (SAP), Gregory Neven (IBM)
Chapter 5: Applicability of Candidates to IdMIP	Andreas Leicher (GUF), Gökhan Bal (GUF), Uli Pinsdorf (EMIC), Stuart Short (SAP)
Chapter 6: Evaluation of IdMIP	Andreas Leicher (GUF), Gökhan Bal (GUF), Sascha Koschinat (GUF), Uli Pinsdorf (EMIC), Stuart Short (SAP)
Chapter 7: Conclusion	Gökhan Bal (GUF)

Executive Summary

Service Oriented Architectures (SOAs) are widely used in infrastructures when different entities (e.g. service providers) from different domains participate. The interactions and exchange of data between different service providers (SP) can directly be applied to the architecture model. In a lot of scenarios (e.g. flight booking, hotel reservation, car rental, etc.) there is a special need for mechanisms that help the different SP to act jointly, as in most cases customers will engage in more than one service. Identity Management (IdM) is an evermore utilized approach that helps to prevent unnecessary overhead when trying to link processes with the respective customer accounts. Therefore it is a key factor for enabling cross-SP business relationships. But agreeable IdM has to come with appropriate privacy mechanisms. Cross-SP scenarios are where privacy risks notably exist. This can be exemplified in the exchange of personal data about a customer between two or more service providers. If no adequate access control policies are established that consider the customer's privacy demands, privacy problems can hinder such architectures from being accepted widely.

Within the EU FP7 project "PrimeLife" there are efforts to solve such privacy problems by developing adequate solutions in the areas of human computer interaction, configurable policy languages, web service federations, infrastructures and privacy-enhancing cryptography. In Work Package 6.1 (WP6.1) there is a special focus on IdM infrastructures. The main point of investigation is the analysis of the problems for the deployment of privacy-enabled IdM infrastructures. As one of the results of WP6.1, this document deals especially with the requirements to be considered when developing mechanisms that shall enhance IdM infrastructures with privacy-enabling features. Furthermore, several existing IdM-related protocols are presented and evaluated against these requirements. The objective is to learn if existing protocols are sufficient to achieve a higher level of privacy in infrastructures. If not, the problems that hinder the establishment of privacy-enabled infrastructures shall be extracted. Such statements should only be made in conjunction with a deeper investigation of a concrete scenario. Therefore, a job portal architecture serves as model in this report. Such a job portal has a manageable set of purposes. In the first place, it provides mechanisms for job applicants to publish electronic versions of their CV (eCV) with the ambition that a headhunter (another job portal participant) will contact them with lucrative job offers. Nevertheless, the applicant may want to disclose only a minimal set of personal data. On the other hand, a headhunter wants to find the best experts for a given job position. Therefore, he has a demand for as much information about the applicant as possible. Beyond that, the information stated in the CVs has to be trustworthy, because a headhunter will base his initial decisions on this data. These simple examples of application point out that finding simple solutions for the IdM and privacy problem in service oriented infrastructures is quite impossible.

Thus, the work for this report will not try to find the ultimate solution in form of a new protocol or standard. This work shall sketch out how far existing protocols can be employed for enhancing privacy in IdM infrastructures. This includes setting up an architecture scenario, requirements that make up viable privacy-enhancing infrastructure protocols and the evaluation of several existing protocols against these requirements. As a result, the (still existing) challenges will be identified to support further work in finding solutions for them.

Contents

List of Figures

Chapter 1

Introduction

More and more Identity Management (IdM) is becoming a key function in the administration of private and business IT systems. Shifting tasks and processes from the physical world into the digital world undoubtedly facilitated new business models and leveraged tasks, which hardly were feasible without the potentials of modern computer systems. Consequently, appreciating the economic value of "digitalizing" their processes, companies adapted these by establishing information technology wherever saving of costs could be reached. One famous example is the shifting from physical mails to digital e-mails. A similar observation can be made in the area of employee, customer or citizen management, meaning the administration of identity related information. These tasks also have been transferred to the digital world by now.

Identity Management helps users of computer systems or companies to simplify the handling of services or optimizing the processes. The golden rule here is: The more information about the respecting person is available for computer systems, the more services can be improved in terms of user convenience, because services then have more opportunities to be optimized. This perception is also applicable to the web. There exist a lot of services that are quite popular and used by many people. These include social network services, instant messaging, online shops, career portals, multimedia sharing portals and so on. Even if varying in the extents, all these "web 2.0" representatives have an essential need for identity related information of the users.

As it has shown in a number of disciplines in IT security, the main focus of the implementation of such systems was the basic functionality. Consideration of factors like security, privacy or usability is more regarded as a burden for the diffusion and acceptance of the solutions. This can be illustrated in the bounded awareness of a lot of users to the (in fact limited set of) privacy settings of Social Network Services (SNS). This widespread point of view of developers to such technologies led to implications, which today show that a privacy-disaster looms. The lack of adequate control mechanisms for identity information is not only a major privacy concern, but also has effects in public reports about computer systems that failed in protecting identity related information in an appropriate manner. The origins of these failures are multi-facetted. For instance, this can be the lack of enforceable access control policies, usability problems or limitations in the interoperability of mostly differing protocols. All these problems build an obstacle for a widespread establishment of IdM infrastructures in a global scale.

Improving privacy enhancing technologies (PETs) to such an extent that similar "data leakages" will not happen anymore is just one school of thought. Maybe the more promising approach is to enforce one well-known purpose of data protection, and that is data minimisation.

Looking through long-ranging glasses, the previously mentioned problems can put the privacy of individuals of the Information Society at risk. This statement can be backed up by thinking about the high complexity of the administration and the control of personal data that in the course of time will diffuse in the massively distributed storage of the Internet. As a result, there is the risk that individuals will leave a life-long trail of personal data over which they won't have control. These problems also form an obstacle for the successful pervasiveness of IdM-enabled service oriented architectures (SOA). The more participant systems are involved in such a service architecture, the more difficulties it raises for the management of personal data.

Different service providers (SP) need different parts of the information. So, one of the challenging questions is, where to store which parts of the data under which access control policies? This and other similar challenges hamper the spreading of privacy-enabled IdM service oriented architectures. Having identified these types of privacy-related problems, "PrimeLife" faces them by developing solutions and mechanisms which could help to resolve the problems without compromising on functionality. Already published works include reports on existing protocols for privacy and identity management. There also is a listing and discussion of available open source initiatives dealing with this [PL08, PL09]. Solutions that PrimeLife aims to develop cover the areas of human computer interaction, configurable policy languages, web service federations, infrastructures and privacy-enhancing cryptography. Activity 6 of PrimeLife concentrates on identity management infrastructures and the problems associated with them with regard to privacy. In Work Package 6.1 of Activity 6 we have a more specific focus on web architectures, web service architectures and related architectures. The aim is to define requirements for the establishment of privacy-enhanced IdM service architectures. Moreover, we will investigate how privacy-enabled SOA can be implemented with already existing standards and protocols. Furthermore, this requires the establishment of criteria that have to be met, if such architectures shall be achievable.

1.1 Aim of this document

In this document we want to pick up a real-life IdM infrastructure scenario and examine how this scenario can be enhanced by privacy features. Instead of developing new protocols and standards, in this work we want to investigate how already existing, but maybe not that widely used protocols can be utilized as Identity Management Infrastructure Protocols (IdMIP), which enrich service oriented architectures by privacy and IdM related features. The main focus will be actually an evaluation of the applicability of selected existent privacy and IdM supporting protocols and standards. This evaluation should help in identifying the problems and challenges that arise when an interoperable and scalable IdM service infrastructure shall be realized. As a requirement for such an evaluation we will define a set of evaluation criteria which need to be fulfilled if privacy enhanced infrastructures shall be successful. Such a scenario should bear as much privacy-related challenges as possible, so a more in-depth evaluation can be done. This could include several parties and the need for the diffusion of personal data. We select a limited set of existing protocols and standards and analyze how these protocols can support implementing privacy enhanced infrastructures and what are the problems and challenges they cannot meet. The results are to help in giving a direction for the development of improved protocols with more capabilities for implementing privacy enhanced IdM infrastructures.

1.2 The Scenario

The chosen scenario-to-analyze is a job portal system, which is used by several parties with different interests, like applicants, headhunters, claim issuers and so on. The scenario is fully based on the eCV scenario introduced in the requirements document PrimeLife Heartbeat H6.3.1 [PH09]. All the parties have different interests and expectations with regard to the system. The scenario and the involved parties will be described in more detail in Chapter 2. We consider such a system suitable for this study, as it raises several privacy risks that arise due to the need for distribution of personal data over different service providers.

As a job portal system can be regarded as a broker for open job positions, it includes a facility for applicants to publish electronic CVs. Such data structures obviously contain personal information, which – in the interest of the applicant – should not fall into wrong hands, respectively be accessible only to a selected set of people. Already this example raises privacy related issues to be countered by protocols that bring more privacy. A further challenge comes from the CVs in such a system being electronic. Therefore the spreading and distribution of such documents is much easier than in classical paper-based approaches.

So, this scenario seems to be quite applicable for the evaluation. Furthermore, the scenario will be differentiated into two variants. The idea behind that is to have different settings with different demands and requirements for privacy enhancing mechanisms. Expectably, different scenarios will imply different legal technical and economical requirements and consequences. Beyond that, different scenarios possibly imply differing domains of interests, which in turn reflects on the trust relationships between the actors and consequently on the requirements for privacy enhancing features.

Chapter 2

Scenarios

This chapter introduces the Job Portal scenario and its flavours. Two main application scenarios exist, the Inhouse Platform Solution (IPS) and the Open Platform Solution (OPS). These variations of the architecture will be described in more detail after giving a general overview of the basics of the Job Portal architecture.

2.1 The Job Portal Scenario

The emerging use of social networks and Internet based platforms for job search and offers leads to an increased extent of business relationships, where the use of digital identities plays an important role. The identities used in the context of job search and job offer scenarios contain similar information to that in a classical CV. The classical, paper-based CVs are user-centric maintained by the respective user. The applicant decides on which information to provide to employers, e.g. the HR department of a company. They collect claims (e.g. certificates) for gained experience from third parties (e.g. schools, trainers, etc.) and include them in their CV to provide evidence for their work and educational experience.

The increasing business-use of the Internet opens the path to new applications which allow users to maintain an electronic CV (eCV). We present scenarios which deal with the establishment and management of eCVs, job advertisements by companies and the matching of job offers to appropriate candidates by examining the CV. The eCV, as the digital counterpart of the classical CV contains the same amount of privacy related information. Due to their nature, eCVs are considered as highly privacy sensitive. With migrating to eCVs, we are faced with the following challenges. On the one hand, an eCV should provide the same level of reliability for the receiver, e.g. an HR department, as the paper based versions with printed training and school certificates and claims. So, there should be means to verify the genuineness of an eCV. On the other hand, the person who publishes her eCV should have the ability to control and manage the spread and distribution of the personal data contained within the eCV. This raises the need for mechanisms for access control and privacy policies. Again, due to the digital nature of an eCV, solving these problems is far from a trivial issue. Classical approaches, like career-websites or communities allow a user to publish CV information to a multitude of prospective employers and interested parties. The published information is mostly private and sometimes should remain confidential or targeted to a restricted audience only. Nevertheless, most of the existing platforms merely have

coarse-grained approaches for privacy settings. Hence, privacy concerns and solutions that face them will play an important role in the acceptance of eCV applications.

In general, the following interested parties can be identified in the Job Portal scenario:

- The applicant (OPS) / employee (IPS) publishes information on his knowledge, experience and personal data (eCV). The information can e.g. be used by prospective employers to select and contact the user in case of a vacancy.
- The employer, e.g. represented by an HR manager or headhunter, etc. browses all available eCVs to find a suitable candidate among the presented eCVs for a given job position. Depending on the scenario, the respective implementation and the privacy policy settings for the eCV, this entity can access all or only a subset of the data provided by the applicant. For instance, if we think about company internal employee movements, the HR manager of that company already has access to a set of personal information on the employees, while these data are not accessible to externals.
- The claim issuer is able to provide evidence for all or some of the claimed experience and knowledge the user lists in his eCV.
- The entity which collects, stores, manages and publishes the eCVs will be called the job portal. The job portal also has the ability to store job searches and offers from HR managers and headhunters. Using this information, the job portal is able to perform internal searches on the candidates and find a possible matching. The job portal is the actual service provider entity in the architecture. Any entity that participates, whether applicant or headhunter, has to register to the job portal in order to being able to participate.

The following sections give a general overview of typical process flows for the job portal scenario. We focus on selected functions to demonstrate basic operations and relations between the stakeholders. The diagrams show functions and their associated data which is needed to perform the given operation. While being as general as possible, we try to give as much detail as possible. All operations are sequential and ordered, function names are in bold face type, parameters for the functions in italic type. Wherever possible, optional flows are given to give a more detailed understanding of the processes.

The diagrams presented in this section apply in the same way to the Open Platform Solution and the Inhouse Platform solution. The specific needs and differences between the two solutions are discussed in their respective sections in this document.

2.1.1 Authentication

Unless stated otherwise we will assume, that the job portal provides authentication mechanisms for its users, i.e. employees, applicants, HR managers and trainers. Authentication will have to take place prior to further communication such as publishing eCVs or job offers. Mutual authentication between both entities is a desirable option in terms of privacy and security. Authentication steps are necessary for all presented application scenarios and can be implemented using the authentication mechanisms the possible candidate solutions provide.

The goal is not to define a pure authentication scheme but to focus on data flows and their privacy requirements. All candidate solutions shall therefore be able to support authentication although it is left out in the diagrams.

2.1.2 Publishing an eCV

This section describes the general steps which are necessary to publish an eCV to the job portal.

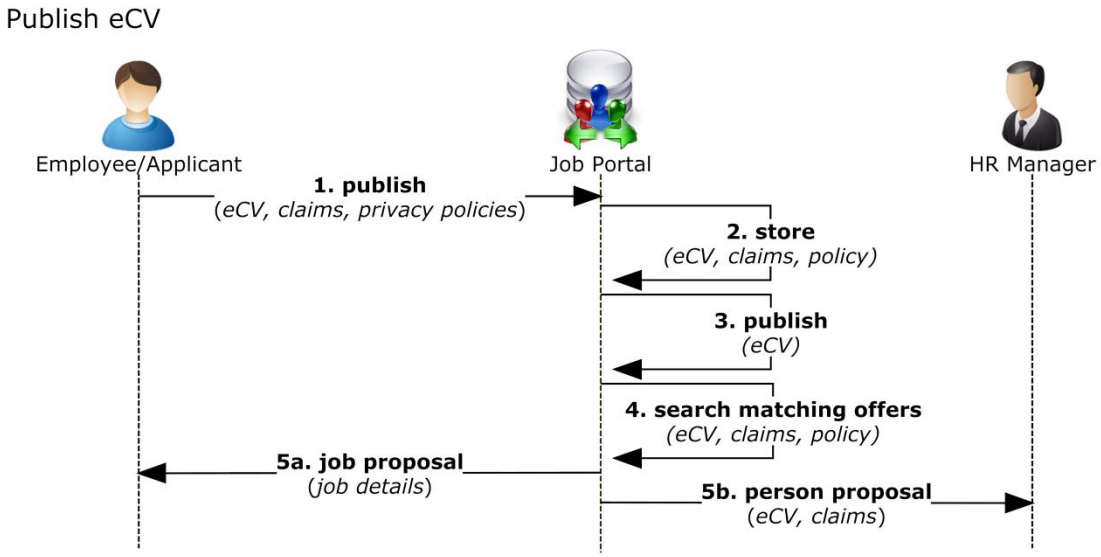

Figure 1: Protocol Flow of eCV publication

1. The employee/applicant publishes his eCV, claims and privacy policies which define the access control to the published data to the job portal.

2. The job portal stores the received data.

3. The job portal then publishes the eCV, allowing HR managers to search for possible candidates matching their criteria.

4. The job portal performs an automatic search for matching job offers in the job offer database depending on the eCV data, claims and the defined privacy policy.

5. Proposal:

    a. The job portal forwards the job details to the employee/applicant.
    b. The job portal forwards the personal information, including the eCV and claims allowed by the privacy policy to the HR manager.

The Steps 5a and 5b can either be performed simultaneously or independently to only one stakeholder.

2.1.3 Publishing a Job Advertisement

The following figure demonstrates how an HR representative can publish a job offer using the job portal (cf. Figure 2).

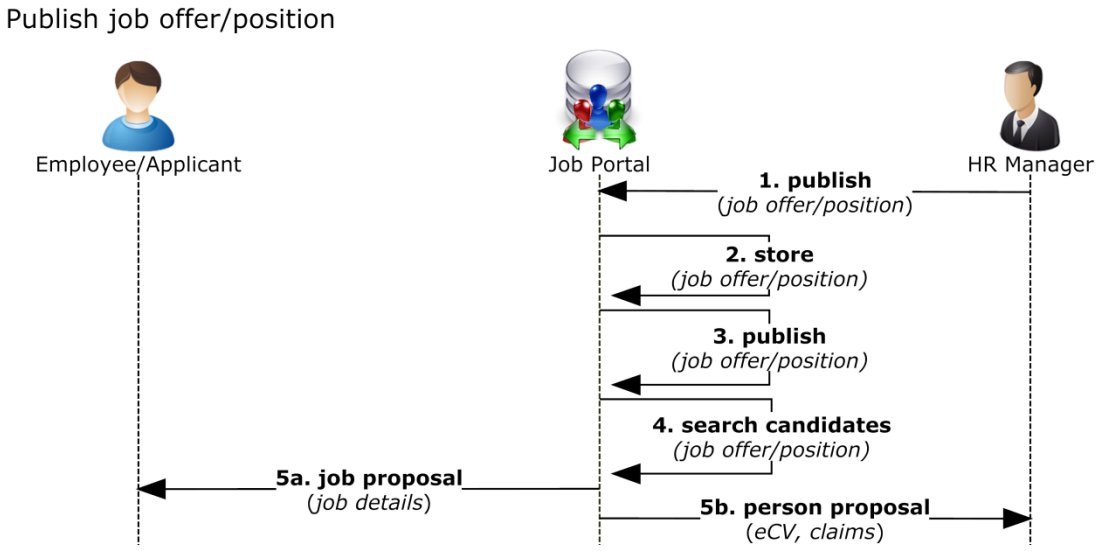

Figure 2: Protocol Flow of job offer publication

1. HR Manager publishes a job or position offer to the job portal
2. The job portal stores the offer
3. The job portal publishes the offer
4. The job portal performs a search for possible candidates in the database of stored eCVs.
5. Proposal:
    a. The job portal forwards the job details to the employee/applicant.
    b. The job portal forwards the personal information, including the eCV and claims allowed by the privacy policy to the HR manager.

It is important to note that the job portal will only publish information to HR managers if this allowed by the privacy policy defined on the eCV. Nevertheless, one can imagine that the job portal performs the search for matching candidates without respect to the privacy policies first and then contacts all possible candidates to get the consent from those whose privacy policy doesn't match yet. This variant allows the candidate to modify the privacy policy as soon as an interesting job offer is presented. Furthermore it will increase the number of matchings that the job portal can provide.

2.1.4 Candidate Matching

If a candidate matches with a job offer (or vice-versa) the job portal will signal this to the two entities as described in steps 5a and 5b from the previous sections. Depending on the option (5a or 5b) two different flows are presented. The flows are shown using the appropriate letter in combination with the step number in the figure (cf. Figure 3). Both activity flows can be combined, i.e. the employee/applicant first retrieves some additional claims and publishes them to the job portal, and then the HR manager requests some more additional claims which are then requested by the employee/applicant. In both cases (a and b), operation resumes with step 8 (answer from employee/applicant to HR manager) and is the same for a and b.

For all flows which are required for the claim management, i.e. requesting a new claim, publishing the claim etc. grey boxes are used. These processes are detailed in the sections on the different

scenarios. We therefore assume that as a result of a claim request operation, the employee/applicant will be supplied with a valid claim from a claim issuer.

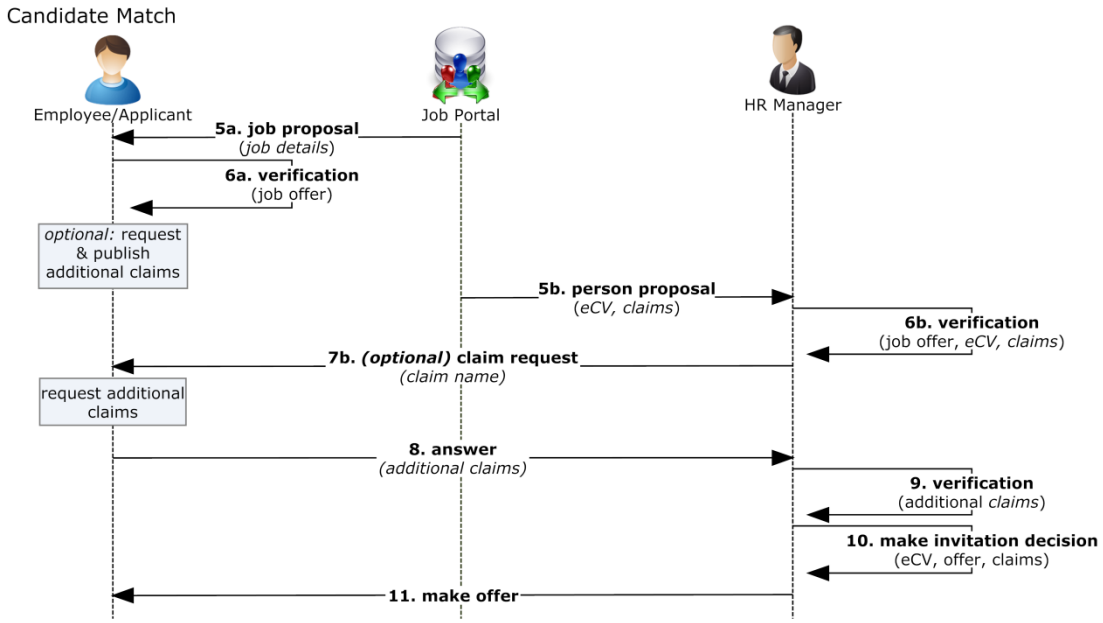

Figure 3: Protocol Flow of candidate matching

Flow A:

5a. The employee/applicant receives the job proposal with job details from the job portal

6a. The employee/applicant verifies the job offer and then decides to publish additional information to his eCV.

Flow B:

5b.The HR manager receives the eCV and some claims from the job portal according to his job offer.

6b. The HR manager then verifies that the proposed candidate indeed matches the given job offer by comparing the eCV and the claims to the offer.

7b. (optional) The HR Manager can then decide to request additional claims from the candidate. This triggers a claim request process.

8. The candidate then answers the request providing the additional claim.

9. HR manager verifies the received claim

10. HR makes an invitation decision based on all received data

11. HR manager offers an invitation to the candidate

2.2 Inhouse Platform Solution

The Inhouse Platform Solution (IPS) is a one-click career portal which is solely used as an intra-organizational infrastructure. It restricts the realm of all actors to the multinational company domain and is intended for national and international career movements within the company. The roles are distributed among employee, HR manager and Inhouse Trainer as document and claim

issuer. The central job portal service is accessible to those three entities and allows for the company-wide management of employee resources. This internally managed and centrally available platform would then be used by employees to provide job requests and request job offers and claims for the CV-portfolio. HR managers can align human resources along different positions in the company according to their CV profiles. Inhouse Trainers can issue claims and provide training offers for users and interact with the job portal.

2.2.1 Actors

The following three roles and tasks can be identified in the IPS scenario (cf. Figure 4)

Employees:

- request job offers and claims for the CV-Portfolio from the HR Manager
- provide job requests and CV information to the HR Manager
- request training offers and training claims for the CV-Portfolio from the Inhouse Trainer
- provide training requests and CV information to the Inhouse Trainer

HR Manager:

- receives job requests and CV information from the Employee
- provides job offers and claims for the CV-Portfolio to the Employee

Inhouse Trainer:

- receives training requests and CV information from the Employee
- provides training offers and claims for the CV-Portfolio to the Employee

2.2.2 Processes

The job portal is the core of the IPS. The job portal must guarantee that only authenticated users can modify the stored eCVs. As such a solution is possibly implemented across national borders, different requirements for user privacy will apply. Therefore it should be possible to implement access restrictions to the data stored by the job portal. Preferably, the user is put into power to allow or restrict access to his data. The Inhouse Trainer should have the ability to approve claims or to store documents such as certificates in the job portal. Such access must be authenticated and restricted. The HR manager will have access to all users' eCVs according to national and international law requirements and restrictions. HR manager will typically have a read-only access.

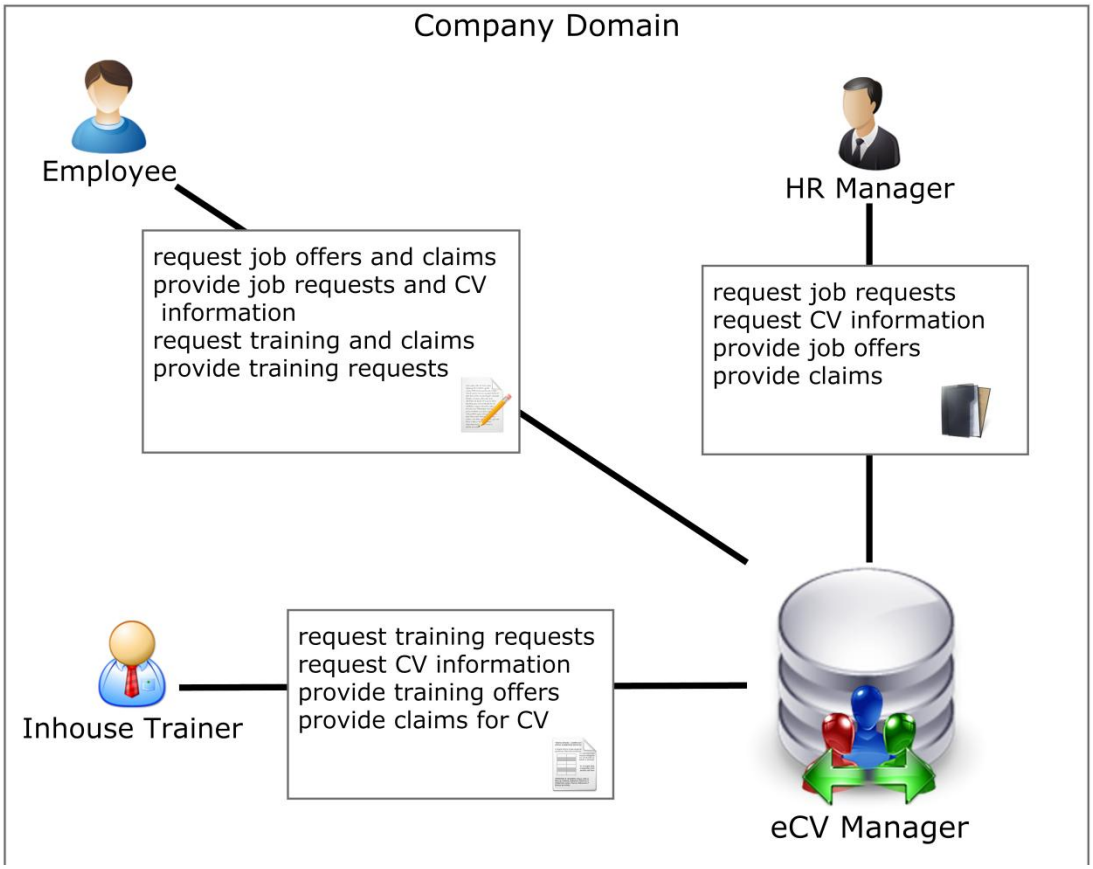

Figure 4: Inhouse Platform Solution Use Case Diagram

If an employee is required to provide an additional claim (or wants to add a claim to his eCV), the following steps would apply (cf. Figure 5):

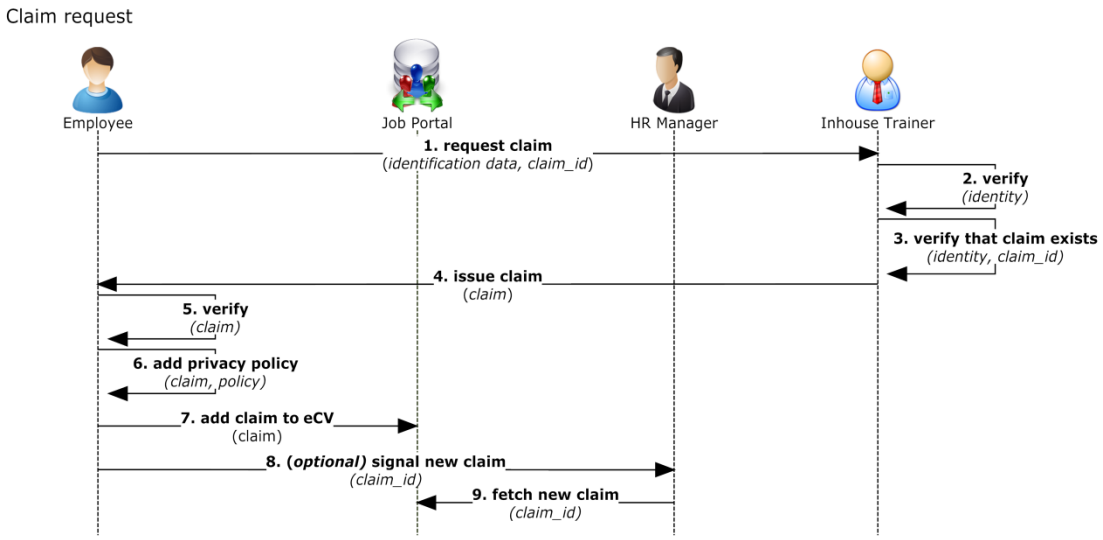

Figure 5: Protocol Flow of claim request

1. The employee requests a claim from the Inhouse Trainer. This request contains an indicator (claim_id), pointing to the desired claim, and authentication information.
2. The Inhouse Trainer verifies the identity of the employee and then verifies that the claim exists, i.e. a training certificate is stored in a local training database.

3. The Inhouse Trainer issues the claim to the employee.
4. The employee would typically inspect and verify the claim upon receipt and then add a privacy policy to it, to restrict access to the claim before publishing it.
5. The employee adds the claim (with the appended privacy policy) to his eCV in the job portal.
6. As an optional step, the employee signals to the HR manager that a new claim is available in the eCV, using the claim_id.
7. The HR Manager will then fetch the new claim from the job portal using the claim_id.

2.3 Open Platform Solution

The Open Platform Solution pursues the classical approach of an online career portal (similar to monster.com and others). A central eCV system is provided for all stakeholders from different domains, used as a multinational and multiorganizational platform. An applicant can access the eCV platform to publish his eCV. For claim issuers it is accessible to provide additional documents and information to the applicant's eCV. Headhunters have the possibility to search for applicants matching a given vacancy. It can be used by a variety of independent organizations and institutions and is intended for national and international career moves across different companies.

The user decides to publish a set of CV and identity related information on a career platform. The access to the career platform is normally restricted to registered users only. The applicant has to authenticate to upload, modify or delete his eCV from the platform. Policies have to be established to guarantee privacy for the user supplied data to the eCV platform. In traditional scenarios, every registered user is able to browse all published eCVs. Depending on the eCV platform, measures can be applied for users to restrict access to their eCV to a closed group of persons. Headhunters can register with the site to select users searching for new job opportunities. Claim issuers will have the possibility to add seconding documents to the eCV of specific users. This scenario allows users to reach a large number of recipients with their published eCV and provides attractive means for headhunters to search for qualified people.

2.3.1 Actors

The following roles can be identified in the Open Platform Solution (cf. Figure 6):

Applicant:

- requests job offers from the Headhunter
- provides job requests and CV information to the Headhunter
- requests claims for the CV-Portfolio from the Claim Issuer
- provides CV information to the Claim Issuer

Headhunter:

- request job requests and CV information from the User
- provide job offers to the User

Claim Issuer:

- request CV information from the User
- provide claims for the CV-Portfolio to the User

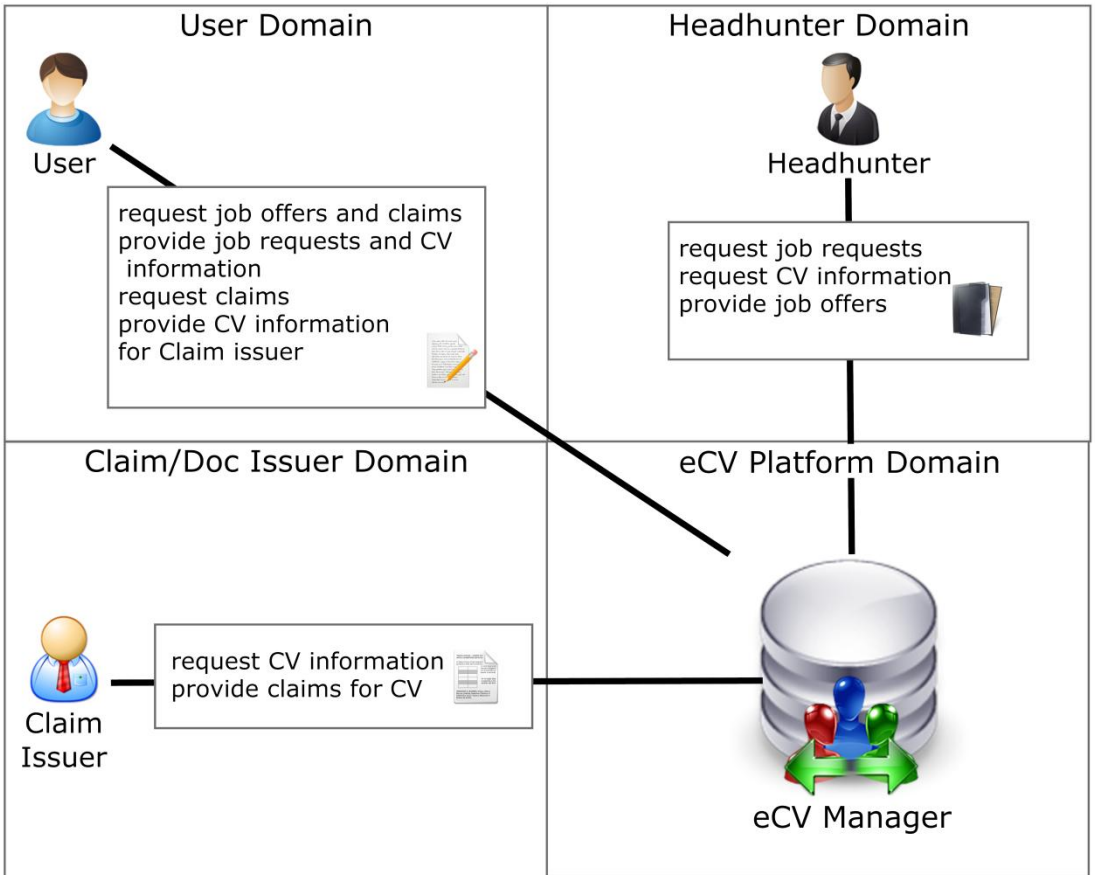

Figure 6: Open Platform Solution Use Case Diagram

2.3.2 Processes

A typical protocol flow in the Open Platform Solution would be like this (cf. Figure 7):

1. A job applicant triggers the publication of his eCV. The dataset also contains claims and a privacy policy.
2. The eCV Manager subsystem of the job portal stores the eCV and the claims and performs the publication of the dataset.
3. Optional: The Job Portal immediately performs an automatic matching between the eCV and job offers already published in the database. If any matching is found, the result is forwarded to the applicant and to the headhunter who published that job offer.
4. A headhunter publishes a job offer and requests the job portal for a suitable candidate for that position.
5. If the job portal finds a candidate, it checks whether the privacy policy of the applicant allows for sending the claims to the headhunter.
6. If the policy allows for that, the eCV and the claims are sent to the headhunter.
7. The headhunter verifies the validity of the claims.
8. If the headhunter is interested in that applicant, he requests the applicant for additional claims.
9. If the applicant is also interested in that position, he requests the respective claim issuer for the new claims.

10. The claim issuer checks whether the requested claim exists. If so, he sends the claim to the applicant.
11. The applicant verifies the received claim.
12. The applicant forwards the claim to the headhunter.
13. Optional: The applicant publishes the new claim in the job portal database.
14. The headhunter checks the claims and decides whether to send the applicant an invitation or a refusal.

2.3.3 Privacy Analysis

Along all its merits, this solution requires a higher level of protection for privacy sensitive data since several independent domains of interest participate in the system. While all trust relationships in the Inhouse Platform Solution reside inside a single trust domain, new problems arise with the separation of different actor domains, leading to more complex trust relationships. All actors must establish a trust relationship with the job portal platform provider. The trust relationship can then be extended by two different trust chains: The job portal platform provider establishes a trust relationship with every single entity from the different domains. Headhunter A trusts the platform provider B who maintains a trust relationship with user C. Hence, with trust transitivity, A can trust C. Another option is to establish direct trust relationships between the different stakeholders. All participants trust the platform provider to behave in a diligent way. All other trust relationships are established on a one-to-one basis, similar to social network concepts – Headhunter A visits User C's profile and has the option to contact him via a built in message system. C then accepts or declines this request, based on information provided by A. He can then add A to his set of trusted page visitors who are allowed to see more details from the profile page. In both scenarios, legal aspects play an important role since such a solution will presumably not be limited to a single country, so different jurisdictions concerning user privacy can apply. A privacy sensitive implementation of this scenario must be able to deal with the complexity of trust relationships.

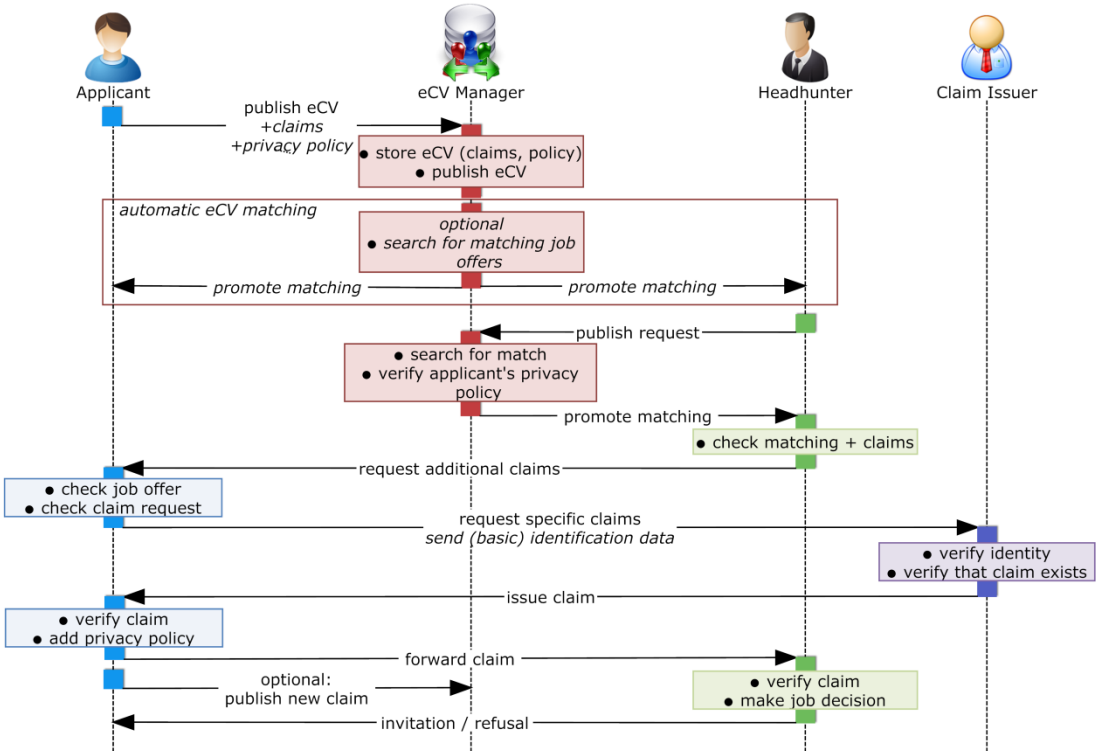

Figure 7: Open Platform Solution Protocol Flows

Chapter 3

Evaluation Criteria and Requirements

Due to their distributed nature, service oriented architectures (SOA) like the proposed job portal scenario raise new privacy challenges. These result from SOAs being composed of different cross-domain or even cross-national legal entities, which are required to store and act upon data which is provided by different stakeholders. PrimeLife H6.3.1 deals with this subject by listing relevant requirements that facilitate the design of privacy-enhancing Service-oriented architectures [PH09]. Based on the previously identified requirements, this section aims to define criteria and categories of criteria for the evaluation of IdM Infrastructure Protocols (IdMIP) that could help implementing a privacy enhanced SOA with support for identity management. While realizing such an architecture, the requirements listed in H6.3.1 should be considered. As some of the criteria relate to specific requirements defined in H6.3.1, the applicable requirement will be listed under the respective criterion. Therefore we follow a more comprehensive approach instead of focusing just on privacy enhancing features. The proposed criteria will cover aspects that are important to be followed, if the evaluated protocols shall be applicable for real-life business architectures. The job portal scenario will be examined exemplary, as such an architecture is a good example for a SOA with privacy and IdM requirements. The different aspects that should be regarded when evaluating IdMIPs will be classified as follows:

- Support of Role Concepts
- Privacy Support
- Manageability
- Usability
- Economic Viability

A multi-factor approach for the evaluation of protocols is essential, if we want a convincing statement about their applicability to real-life scenarios and architectures as a result. Besides technical features, such an evaluation should also consider the economic point of view, as this more often is the spot of failure. A famous example for this is the modest spread of public key infrastructures. From the technical side their added value is undeniable. The reason for the lack of acceptance has to be extracted elsewhere.

In the following sections, the different categories of criteria will be described. Each of the categories consists of several criteria. We use the following simple naming scheme *Criterion X.Y*, where X represents the initial letter of the category and Y is the numbering. The iteration is done inside a single category domain.

3.1 Support of appropriate Role Concepts

In both presented scenarios, the Inhouse Platform and the Open Platform Solution, multiple entities with differing interests are identified. Consequently, the used protocols in the job portal system should consider the potentially divergent requirements of the distinct players. Since there are three subsystems defined for both scenarios, the job portal system should provide mechanisms to enable Role Based Access Control (RBAC) logics. This helps to realize a strict logical separation of the different entities, e.g. by circumventing employees publishing new job offers or headhunters manipulating the eCV of an applicant. We define following evaluation criteria for role concepts:

Criterion R.1 *Strict Role Assignment*

For the job portal we defined several interested parties that will act as a user of the system. Therefore the portal subsystem should support the assignment of roles to registered users. Possible roles are employee, HR Manager and Inhouse Trainer in the Inhouse Solution and Applicant, Headhunter and Claim Issuer in the Open Platform Solution. Role concepts make it easier to assign users access to functions to which they are authorized. In the case of the job portal scenario this will be useful in implementing access control to basic role-based functionalities (e.g. offering jobs as a function for headhunters). Any registered user must be assigned one of these roles. This criterion is a prerequisite for criterion R.2.

Criterion R.2 *Strict Competence Clarification and Compliance*

Preconditioned that Criterion R.1 is fulfilled, the competences (respectively the accessible set of functions) of any role should be well-defined. If C_A is the set of competences assigned to role A, then a member of A should only be able to call functions in C_A. This for example guarantees that new job offers can only be published by HR Managers (or headhunters). Furthermore, this "separation of duties" is essential for accountability. The more trustworthy the RBAC mechanisms are, the more reliable statements can be made about accountability.

Criterion R.3 *Integrity*

There should be mechanisms that prevent the data from being manipulated. A loss of integrity should at least be detected. These mechanisms could help to ensure that CVs are always trusted, that means that all claims and personal information are in an original, unmodified state.

Criterion R.4 *Verifiability*

The job portal system relies on trust relations between the different parties. Thus, one party relies on genuineness and trustworthiness of the data received from another party. The second party should always be able to check the validity of the information received. For example, when an HR Manager receives the CV of an applicant containing claims of different claim issuers, the HR Manager wants to be sure that the claims are not forged. The used protocols should provide mechanisms to validate the source and the content of the received data.

H6.3.1: Requirement No. 2, 5

Criterion R.5 Authentication

Any task executed by a member of the system should require authentication. In addition all operations should be associated with one of the roles.

3.2 Privacy Support

With regard to privacy-related issues a series of challenges has to be faced by the underlying system protocols. Bearing in mind the cross-national or cross-company distribution of the data, e.g. personal data being sent over public networks, especially the Open Platform Solution necessitates adequate protection mechanisms. An applicant wants to prevent his data from being sent to untrusted systems. The used protocols should provide mechanisms to let the user decide to whom he sends which data at what time. The dilemma here can be illustrated by the example of an applicant who wants as much potential employers to inspect his CV, but on the other hand wants to prevent any other party to see his personal data. In order to be able to make a statement about the level of privacy a protocol provides, following privacy-related criteria are identified:

Criteria P.1 *Access Control*

Any holder of personal information of a user should have the possibility to constrain the audience which can examine his personal data. Therefore it should be able to define access control policies. These should enable the user having the last decision of when or under what conditions his data is handed over to another participant.

H6.3.1: Requirement No. 21

Criteria P.2 *Reduced Identity Information*

The users should be given the possibility to control the amount of detail information another entity is provided. He should be able to compose partial identity information individually for any requestor.

Criteria P.3 *Only authorized dissemination of data*

Receivers of personal data from a user should not be able to disseminate these data without the knowledge and permission of the user. For example a company wants a new job offer only to be visible to a limited audience.

H6.3.1: Requirement No. 6, 8, 9, 13

Criteria P.4 *Covering tracks*

It should not be possible to trace or log the history of an employee's applications for jobs. The same holds for job offers issued by HR managers, i.e. the system should provide means to prevent concurrent companies to reconstruct a company profile based on information gained from the collection of job offers.

Criteria P.5 *Confidentiality*

Especially the OP solution requires mechanisms to keep the confidentiality of the data when they are sent over public networks. But also the IP solution requires mechanisms to exchange data confidentially. If for example an employee connects to the database remotely, there should be mechanisms to connect to the company intranet securely. Minimum requirement here is the usage of encryption when exchanging data between the subsystems.

3.3 Manageability

The job portal system enables the cross-domain exchange of confidential data. As the amount of data to be managed can increase, the central management can get a more complex issue. This is not limited to the central job portal subsystem. Furthermore, in the sense of privacy, personal data

should not be managed centrally. As a consequence, more parties have to be involved to the management of the data (e.g. job portal, user himself, storage provider, etc.) For the sake of setting limits to the complexity of the management of the distributed data, several requirements have to be met. Meeting those requirements is an important enabler for the scenario, because these are the factors which have an impact on the adoption of the protocols and with it of the whole architecture.

Criterion M.1 *Minimal competence*

Manageability requires the minimization of the competences of any subsystem. Any entity should only be provided a minimal set of competences in terms of functionalities that are accessible by the specific entity.

Criteria M.2 *Interoperability*

The used protocols should provide simple ways of exchanging data between the subsystems. Any additional overhead that would result from solving interoperability issues shoul be minimized.

H6.3.1: Requirement 1, 19

Criteria M.3 Data minimisation

As one important principle in privacy-awareness and security, data minimisation should be enforced by the job portal system. That means that any subsystem should only store data that is needed to fulfil its duties. As an example, the CVs of a user should only be stored at the users side or in a trusted central storage of the eCV manager subsystem. The aim of data minimisation has a positive influence on the manageability of the system.

H6.3.1: Requirement 7, 23

Criteria M.4 *Scalability*

The used protocols should support scalability of the architecture. The job portal system should be able to operate without noticeable problems. New users and data should be included without disturbing the workflow.

H6.3.1: Requirement 25

3.4 Usability

Any privacy and security enhancing system has to care for a convenient balance between privacy features and usability. Such a system will only be accepted if its usage does not raise new burdens or difficulties for usage. The overall benefit of a privacy enhanced job portal system should exceed the overhead in getting used to work with the system.

Criteria U.1 Easy integration to existing protocols

The used protocols should not necessitate any fundamental adoptions to existent systems. The integration of privacy enhancing features should make use of available systems.

Criteria U.2 Ease of use

The usage and management of privacy and security enhancing protocols and systems should not be complicated. Adding and editing access control policies should be as easy as possible. All actors should be able to use the system without complex tutorials/trainings.

H6.3.1: Requirement 3, 4, 33, 36

3.5 Economic Viability

The job portal system requires the establishment of additional mechanisms and protocols for the establishment of trust relationships. This overhead can only be acquiesced, if there is a benefit for the participants. For the employee/applicant this can be to get a financially more attractive position; the headhunter expects to find the best experts for a specific position. But also the provider of the job portal has some economic expectations. It's in the interest of the service provider that the privacy-enhancing adaptations do not cause high costs, but still enable a high service quality. Meeting the following criteria will help to implement a viable project.

Criteria E.1 *Prefer low-cost protocols*

The used protocols and mechanisms should not cause high costs that exceed the benefits of the system. Therefore it's important to make use of existing and established protocols. When introducing new mechanisms, open and free standards should be preferable. Changes to existing architectures should be held at a minimum and preferably not be associated with high costs.

Chapter 4

Candidate Solutions for IdMIP

In the previous chapters we described the overall architecture of an IdM- and privacy-enabled job portal scenario. We also listed the functionalities which such a system should provide. The crux of the matter here is to implement such a system with enhanced privacy. Therefore, we listed several criteria in chapter 3 which ideally should be met, in order realize this. In this chapter we want to introduce several privacy or IdM- related protocols which can be regarded as potential utilities for achieving the aim of a privacy enhanced job portal system. This chapter will clearly focus on technical descriptions and capabilities of the respective protocols. A mapping of the properties of the protocols to the job portal functionalities will be done in Chapter 5.

4.1 Anonymous Credential Systems

Anonymous credentials are a cryptographic technique that goes back to an idea by David Chaum [Chaum85]. The basic idea is that the user gets credentials that prove statements about him without disclosing any other information than the proved statement. This is applied to allow anonymous presentation of facts. Two such presentation tokens cannot be linked to the same identity. The user Alice can prove to Bob that she possesses a token asserting a property, but Bob cannot deduce Alice's identity, nor can he determine if two tokens proving the same fact belong to the same user or not. Traditional credentials are issued to the user by the credential authority (CA), which acts as trusted third party. Later the user presents the digital tokens to a verifier, usually the policy decision point of some service. The cryptographic token itself consists of a public key and a number of attributes. Both together are signed with the private key of the CA, together forming the credential. It allows the verifier to validate authenticity and integrity of the token with the public key of the CA. The user holds the private key corresponding to the public key in the token. Hence, the user can prove that a presented token really belongs to her. In order to make the tokens untraceable, anonymous credentials use a technique known as *restrictive blinding*.

Stefan Brands extended Chaum's work by basing his scheme on secret-key certificates. Brands developed a system called U-Prove that implements his ideas [Brands00]. Jan Camenisch and Anna Lysyanskaya introduced the idea to build anonymous credentials on group signature

schemes [CL01]. An efficient implementation of group signature based anonymous credentials is called Idemix[3].

This deliverable will not compare protocols like U-Prove or Idemix, but instead will generalize the use of anonymous credentials in the described eCV scenario.

4.1.1 Description[4]

Anonymous credentials – or minimal disclosure tokens – are ideal for sharing identity-related information across organisations. They shall fulfil the following requirements.

- **User-centric:** Using minimal disclosure tokens, organisations can securely share information via the individuals to whom it pertains or via other intermediating parties (such as brokers and outsourcing suppliers). The multi-party security features of minimal disclosure tokens prevent any unauthorised manipulations of protected information, not only by outsiders but also by intermediating parties. For instance, issuers can protect identity claims against unauthorised lending, pooling, cloning, discarding, and reuse by encoding them into minimal disclosure tokens. At the same time, intermediating parties can see the information that is shared, enabling them to boycott inappropriate exchanges. They can also be actively involved in the flow of protected information, helping to determine how organisations conduct data exchanges. Furthermore, they can store protected information upon issuance so that it can be ported and used off-line.
- **Selective disclosure:** Identity information encoded into minimal disclosure tokens can be selectively disclosed in a fine-grained manner. By way of example, the user of a minimal disclosure token stating that its holder is a Dutch citizen born on August 12, 1966 can present the token in a manner that reveals only that the holder is over 18 and European.[5] As another example, a token that specifies its holder's real name can be presented in a manner that proves that the name is not contained on a blacklist of suspected terrorists, without revealing anything else.
- **Unlinkability:** Minimal disclosure tokens can be issued and presented without creating unwanted linkages. This enables organisations to issue authentication tokens to identified individuals that can subsequently be used to access protected resources anonymously or pseudonymously. It also enables account holders to retrieve and present protected identity claims without thereby enabling organisations to link the source and destination accounts. This protects against unwanted profiling across spheres of activity and minimises the risk of identity theft by insiders and hackers. At the same time, individuals who abuse services can be excluded from further participation via several revocation methods that do not contravene the privacy features of minimal disclosure tokens.
- **Non-transferability:** Issuers can prevent users from transferring (copies of) minimal disclosure tokens that convey privileges, entitlements, and other personal credential

[3] http://www.zurich.ibm.com/security/idemix/

[4] This section was taken with kind permission by the authors from: Brands, S.; Bussard, L.; Claessens, J.; Geuer-Pollmann, C., Pinsdorf, U.: Credential Systems, section 4.2.5 in Rannenberg, K.; Royer, D., Deuker, A. (ed.): The Future of Identity in the Information Society, Springer, 2009, 154-158.

[5] Technically, the "over-18" property is proved by providing a zero-knowledge proof that the value (e.g., total number of days or minutes) representing today's date minus the token value representing the birth date is greater than the value that represents 18 years. The "is-European" property is proved by demonstrating in zero-knowledge that the country code encoded in the token is in the set of country codes representing all European countries.

information. One solution is to encode private information of the designated token holder into the tokens; the token holder can hide this data at presentation time (owing to the selective disclosure feature), but would need to reveal it in order to enable others to present the tokens. For stronger protection, issuers can electronically bind minimal disclosure tokens to a previously issued trusted module (such as a tamper-resistant smart card or a Trusted Computing chip) that can enforce security policies (such as non-transferability) throughout the life cycle of the tokens; in contrast to PKI certificates, a single low-cost module can protect arbitrarily many minimal disclosure tokens.

- **Private audit trails:** Relying organisations can capture signed transcripts that prove their interactions with individuals. Prior to storing or forwarding signed transcripts, some or all of their disclosed contents can be censored without destroying their verifiability. This enables organisations to protect their own privacy and autonomy interests vis-à-vis auditors.

The privacy features of minimal disclosure tokens hold unconditionally, in the strongest possible sense: issuing and relying organisations cannot learn anything beyond what users choose to disclose when presenting their tokens, even if they collude and have unlimited resources to analyse protocol data.

Minimal disclosure tokens are not merely an academic construct: leading industry players are working to productise minimal disclosure token technologies. For example, Microsoft has announced plans to implement its U-Prove technology (see http://www.credentica.com) into Windows Communication Foundation and Windows CardSpace, and IBM has developed a similar technology (see http://www.zurich.ibm.com/security/idemix) that has been contributed to open source.

4.1.2 Protocol Flow

Anonymous credential protocols usually consist of two protocols. The issuing protocol allows the user to get credentials from the issuer. The presentation protocol is invoked whenever the user proves something to the verifier. Above that protocols exist that allow revocation of credentials; the revocation can be either user-driven or issuer-driven.

4.1.2.1 Communication Model

In the typical “wine-shop” use case, the User is the customer, the Verifier is the merchant, and the Issuer is the government, or any other instance that is trusted to issue age certificates. The User engages in an Issue protocol with an Issuer to obtain a valid credential on a certain set of attributes. The credential is valid under the Issuer’s public key pk, of which only the Issuer knows the corresponding secret key sk. In its simplest form, the Issue protocol is a two-round interaction where the User submits a request containing her attributes, and the Issuer certifies the fact that the User has the claimed attributes by returning the credential; more generally, the credentials can be generated through a multi-round interaction. The User convinces a verifier that she has a certain set of attributes by engaging in a Show protocol. Again, this can be a simple two-round protocol or a more complex interaction (cf. Figure 8).

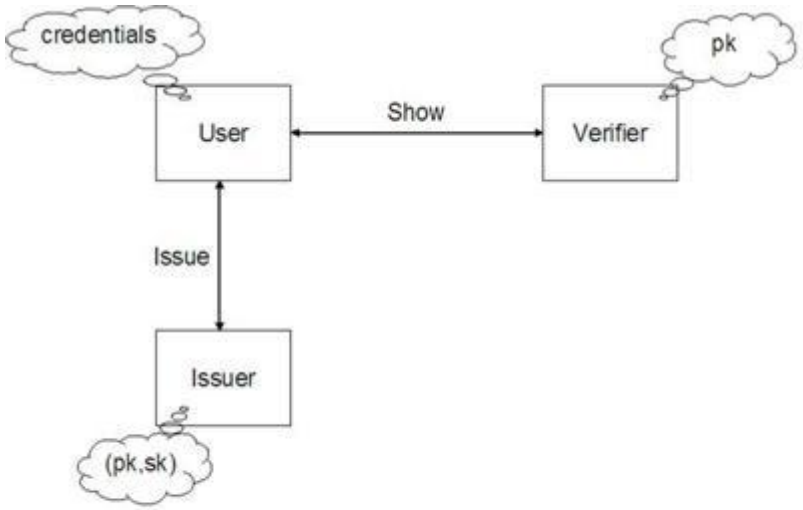

Figure 8: Anonymous Credentials Communication Model

4.1.2.2 Anonymous credentials and selective showing of attributes

Intuitively, an anonymous credential can be thought of as a digital signature by the Issuer on a list of attribute-value pairs, e.g. the list
(fname="Alice", lname="Anderson", bdate="1977/05/10", nation="DE").
The most straightforward way for the User to convince a Verifier of her list of attributes would be to simply transmit her credential to the Verifier. This approach has a number of disadvantages, most notably

- that the User has to reveal all of her attributes so that the Verifier can check the signature;
- that the Verifier can reuse the credential to impersonate Alice with respect to other Verifiers.

With anonymous credentials, the User never transmits the credential itself, but rather uses it to convince the Verifier that her attributes satisfy certain properties – without leaking anything about the credential other than the shown properties. This has the obvious advantage that the Verifier can no longer reuse the credential to impersonate Alice. Another advantage is that anonymous credentials allow the User to reveal a selected subset of her attributes. In the example above, Alice could for example reveal her first name and last name, but keep her birth date hidden. Stronger even, apart from showing the exact value of an attribute, the User can even convince the Verifier that some complex predicate over the attributes holds. In particular, she can demonstrate any predicate consisting of the following operations:

- Integer arithmetic: att+c , att+att , att•c : sum of attributes and/or constants, product of attributes with constants
- Comparative: exp1 = exp2 , exp1 < exp2 , exp1 > exp2 , exp1 ≤ exp2 , exp1 ≥ exp2 : compare arithmetic expressions over attributes/constants
- Logical: pred1 AND pred2 , pred1 OR pred2

Note that revealing a subset of attributes comes down to the special of showing that the predicate att1 = val1 AND att2 = val2 AND … holds. In the example above, Alice could show that she's an overage national of an EU country by showing that he has a credential satisfying bdate ≤ today – 18y AND (nation="DE" OR nation="FR" OR …) . The Show protocol does not leak any information other than the truth of the statement. So in the example above, not only does Alice's

name remain hidden from the Verifier, but so do her exact birth date and nationality; the only information leaked is that her birth date is more than 18 years ago, and that her nationality is one of the EU countries. The Show protocol also allows to prove statements about multiple credentials. For example, when Alice's passport and credit card are anonymous credentials issued by her government and her bank, respectively, then she can show to a Verifier that she has a valid passport certifying that she is over 18 years of age and a valid credit card issued to the same name: pass.bdate ≤ today – 18y AND pass.fname = ccard.fname AND pass.lname = ccard.lname.

4.1.2.3 Verifiable Encryption

There are situations where multiple parties are involved in the same transaction, and each of these parties need different information from the User. For example, in the wine shop use case, the merchant needs to know whether Alice is overage, and the shipping company needs to know her address. The merchant should not learn her address, but wants to be sure that she gives her real address to the shipping company, and not that of an underage friend. Another example is that the merchant may require Alice to submit her real identity to a trusted third party (TTP), so that in case of dispute or fraud the TTP can be contacted to reveal Alice's identity. The merchant wants to be sure that Alice submitted her real identity to the TTP, and not some bogus identity. The TTP in this case plays the role of an identity escrow agent. More generally, there are situations where the Verifier wants to be sure that the User submitted a certain piece of personal data to an External Recipient, but the content of these data should remain hidden from the Verifier. A "low-tech" solution to this problem is to have the User send this data directly to the External Recipient, who either returns a receipt to the User that the User can show to the Verifier, or who contacts the Verifier directly to acknowledge that it received the correct data. This situation is depicted in the left picture below.

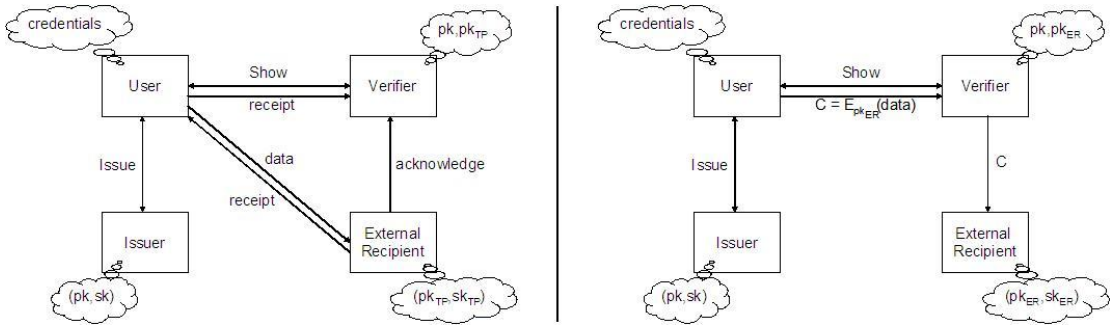

Figure 9: Anonymous Credentials Communication Model

A more "high-tech" setting is depicted on the right hand side of Figure 9. Here, the User sends the required data to the Verifier, but encrypted under the public key of the External Recipient. When she uses a verifiable encryption scheme, the User can prove predicates about the content of the encrypted data of the same form as those in the Show protocol to the Verifier, to assure the Verifier that the ciphertext C actually contains the correct data. (Without such a proof, the User could easily encrypt bogus data, since the Verifier does not know the decryption key sk_{ER} anyway.) The External Recipient does not need to be online at the time of the transaction; rather, the Verifier can forward the ciphertext only when it is actually needed, e.g. when the item is being shipped, or when a dispute actually occurs.

4.1.2.4 Limited Spending

4.1.2.4.1 One-time spending

Because of the anonymity of anonymous credentials, a Verifier cannot tell whether two successful logins were initiated by the same user or by different users. This is good for privacy, but becomes a problem in situations where one wants to place a bound on the number of times that a single credential can be used. We first sketch a simple solution for the special case where a credential can only be used (or "spent") once, and then present a more advanced solution that works for arbitrary bounds.

When a credential is only to be used once (a so-called "one-time credential"), the issuer includes an additional attribute `credentialID` in the credential that is set to a random value. To log into the system, the User has to reveal the value of `credentialID` to the Verifier. The Verifier simply maintains the list of `credentialID`s that it has seen, and denies access when the revealed `credentialID` already occurs in the list.

This solution works fine as long as there is only a single Verifier, or as long as all Verifiers have access to a common database that keeps track of spent `credentialID`s. It is impossible to prevent a User from overspending a one-time credential by using it at multiple Verifiers that are not in constant communication with each other, but there are ways to detect such behavior after the fact. Namely, as part of the showing protocol, the Verifier can choose a random challenge *chall*, which the User has to respond to with a value resp that computed as a function of *chall* and some attribute that uniquely identifies the User, e.g. name, together with a proof that resp was correctly computed. The protocol is such that a single challenge-response pair (*chall*, *resp*) does not reveal anything about name, but name is easily computable from two pairs (*chall*, *resp*) and (*chall'*, *resp'*) whenever $chall \neq chall'$ (which happens with overwhelming probability if the challenges are long enough). By storing triples of the form (`credentialID`, *chall*, *resp*) for each login, Verifiers can later on compare their lists to detect overspent credentials and reveal the identity of fraudulent users.

4.1.2.4.2 Multi-time spending

The above technique is easily extended to allow spending up to *n* times as follows: include *n* different random attributes `credentialID1`,...,`credentialIDn` in the credential, and let the User reveal one of them each time she logs in. This quickly becomes inefficient however, as the number of attributes in the credential grows linearly with the spending bound. Also, the Issuer has to fix a single bound once and for all; it would be nice if different Verifiers could impose their own bounds on-the-fly.

A more powerful approach is the following. As an extra attribute, each credential includes a random hash key `HK` for a keyed hash function H. The hash function is such that it allows the User to prove statements of the form

$$H_{HK}(somestring) = somevalue$$

as part of the showing protocol, whereby the hash input and output are revealed, but `HK` remains hidden. When the User logs in for the *i*-th time, she reveals $H_{HK}(i)$ and proves that the value was correctly computed from `HK` as included in her credential. The Verifier checks that $1 \leq i \leq n$ and that the hash value does not yet occur in its database.

Since the hash function takes any bit string as input, different web sites can easily impose their own spending limits by collecting hash values of the form

$$H_{HK}(websiteID, i)$$

from users. The system is also easily extended to interactive challenge-response protocols that allow after-the-fact detection of abuse.

4.1.2.5 Cryptographic Pseudonyms

In some situations it may desirable to create permanent pseudonymous user accounts. This could be for efficiency reasons, because the user does not want to re-prove that it satisfies all access requirements at each login, or for functionality reasons, for example to store the user's preferences or a list of previous transactions across several sessions. The actions that a user performs under the same pseudonym thereby unavoidably become linkable, but otherwise no new information should be leaked to the server. In particular, it should be impossible to link different pseudonyms of the same user together, whether it be at the same server or across different servers.
So far, the requirements can be satisfied simply by letting each user set up a random username-password pair. The problem with this solution is that the user can set up an account, prove that she satisfies some requirements, and then sell her password to users who do not satisfy these requirements. Since there is no connection between the user and the pseudonym, the user has no incentive not to share her credentials with others.
Anonymous credentials provide a better solution. Namely, each of the user's credentials contains a *master secret* as a special attribute. Each user has her own master secret, and the same master secret is present in all of a user's credentials. The issuing of credentials is done in a special way so that the same master secret must be used in all of the user's credentials, without revealing its value to the issuer however. The idea is that the master secret is the one secret that the user holds very dearly, as it is essentially the cornerstone of her entire electronic identity.
The user's "pseudonym" is not a self-chosen name or a random bit string, but rather a special cryptographic token that is non-deterministically derived from the master secret. (Meaning, you can generate many different pseudonyms from the same master secret.) The corresponding "password" is the master secret itself. Rather than simply sending the master secret to the server to log in, the user proves in a cryptographic way that she knows the master secret underlying her pseudonym, but without revealing its value.
The cryptographic pseudonyms thus derived have the property that (1) no one can tell whether two pseudonyms were derived from the same master secret or not, and (2) it is impossible to successfully log in under a pseudonym without knowing the underlying master secret.
If the user now wants to share her account at a server with other users, then she has to give away her master secret, which immediately gives access to all other servers where she ever registered. It is hoped that this all-or-nothing situation (either you share nothing, or you share your entire identity) forms enough of a disincentive to prevent users from sharing their credentials.

4.2 Liberty Alliance

This section provides an overview of the Liberty Alliance (LA) and the defined IdM frameworks. The core concepts of IdM using Liberty Alliance are outlined according to the specifications.

The Liberty Alliance is a business alliance which was formed in September 2001 under the lead of SUN Microsystems and joined by at least 30 organizations to establish an open standard for federated IdM. Liberty Alliance now includes more than 160 member companies. The key aspects addressed in Liberty Alliance are security, confidentiality and interoperability. The goal is to enable users to authenticate and sign-on to one network entity first and then allow them to visit other services without the need for re-authentication. By providing an open standard and business guidelines for federated IdM, spanning all network devices, Liberty IdM allows users to maintain personal information more securely. The decentralized authentication and open authorization are part of the approach for an open and secure SSO system [TG09].

4.2.1 Description

One of the aims of the Liberty Alliance is to provide frameworks and infrastructures that help to realize efficient network identity management for businesses and consumers. Therefore, the Liberty Alliance has specified several architectural frameworks. This section gives an overview of the key concepts and terminologies defined by the Liberty Alliance. One major characteristic of the Liberty Alliance architecture is its being composed of several frameworks which are partly based on each other. Three frameworks were published in several phases. In **phase 1** the Identity Federation Framework (ID-FF) was published [LA05]. This framework describes basic identity enabling protocols and principles that help to create more complex identity management infrastructures. ID-FF e.g. contains descriptions of SSO protocols and identity linkage mechanisms. The identity web services framework (ID-WSF) was published in **phase 2** [LA07]. ID-WSF describes mechanisms to provide interoperable identity services and services for describing and discovering identity services which shall support the development of identity-based services, both for service provider and consumer. ID-WSF contains descriptions for [LA08]:

- **Discovery Service**: Through the Discovery Service other Service Providers can identify the (previously registered) IdP of a user which holds specific parts of the identity information. This can be exemplified as follows: An online music streaming service Provider wants to know where to get information about the taste of music of a user. The Discovery Service then responses with a handle to a social network service (SNS), where the user previously had entered his favourite bands. Getting the actual information from the SNS should require authorization from the user, e.g. by using the Interaction Service (see below).
- **Identity Mapping Service (IMS):** This web service allows to map user identifiers between different provider namespaces. Identity mapping may be required when providers use pairwise pseudonyms to refer to users, a technique often used for its privacy advantages.
- **Interaction Service (IS)**: This service defines mechanisms that enable a Web Service Provider to get admission from the owner of a resource to access the resource.
- **People Services (PS):** These services allow the management of person-to-person (p2p) relationships. Via PS it is possible to obtain information of the contacts of a user, e.g. the information about the IdP that holds information of a specific person. The basic principle is comparable to a SNS, which also reflects the p2p relationships of a user.

The introduced web services are essential for realizing privacy-respecting cross-SP architectures with identity federation. The term of identity federation and other key concepts of the Liberty Alliance initiative can be described as follows [LT03]:

- **Single-Sign-On (Simplified Sign-on)**:
  Describes mechanisms that enable a client the usage of several services by logging in into only one Liberty enabled site. Access to the other services does not require any additional authentication.
- **Single Logout**:
  Provides synchronized session logout functionality across all sessions that were authenticated by a particular identity provider.
- **Federated (Network) Identity**:

Identity federation is the way how a user can use single sign-on without the need to store all identity information centrally. Through federation of pieces of ID information through several identity providers, the user has the control over how his personal information is used by service providers. The group of service providers that share linked identities and have business agreements in place is known as a *circle of trust*. The attribute sharing policies base on agreements between the service providers, notification to the user of information being collected, user granting consent for types of information collected.

- **Circle of Trust (COT)**:
  A circle of trust is a federation of service providers and identity providers that have business relationships based on Liberty architecture and operational agreements and with whom users can transact business in a secure and apparently seamless environment.

The following lists the actors/roles which participate in Liberty enabled architectures [LA08]:

- **Principal**: An entity that can acquire a federated identity, that is capable of making decisions, and to which authenticated actions are done on its behalf.
- **Identity Provider (IdP):** Liberty-enabled entity that creates, maintains, and manages identity information for Principals and provides Principal authentication to other service providers within a circle of trust.
- **Service Provider (SP):** An entity that provides services and/or goods to Principals.
- An **assertion** is a piece of data produced by a SAML authority regarding an act of authentication performed on a Principal, attribute information about the Principal, or authorization permissions applying to the Principal with respect to a specified resource.
- **Web Service Consumer (WSC):** A system entity that is requesting some information or action from a web service.
- **Web Service Provider (WSP):** A system entity that provides a web service.
- **SAML** is an XML standard for exchanging authentication and authorization data between security systems.

4.2.2 Protocol Flow

This section gives an overview of the basic protocols of Liberty Alliance specifications that enable identity federation and management.

4.2.2.1 Identity Federation

The main concept of architectures like Liberty Alliance is the possibility for a user to link together pieces of identity information that are distributed over several identity providers. The benefit for the user is that he/she does not need to store personal data at every identity or service provider redundantly. This concept of ‘identity federation’ builds the basis for most of the protocols described by the LA. Therefore, this section shall give an overview of the federation protocol.

In order to enable identity federation, the Liberty Alliance envisages that identity and service providers who want to link identity information of a user, have to make business agreements on how the data shall be treated and shared. Service providers and one or more identity provider together build a “circle of trust” (COT). Inside this trust relationship, identity information of a user is exchanged, based on policies which have to be defined at the stage of building the COT.

In an example from the users’ perspective, identity federation proceeds as follows [LA05]:

1. The user logs in to the portal of Airline.inc by providing username and password. The website of Airline.inc is Liberty enabled.
2. Airline.inc notifies the user that he/she can federate his local identity (stored at Airline.inc web server) with partner websites. Therefore, Airline.inc asks the user for permission to introduce his identity to the partners of the Affinity Group. (Note: Introduction is not the same as federation, but it can be seen as a preparing step for the actual federation)
3. As the user is looking for a maximum level of convenience, he/she gives his permission to the introduction.
4. A couple of hours later the user visits the website of CarRental.inc which is also a member of the Affinity Group.
5. The user logs in to CarRental.inc
6. As the user gave his permission to Airline.inc to introduce his identity to members of the Affinity Group, CarRental.inc recognizes him/her and asks if his/her identity shall be federated with Airline.inc.
7. The user approves the federation.
8. Now the user is logged in to CarRental.inc and he/she can access its services as usual. In addition the user has now more options as his local account is federated with his/her account at Airline.inc.

For this procedure one important requirement is asking the user for permission before any federation is taking place. On the other hand the members of the Affinity Group which form a COT have to reach agreements on the policies of how to handle with personal information of a user.

4.2.2.2 Simplified Sign-On (Single Sign-On)

The SSO protocol of Liberty Alliance is shown in Figure 10. The general setting is a user who wants to access a web service (WS). For authentication either the user will be redirected to a Liberty enabled IdP (if the user first visits the web service) or he directly performs an authentication request at the IdP. In both cases the user will be issued an authentication assertion which contains the necessary ID information about the user. The user then redirects this authentication assertion to the SP which can verify the information listed in that assertion.

Preconditioned that identity federation has been done, the user experience of liberty enabled SSO can be listed as follows:

1. The user logs in to Airline.inc
2. User visits CarRental.inc
3. The login status at Airline.inc is (automatically) forwarded to CarRental.inc. This process is transparent to the user.
   → As no identity information is exchanged between the two service providers during identity federation, a service provider is not able to see any personal information stored at the other service provider.
   ➔ Local policies are still relevant for data processing.

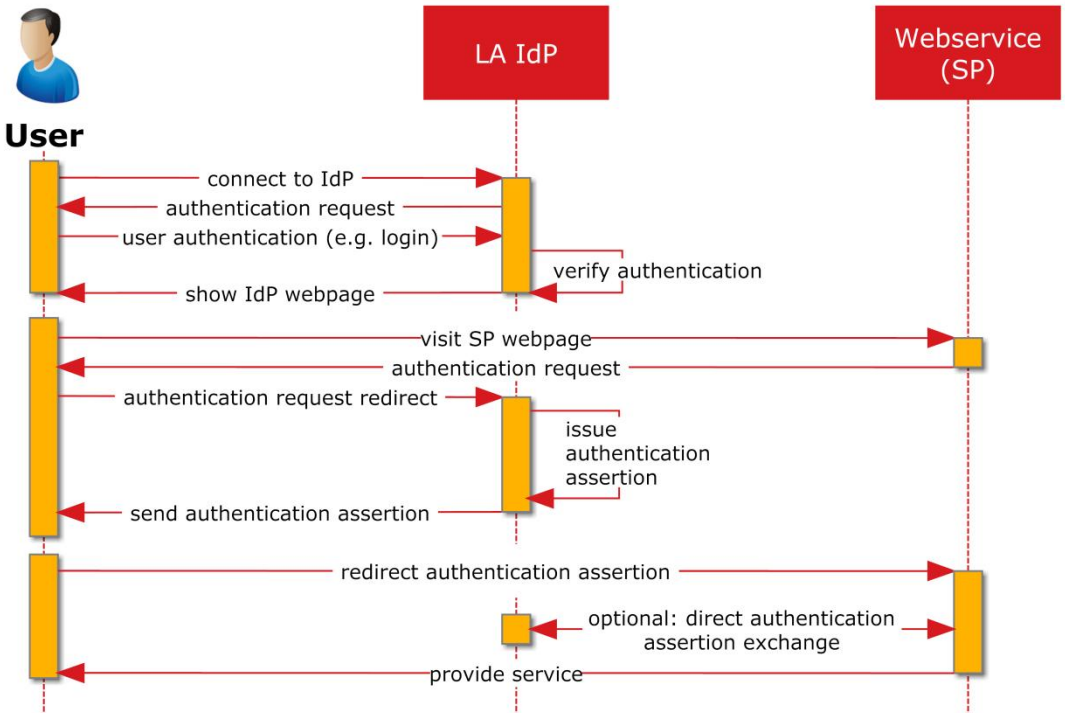

Figure 10: Liberty Alliance simplified sign-on protocol flows

4.2.2.3 Single / Global Logout

The same convenience that a user has by using SSO is offered to the user for the logout. The Liberty Alliance Framework describes services that make a global logout possible. By using the appropriate protocols, the user initiates the logout operation at the identity provider. The IdP then signals to all SP in the same COT in which active sessions are running that the user wishes to logout.

4.2.2.4 SAML Token

SAML (Security Assertion Markup Language) is an XML-based framework that enables the exchange of authentication information in identity management. These pieces of information are contained in so called SAML tokens (cf. Figure 11). These tokens contain claims or assertions about a subject. These include [TG09]:

- Authentication Statements: Testifies that a subject has been authenticated successfully by using a certain protocol
- Attribute Statements: Contains assertions about certain attributes of the user
- Authorization Statements: Declares if the user is given access to specific resources

Furthermore, a SAML token contains information about the issuer of the token, a timestamp, the subject of the assertion, assertions about the subject and conditions under which the assertions are valid.

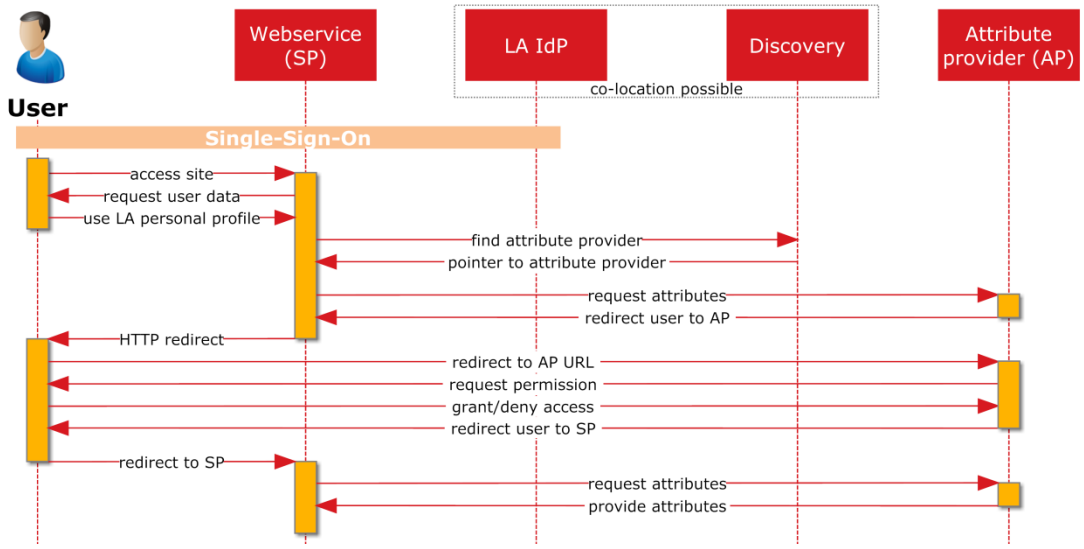

Figure 11: Liberty Alliance Attribute Exchange

4.2.3 Advanced Liberty Alliance Scenario

Alice uploads some photos of her holidays to a photo sharing portal. She wants Bob to have access to these photos but Bob does not want to create an account on the portal. The Liberty Alliance services enable such a cross-principal resource sharing by using the People Services (PS). Requesting Alice's People Service Provider the IdP gets a list of Alice's friends. Alice's PS can be found by requesting a Discovery Service (Disco). If Bob is not a member of this list, the photo service can send a request for adding him to the list to friends.idp.com (Alice's PS). Bob receives an e-mail including an invitation to Alice's list of friends. The invitation also includes the authorization for Bob to be able to access Alice's photos. Furthermore, the possibility to federate one of his accounts with friends.idp.com is presented to Bob (this requires that the account belongs to the same COT as friends.idp.com). If Bob wants to see the photos and accepts the invitation, an ID is generated for him at photos.example.com that can be used by the SP to be able to identify Bob, who is a legitimated person to see the photos [LA07].

4.3 OpenID / OAuth

This section gives a description of the OpenID IdM system and the OAuth access delegation and authorisation scheme. After a general presentation of the two protocols, the approach of a combined OpenID/OAuth protocol is shown.

4.3.1 OpenID

OpenID is an open, decentralized framework for an IdM system, which was developed to provide a single sign on experience for users of distributed services across the Internet. Using OpenID, it is possible to sign on to multiple services, using a single identity, thus eliminating the need to create separate identifiers and logins for the various websites, forums, blogs, etc. OpenID is growing

quickly and is supported by large companies, such as AOL, Facebook, Google, Microsoft, Yahoo, Sun, MySpace, Novell, Telecom Italia, France Telecom, etc. According to the OpenID website[6] (May 2009) there is an estimate of one billion OpenID enabled user accounts and over 40.000 websites which support OpenID logins.

OpenID emerged from the open source community, mainly steered by Brad Fitzpatrick and is as such not owned by anyone. To promote and support the expanded adoption of OpenID, the OpenID Foundation (OIDF) was created in June 2007. In February 2007, AOL already added OpenID support to all of their ~ 63million users worldwide[7]. They became the first large OpenID provider. Yahoo followed and enabled OpenID support for their userbase of around 248 million registered users in January 2008[8]. In late 2008, Google and Microsoft announced that the user accounts from Gmail and Windows LiveID will support OpenID[9] [10].

4.3.1.1 Description

OpenID provides users with a single identity which can be used across a variety of Internet services, without having to remember multiple identifiers and login information. The OpenID identifier is in the form of an URI, e.g. http://myname.myopenid.com/. This identifier is then used to sign in to the favorite website which supports OpenID login. OpenID identifiers are issued by OpenID providers (IdP). As stated above, major companies act as OpenID providers for their customers and provide them with an OpenID identifier. As OpenID is an open standard framework, OpenID accounts are not limited to commercial IdPs and independent IdPs (such as myopenid.com) emerge.

All websites that support OpenID login are referred to as Relying Parties (RP). If an OpenID user wants to sign in, he enters his OpenID identifier in the login box. The RP visits the site from the user supplied URI and extracts the necessary information to reach the IdP for this identity. The IdP and RP establish an association using a shared secret. This association is then used to sign any further communication between IdP and RP. After establishing the association, the RP redirects the user to their chosen IdP login page. The user authenticates at the IdP, e.g. via a login-form. The manner in which authentication between user and IdP takes place is not considered by OpenID specifications. The IdP then redirects the user back to the RP, making an assertion on whether authentication succeeded or not. The user can now use all services provided by the RP, while not having to register with the RP directly.

4.3.1.2 Protocol Flow

The roles of the OpenID protocol are defined as follows:

> **OpenID Provider (IdP):** hosts the identity information for the end-user and allows users to register for an OpenID identifier. The IdP provides a method to authenticate users when they want to sue their OpenID. Further, the IdP provides functions for Relying Parties to discover IdPs and establish associations with them
>
> **Relying Party (RP):** offers a service to end-users. End-users have the ability to use their OpenID identifier to log on to the service. The RP will then associate with the IdP and

---

[6] http://openid.net
[7] http://blogs.zdnet.com/BTL/?p=4513
[8] http://yhoo.client.shareholder.com/releasedetail.cfm?ReleaseID=287698
[9] http://www.golem.de/0810/63249.html
[10] http://openid.net/2008/10/30/microsoft-and-google-announce-openid-support/

then get an assertion on the outcome of user authentication. The RP is not required to maintain a local user management. All authentication tasks are handled by the IdP

User: The user registers with an IdP for an OpenID identifier. This identifier can be used at every OpenID enabled RP. The authentication takes always place towards the chosen IdP.

The general protocol flow of OpenID authentication is as follows (cf. Figure 12):

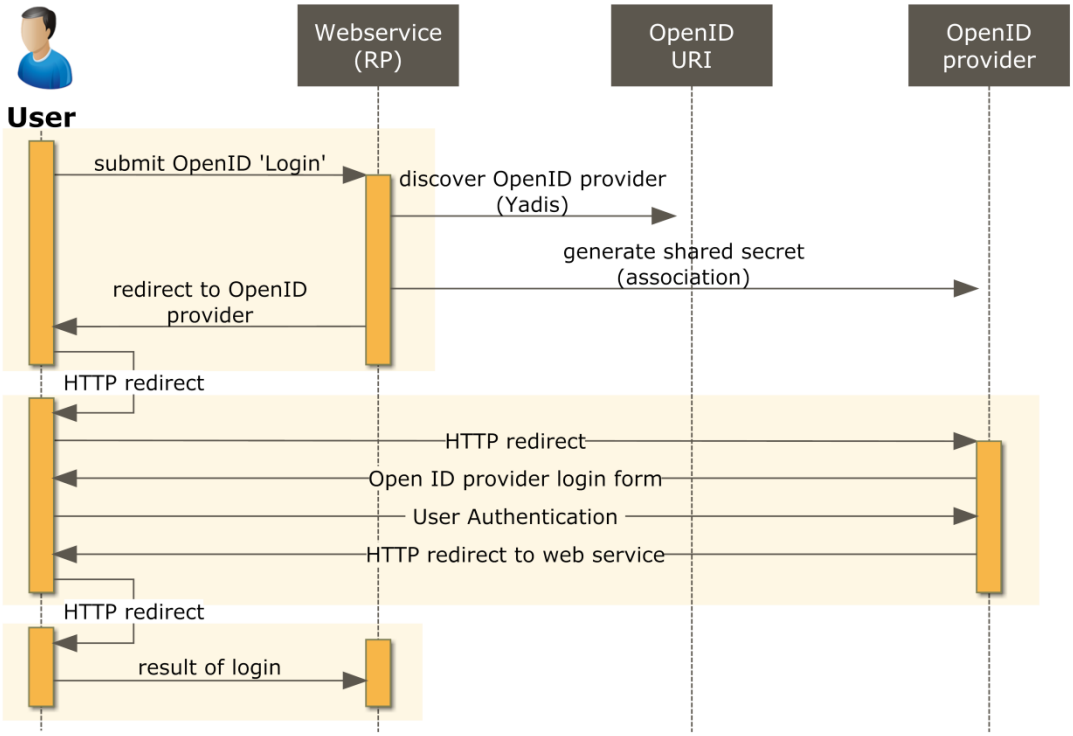

Figure 12: OpenId Protocol Flows

1. The user visits a website from RP and uses his OpenID URI as login
2. RP performs IdP discovery on the OpenID URI
3. RP associates with IdP, establishing a shared secret
4. RP redirects the user to the IdP
5. IdP performs authentication
6. IdP redirects the user to the RP with an assertion on the outcome of authentication

4.3.1.3 OpenID specifications

OpenID specifications are developed by a community of editors, the following specifications are finalized and considered official. There exist further specifications as proposed drafts which are still evolved by the community

4.3.1.3.1 OpenID Authentication 2.0

OpenID Authentication Protocol 2.0 provides a way for an end user to prove that he is in control and possession of an identifier. As a decentralized solution, the protocol allows an end user to freely choose which OpenID provider to use. It is possible to preserve the identifier if OpenID providers are switched. The protocol relies on standard HTTP(S) requests and responses and doesn't require Javascript. It is possible for an end user to prove his identity without leaving the current web page. The re-use of an OpenID identifier for another user (e.g. after a user changed his username or left the OpenID Provider), the so-called 'identifier recycling' is addressed by OpenID 2.0 by either using XRIs which are persistent or by adding so called 'fragments' to the identifier such as myopenid.org/user#A is different from myopenid.org/user#B.

4.3.1.3.2 OpenID Simple Registration Extension

The OpenID Simple Registration Extension to the Authentication Protocol allows for light-weight profile exchange. Its purpose is to centrally store and manage eight commonly requested pieces of information (nickname, email, Full name, date of birth, gender, postcode, country, language, timezone). When an end user registers a new account with a web service, this information can be passed to the web service.

4.3.1.3.3 OpenID Attribute Exchange 1.0

The Attribute Exchange extension to the OpenID Authentication Protocol adds the option to exchange identity attributes between endpoints. Different to the Simple Registration Extension, the set of attributes is not fixed. Options are defined to receive and store attributes from and to the OpenID IdPs.

4.3.2 OAuth

OAuth is an open protocol for access delegation and authorisation. While OAuth is no IdMS, it is closely related to IdM concepts. Blaine Cook started working on OAuth around November 2006, while he was working on the Twitter OpenID implementation. Together with Chris Messina he tried to use OpenID together with the Twitter API to delegate authentication. OAuth allows users to grant access to private data stored on one site to another site. This permission to access the information can be given without having to share the login password or any other secret with the site. While OpenID is designed as an IdMS to provide users with a single identity they can use to sign up for different services, OAuth allows users to share private data with a specific service without exposing the identity at all. The current version of the OAuth specification is 'OAuth Core v1.0' from December 2007.

It is worth to be mentioned, that recently (late April 2009) a security flaw in the OAuth protocol has been discovered. The flaw allows an attacker using social-engineering techniques to trick users exposing their data. The attacker initiates the OAuth authorization and then convinces a victim to follow the link for the authorization request. The victim then continues the attacker's request, including the request token. After approval of the request token by the victim, the attacker can use the saved request token to complete authorization and gain access to the victim's data. The attack relies on a flaw in the OAuth 1.0 protocol, and is not limited to any particular implementation. A modification to the specification will mitigate this attack.

4.3.2.1 Description

The OAuth protocol enables the following scenario: A user wants to print photos stored on another site. Therefore, the user signs into the printing service website and places an order for prints. The user chooses the name of the site where the photos are stored and the printer website sends the user to the photo site to grant access. The user signs into her account at the photo site and is asked if she really wants to share her photos with the printer service website. If she agrees, she is sent back to the printer site which can now access the photos. The user did not share her username and password with the printer site at any time[11].

Three entities can be identified from the above example and are present in the OAuth protocol: User, Service Provider and Consumer. The user has some private data which is stored at a trusted site. In order to access this site and the data, authentication mechanisms are established to restrict access to legitimate users only. The user doesn't want to make his data public but wants to share it with other sites when needed. The site which hosts the data provides a registration and authentication service for its users. This site is called the Service Provider (SP). The SP controls all aspects of the OAuth implementation and denotes the place, where a user's private and restricted resources are located. All applications which are trying to access this data are called Consumers. This includes websites, applications, mobile devices, set-top boxes and the like. The Consumer gets permission from the user to access the user's data, provided by the SP.

OAuth builds its communication on the HTTP protocol. It uses so-called tokens instead of user credentials to access resources. Tokens are unique, random strings which are paired with a secret to protect them from abuse. Two types of tokens exist: request and access tokens. A consumer first requests a request token from SP, which can then be leveraged to access tokens after user's consent at the SP. Using the access token, the consumer has access to the requested data. Tokens can live forever, have a limited lifetime or be restricted to a specific activity. Different types of access control can be realised using the access tokens.

[11] http://oauth.net/core/1.0a#anchor41

4.3.2.2 Protocol Flow

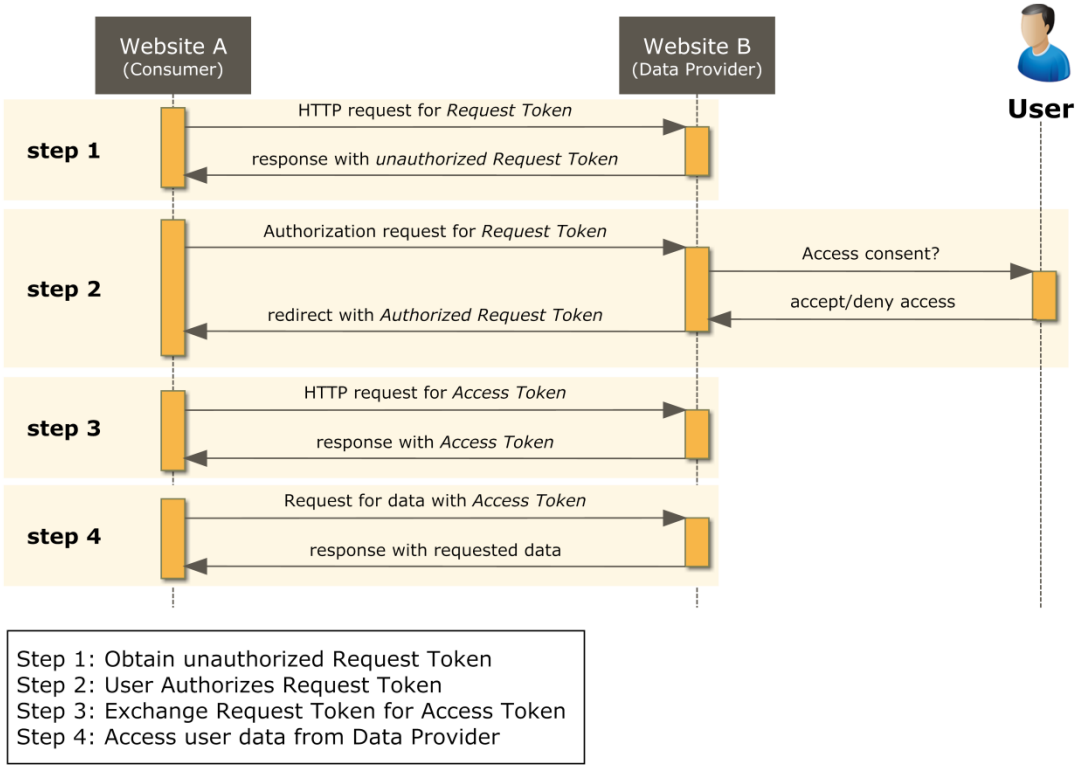

Figure 13: OAuth Protocol Flows

The OAuth protocol can be divided in 4 major steps (cf. Figure 13):

1. Request for request token:
    a. The consumer sends a request for a request token to the data provider. For example a website A which wants to have access to documents stored on another website B.
    b. The data provider then responds with an unauthorized request token. The tokens are used to control associations of different consumers to the data provider and allow for a differentiated access control management.
2. User Authorization for request token
    a. The consumer sends an authorization request for the obtained request token to the data provider
    b. The data provider allows the user to grant or deny access for the consumer to the data stored at the data provider.
    c. If the user accepts the request, the data provider returns an authorized request token to the consumer,.
3. Exchange of request token for an access token
    a. The consumer can then use the authorized request token to obtain an access token from the data provider
    b. The data provider checks if the authorized request token from the consumer is still valid and then sends an access token to the consumer.

4. Access data
    a. The consumer uses the access token to request user data from the data provider
    b. After verification of the access token, the consumer has access to the data.

4.3.3 Combining OpenID and OAuth

In early 2009 Google, Plaxo and Yahoo drafted a proposal for an OpenID OAuth Extension. The goal is to define a process to combine an OpenID authentication request with the approval of an OAuth request token.

The OpenID provider (IdP) acts as OAuth Service Provider and can issue access tokens to the Consumer. The IdP performs user authentication and consent to the OAuth request. The user has a combined authentication and consent page which allows him to securely log into his OpenID account at the IdP and allow the third party web application to access the requested data. By this combined approach an enhanced exchange of profile and user data is possible while putting the user in control of which data can be accessed by which external party.

4.3.3.1 Protocol Flow for combined OpenID/OAuth

When using the combined protocol OpenID/OAuth protocol, the following steps apply (cf. Figure 14):

The user signs in at a web application using his OpenID login. The web application performs discovery on the OpenID URI and requests a login authentication from the IdP found in discovery. Additionally, OAuth parameters (request token) are passed to the IdP. The request token includes the consumer key from the web application and the scope of the OAuth request. The user is redirected to the sign-in page of the IdP, logs in using his OpenID credentials and approves the request token. The IdP then returns the user identity and the OAuth request token to the consumer (web application).

The consumer then uses the request token to obtain an access token. It sends an access token request to the access token endpoint of the combined OpenID/OAuth provider. After verification at the IdP, an access token is returned. The consumer can then use the access token as in the OAuth protocol to request data from the user.

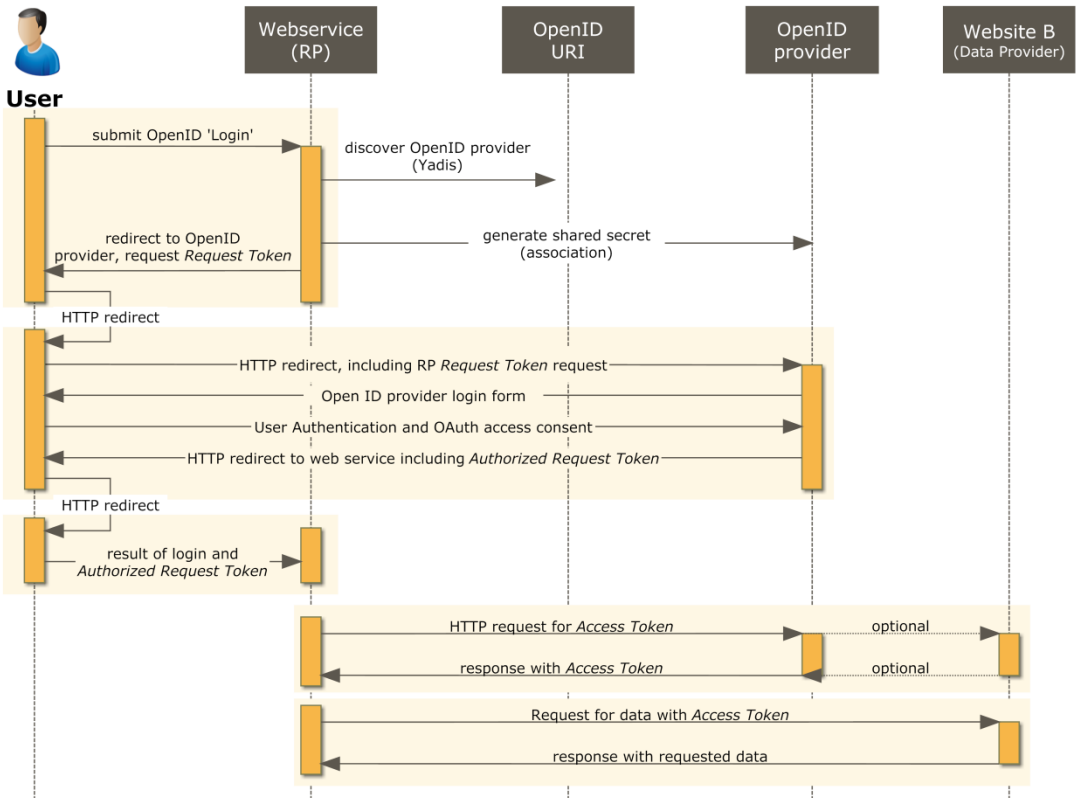

Figure 14: OpenId/OAuth Combination Protocol Flows

4.4 Classic Remote Login (TLS/SSL)

This section gives an overview of the Secure Sockets Layer (SSL) protocol and its successor Transport Layer Security (TLS). TLS will be mainly discussed as it is the latest version.

During the early to mid 90s, it was realized that there was a potential for electronic commerce; however, there was also recognition that communication across the Internet was not secure. Electronic business transactions were susceptible to being read through eavesdropping, tampering or message forgery (ref: http://tools.ietf.org/html/rfc5246). Consequently, Netscape Communications developed the SSL protocol so as to permit sensitive information such as personal details and credit card numbers to be transmitted safely on the net.

The original version was designed in 1994 but was not publicly released. Following some revision with external parties, SSL version 2.0 was officially released in February 1995 however, as a result of it containing some security flaws, a new improved version (3.0) was released in 1996. The latter was used as the basis of the Transport Layer Security (TLS) version 1.0 (also known as SSL version 3.1) which had minor differences and became an Internet Engineering Task Force (IETF) standard protocol in January 1999. The current approved version is TLS protocol version 1.2. This protocol has been adapted by many financial institutions and credit card providers for online commercial transactions.

4.4.1 Description

The TLS protocol main goals are to provide for encryption, server and client authentication and message authentication in order for communications to take place in a secure manner. In the context of the Internet Protocol Suite, the SSL protocol is situated below Telnet, FTP and HTTP in the Application Layer and above TCP and IP in the Transport and Internet Layer respectively.

The SSL/TLS protocol can be divided into two layers, namely the Handshake Protocol Layer and the Record Protocol Layer. The latter is at a lower level than the former and provides connection security that is private by means of data encryption using symmetric cryptography. It also provides message integrity checking using a keyed MAC. The Record Protocol is used to encapsulate higher layer protocols such as the TLS Handshake Protocol which is responsible for establishing secure communication or negotiate session information between client and server. This is achieved by both server and client authentication and the negotiation of an encryption algorithm and cryptographic keys.

Other protocols, within the Handshake Layer, that use the Record Protocol are the alert protocol which is used to indicate a change in status or an error condition to the peer (other party in SSL/TLS session), and the change cipher spec protocol which is used to change the raw data (or keying material) that is used to create keys for cryptographic use. The latter is in the form of a single message that informs a peer that the sender wants to change to a new set of keys (which is computed from the information exchanged in the Handshake protocol).

Some algorithms used in SSL/TLS are, for key exchange: RSA, Diffie-Hellman, ECDH, SRP, PSK; for authentication: RSA, DSA, ECDSA; symmetric ciphers: RC4, Triple DES, AES, IDEA, DES, or Camellia; and for cryptographic hash function: HMAC-MD5 or HMAC-SHA are used for TLS, MD5 and SHA for SSL, while older versions of SSL also used MD2 and MD4. Normally, the key information and certificates necessary for TLS are handled in the form of X.509 certificates, which define required fields and data formats.

The SSL/TLS process starts with negotiation, using a handshake procedure, between the TLS client and server in order to establish a stateful connection. At this stage there is an agreement on what parameters should be used to ensure a secure connection. The client presents a list of supported ciphers and hash functions to the TLS-enabled server and the latter picks the strongest that it supports and informs the client of the decision. The server hence identifies itself with a digital certificate that contains the server name, the trusted certificate authority (CA) and the server's public key. Before continuing, the client verifies the authenticity of the certificate by verifying the CA's signature on it; moreover, the client has the option of consulting a certificate revocation list to ensure that the certificate has not been revoked, and the option to reject the certificate should the CA not belong to the client's list of trusted CAs. After this phase, there are multiple ways for the client and the server to share a key. Perhaps the most common one operates as follows: the client encrypts a random number with the server's public key so as to generate the session keys used for the secure connection. This is sent to the server which is the only one capable of decrypting it by using its private key. Therefore only the client and the server have access to the data unless the private key is compromised. At this stage the handshake is concluded and the secure connection begins leaving the way for both parties to generate material to be encrypted and decrypted until the connection is closed. If any part of the above process fails then the connection does not take place.

The main objectives of the SSL/TLS protocol are:

- Cryptographic security: In order to have a secure connection between two entities, a secret key is defined through an initial handshake and is followed by symmetric encryption.

- Integrity: there is a message integrity check, in the form of a keyed MAC, used during message transport.
- Interoperability: the exchange of cryptographic details can take place without knowledge of a programmer's code.
- Extensibility: allow for new public key and bulk encryption methods to be used within the protocol so as to avoid having to implement an entire new security library.
- Relative efficiency: the protocol contains options (e.g., caching and compression) that decrease the amount of connections from scratch and thereby reduces the network activity (ref: http://tools.ietf.org/html/rfc5246#section-2 accessed 20.06.2009;)

4.4.2 Protocol Flow

A simple TLS handshake where the server, and not the client, is authenticated by its certificate involves the steps outlined below:

The client authenticated TLS handshake (CA TLS) version includes the option of a client being authenticated (in addition to the server like above) via TLS, using certificates exchanged between both peers.

The resumed (also known as abbreviated or restart) handshake is a secure shortcut in the protocol whereby a session id is sent as part of the ServerHello message. In order to shortcut the handshake, the client associates the session id with the server's IP address and the TCP port. On the server side, the session id is linked to the cryptographic parameters, such as the master secret, already negotiated. To prevent an eavesdropper from using the session id, the resumed handshake fails when the master secret is not the same. Furthermore, the fact that the data in both the client and server hello messages are random ensures that generated connection keys are different than the previous connection.

The following sequence describes the simple protocol and outlines when the client authenticated handshake and the resumed handshake differ (cf. Figure 15).

1. A client sends a ClientHello message specifying the highest TLS protocol version it supports, a random number (Nc), a list of suggested cipher suites and compression methods.

   Notation:

   ClientHello(Version, crypto_suite, Nc, [session_id])

   Resumed Handshake Version:

   Included in the message is the session id from the previous TLS connection.

2. The server responds with a ServerHello message, containing the chosen protocol version, a random number (Ns), cipher suite, and compression method from the choices offered by the client.

   Notation:

ServerHello (Version, crypto_suite, Ns, session_id)

Resumed Handshake Version (cf. Figure 16):

If the server recognizes the session id sent by the client, it responds with the same session id. The client uses this to recognize that a resumed handshake is being performed. If the server does not recognize the session id sent by the client, it sends a different value for its session id. This tells the client that a resumed handshake will not be performed. At this point, both the client and server have the "premaster secret" and random data to generate the key data to be used for this connection.

3. The server sends its Certificate(X.509) message (depending on the selected cipher suite, this may be omitted by the server)

   Notation:

   [ServerKeyExchange(Server's Diffie-Hellman(D-H) public exponent)].

   [Server's Public Key Certificate]

   Client-Authentication Version:

   The server requests a certificate from the client, so that the connection can be mutually authenticated, using a CertificateRequest message.

4. The server sends a ServerHelloDone message, indicating it is done with handshake negotiation.

   Notation:

   ServerHelloDone

   Client-Authentication Version:

   The client responds with a Certificate message, which contains the client's certificate.

5. The client responds with a ClientKeyExchange message, which may contain a PreMasterSecret, public key, or nothing. (Again, this depends on the selected cipher.)

   Notation:

   ClientKeyExchange(Pre-master Key encrypted

   Under Server's Public Key

   Or Client's D-H public exponent)

   [Client's Public Key Certificate]

   [CertificateVerify]

   Client-Authentication Version:

   This PreMasterSecret is encrypted using the public key of the server certificate.

6. The client and server then use the random numbers and PreMasterSecret to compute a common secret, called the "master secret". All other key data for this connection is derived from this master secret (and the client- and server-generated random values), which is passed through a carefully designed "pseudorandom function".

   Client-Authentication Version:

   The client sends a CertificateVerify message, which is a signature over the previous handshake messages using the client's certificate's private key. This signature can be verified by using the client's certificate's public key. This lets the server know that the client has access to the private key of the certificate and thus owns the certificate.

7. The client now sends a ChangeCipherSpec record, essentially telling the server, "Everything I tell you from now on will be authenticated (and encrypted if encryption parameters were present in the server certificate)." The ChangeCipherSpec is itself a record-level protocol.

   Finally, the client sends an authenticated and encrypted **Finished** message, containing a hash and MAC over the previous handshake messages.

   The server will attempt to decrypt the client's *Finished* message, and verify the hash and MAC. If the decryption or verification fails, the handshake is considered to have failed and the connection should be torn down.

   Switch to negotiated cipher

   Finished

   Client-Authentication Version:

   The client now sends a ChangeCipherSpec record, essentially telling the server, "Everything I tell you from now on will be authenticated (and encrypted if encryption was negotiated)."

8. Finally, the server sends a **ChangeCipherSpec**, telling the client, "Everything I tell you from now on will be authenticated (and encrypted with the server private key associated to the public key in the server certificate, if encryption was negotiated)."

   The server sends its authenticated and encrypted **Finished** message.

   The client performs the same decryption and verification.

   Notation:

   Switch to negotiated cipher

   Finished

9. Application phase: at this point, the "handshake" is complete and the application protocol is enabled. Application messages exchanged between client and server will also be authenticated and optionally encrypted exactly like in their *Finished* message.

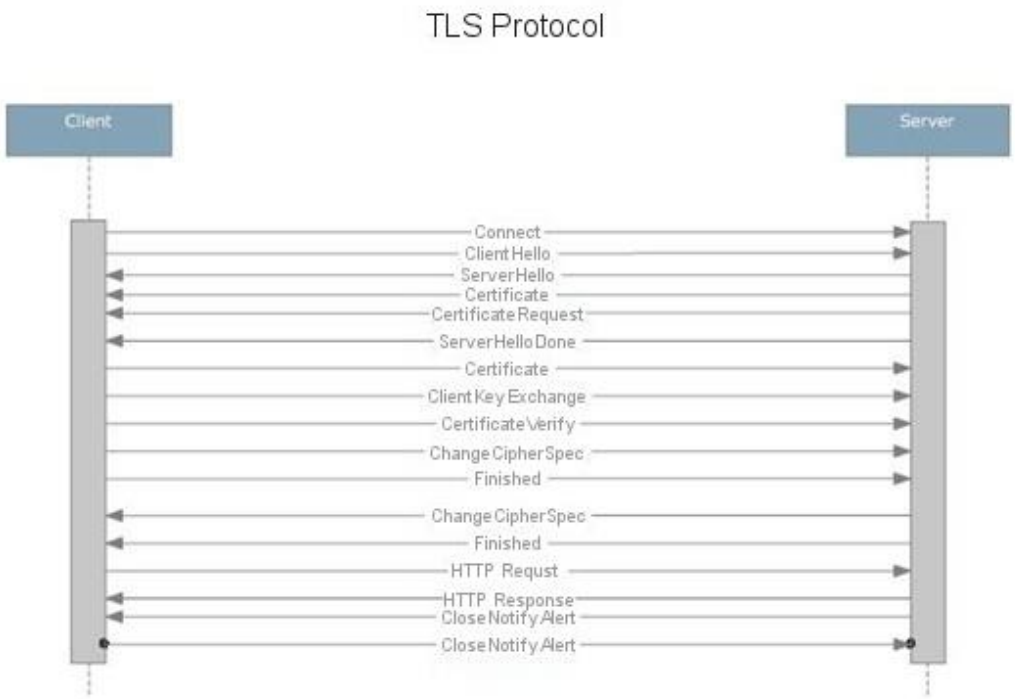

Figure 15: TLS standard protocol flow

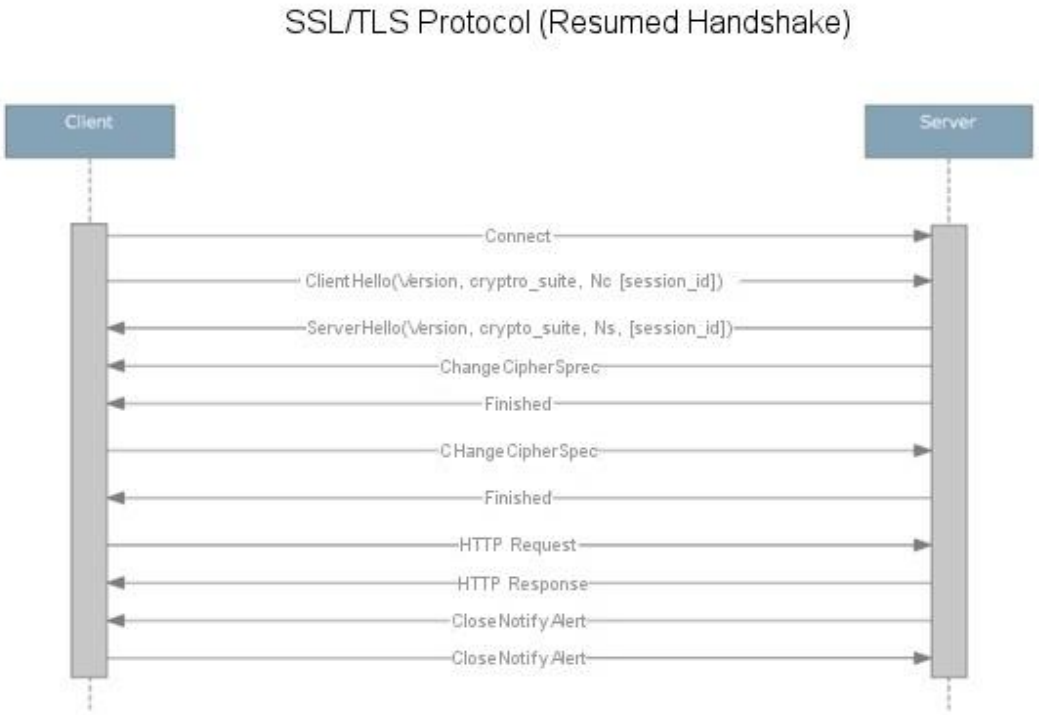

Figure 16: TLS: Resumed Handshake Variant

Chapter 5

Applicability of Candidates to IdMIP

The following chapter gives a review of the applicability of the proposed solutions to the job portal scenario and how far these can be regarded as privacy enhancing IdM Infrastructure Protocols (IdMIP). While different solutions might be possible, the current text tries to focus on a single variant for each of the candidate solutions to facilitate their evaluation. Nevertheless, variants are proposed wherever possible to show different flavours of IdMIP within a single technical protocol. The evaluation in Chapter 6 will focus on the solutions proposed in this chapter. Variants will be excluded from the evaluation and could possibly be elaborated in future work of PrimeLife.

5.1 Anonymous Credential Systems

Anonymous credentials can be used just like other credential systems to base access control to a resource upon possession of an expected attribute. In this sense anonymous credentials allow single-sign-on scenarios as shown in section 4.3.1.2 and 4.2.2.2.

For example, the Liberty Alliance protocol could be implemented with anonymous credentials. The user authentication could be based on a credential which the identity provider verifies. This example illustrates that the protocols for anonymous credentials reside on a different layer than the other presented technologies. While Liberty Alliance covers the protocol flow of browser redirection, single-sign-on and further web-based indentity federation services, the anonymous credential protocols are on a cryptographic token layer.

This fact gives anonymous credentials a lot of flexibility in the given scenario, since this cryptographic schema is not limited to standard authorization or protocol flow, but is also applicable to *data*. Any data statement in an eCV could be backed up with a cryptographic token that asserts this information. The tokens would act as trust anchors for subsequent decisions on the data. In principle, this could be realized with any cryptographic signature system. For instance, some trusted third party could attest with its signature that a statement is true. However, using anonymous credentials adds privacy-related features in such application scenarios.

We want to motivate the attestation of eCV statements with anonymous credentials by an example. Let's assume that Alice possesses an eCV data record with the following facts:

- Name and date of birth

- University degree
- Current employer
- Recommendation from customer

In a standard data record she just needs to drop certain facts (such as her name and date of birth) to anonymize the whole record. Unfortunately, this does not allow the receiver of the record to verify any of the stated facts. An obvious way to achieve that is to annotate each data item with an attest signed by somebody who can certify this fact as true. The resulting record using standard signature schemes would look like this:

No	**eCV Data Item**	**Certifications signed by**
1	Name: Alice, Date of Birth: 1/2/1973	Government certifies that person does exist and date of birth
2	Master Degree in Information Technology from Fabrikam University	David, the Dean of Fabrikam University certifies a degree in Information Technology
3	Works at company Contoso as consultant	Frederick, Alice's manager, signs that she is with Contoso
4	Excellent recommendation for her great work from customer Woodgrove Bank.	Charlie, working at Woodgrove Bank, attests her great consulting achievements

This data set could be shared by Alice. But even if she omits her identity (data item 1), she is identifiable, since her name may appear in the attests accompanying the real data items. Dropping all attestations is no option, since Alice may want to prove that she really has a university degree and for this she needs a way to prove that fact without disclosing her identity though the attestation. Moreover David, Frederick and Charlie could collaborate and identify Alice.

Obviously, using standard signature schemes has a number of privacy drawbacks:

1. The owner of the information, the data subject, may want to disclose information anonymously. Although no information is in the data set, it may be included in the attestation.
2. Alice cannot take advantage from partial disclosure of a signed data set. She may want to reveal that she has some Master degree, but not her main subject. Since she cannot alter David's signature she is forced to disclose all or nothing.
3. Receiver of a data set and the issuer of an attestation could collaborate in order to identify Alice.

Applying anonymous credentials here would solve the problem. If all parties would use them, Alice could disclose only data items she wants to share and alter the certifications in a way that she blinds out information she does not like to share. She could omit any information pointing to her.

Alice could anonymize her data from the above example like this. The blinded data is noted as stroked-through text:

No	**eCV Data Item**	**Certifications signed by**
1	Name: ~~Alice~~, Date of Birth: ~~1/2/1973~~	Government certifies that person is

		does exit and is above 18.
2	Master Degree ~~in Information Technology~~ from Fabrikam University	~~David the dean of~~ Fabrikam University certifies a degree ~~in Information Technology~~
3	Works ~~at company Contoso~~ as consultant	~~Frederick, Alice~~'s ~~manager~~, signs that she is with Contoso
4	Excellent recommendation for her great work ~~at customer Woodgrove Bank.~~	~~Charlie, working at Woodgrove Bank,~~ attests her great consulting achievements

Please note that the eCV data item is just a data record. Omitting data here simply means altering the data set. Unlike the data item, the certification is an anonymous credential and can thus be altered while keeping the issuer's signature valid. The owner of the credential (Alice) is able to hide attributes without interaction with the issuer. The issuer's cryptographic signature remains valid after such operations.

Now, Alice is in the position to alter her own CV data and limit the information given through assertions. If she carefully selects the information she can create an anonymous CV that is to some extent verifiable by a third party. Certainly, she has to be very careful to blind all identifying information when she wants to stay anonymous.

Now we want to look at the eCV protocol flows specified in section 2.1 and describe how they need to be altered in order to take advantage from anonymous credential attested information. We will not describe the fact that anonymous credentials can be used as login tokens since this is straight-forward would not add much to the already described scenario. Instead we describe how to use them as attestations for data items.

5.1.1 Publishing an eCV

Applying anonymous credentials to the publishing step introduces those types of claims to the system (cf. Figure 17). Before actually submitting the eCV, the user needs to get the claims. This could be an out-of-band communication happening even way earlier that the creation time of the eCV. We assume that all parties are either technically able to understand and verify the claims.

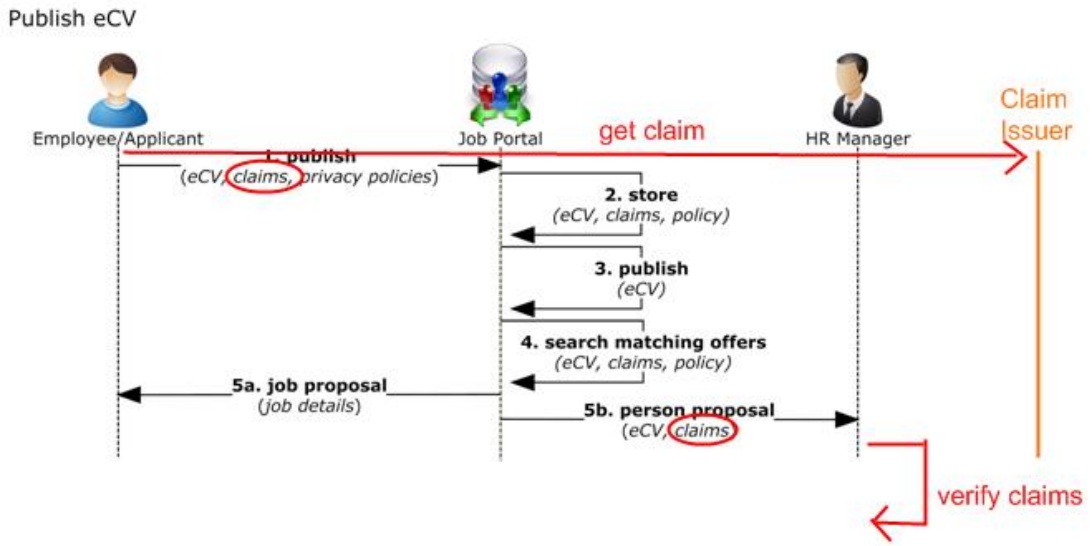

Figure 17: Protocol Flow of eCV publication

1. The employee retrieves claims that back up his statements from the claims issuer. This step will be repeated multiple times, even with different issuers. It could even be an out-of-band communication.

2. The employee creates the eCV he wants to publish. He adds only those data items that he wants to disclose. Similarly he blinds information from the certification that would identify him and/or that he does not want to share.

3. The employee/applicant publishes his eCV, claims and privacy policies which define the access control to the published data to the job portal.

4. The job portal stores the received data.

5. The job portal then publishes the eCV, allowing HR managers to search for possible candidates matching their criteria.

6. The job portal performs an automatic search for matching job offers in the job offer database depending on the eCV data, claims and the defined privacy policy.

5.1.2 Publish Job

Job publishing by an HR manager is not affected by the anonymous credentials. Certainly, the HR manager may want to withhold data as well, but the main privacy concern in this scenario is on the employee/applicant. Therefore we focus on interactions with the applicant.

5.1.3 Candidate Matching

As illustrated in Figure 18, protocol-wise the matching procedure does not change. All that is introduced is an optional step where the HR Manager verifies the certification backing up the claims in the profile (6b, 9).

It is important to note that the applicant could send the same anonymized eCV to the HR manager twice and the HR manager is not in the position to proof if both eCV records belong to the same user or not, given that the eCV itself is properly anonymized.

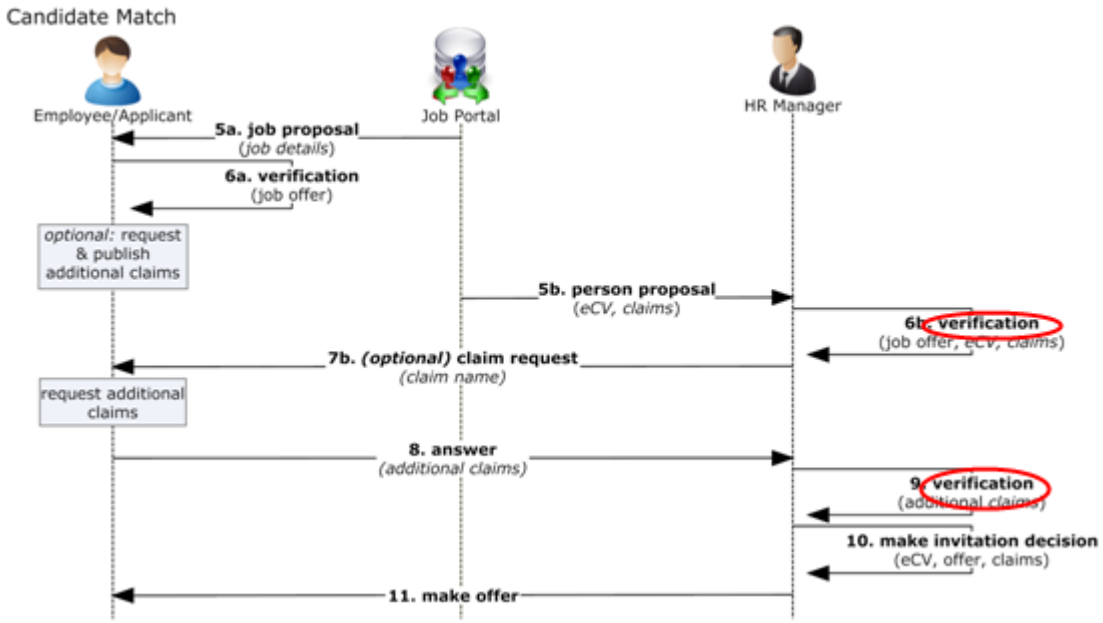

Figure 18: Protocol Flow of candidate matching

5.2 Liberty Alliance

As Chapter 4 indicates, the Liberty Alliance framework and its services are not limited to be used for a single use case or interaction process. The complex Liberty framework allows a lot of variations and flexibility, depending on the requirements of a specific scenario. As showing all possibilities for the implementation of the job portal scenario with Liberty technology is not the aim of this document, we will concentrate on a limited set of use cases. The purpose of this section is to give a feeling for the ability of Liberty to be used in an infrastructure scenario like the job portal. Therefore, we will employ as much of the LA services as possible. Figure 19 shows the interaction of the entities involved in the Liberty enabled Job Portal scenario. The order of the different interaction sequences should not be regarded as compulsory. That means that these steps could be performed in any order. The set of steps which should be regarded as atomic are marked by lower-case letters. In the following the scenario will be described in an arbitrary chosen interaction sequence. Any step will be referenced to the respective sequence in Figure 19.

Involved parties:

- Global LA IdP: A global Identity Provider which manages basic identity information of a user. These could be a user id, name, address, date of birth, etc. This IdP is independent from the job portal.
- User/Employee: Person that uses the job portal to find new jobs.

- Headhunter: Uses the job portal to find suitable candidates for specific job positions.
- Inhouse Trainer/Claim issuer: Backs up assertions of an applicant.
- Job Portal: The job portal service infrastructure.
- Claim storage (secondary IdP): A service provider which acts as storage for claims that an applicant already possesses. Through several service interfaces other SP have the possibility to request for claims of a user.
- LA Discovery Service: This service provides information about which IdP provides specific identity information.

5.2.1 Complex Liberty enabled Job Portal ecosystem and scenario:

In this Liberty-enabled job portal scenario Inga Vainstein, an applicant looking for a new job position, first logs in to her Global IdP (1). The Global IdP manages basic identity information (name, date of birth, etc.) of Inga. It is used as main access point for basic identity related information of Inga. Next, Inga logs in to the claim storage, which quasi acts as a second IdP (2). Inga stores all claims that she already possesses at the claim storage. Access to these claims is only available for Inga herself or by explicit permission from Inga. By using the Liberty "Identity Mapping Service (IMS)", the two identities of the (different) accounts from the global IdP and the Claim Storage are "linked". This is not the same as identity federation. IMS is especially useful when it is required to map identity information of two or more different identity providers during a single service usage session. For this step, there is no need to establish complex business relationships as it would be necessary for identity federation. In the job portal scenario the IMS enables the claim storage (second IdP) to refer to the correct identity when providing claims (3). After that, Inga uses Liberty SSO to login to the job portal for the first time (4a). For the job portal it is essential to know where Inga stores her claims. Therefore, the job portal requests Inga's discovery service for the claim storage provider (5a). The discovery service returns a handle to Inga's claim storage that contains the necessary information to be able to interact with it. As there will be a continuing relationship between the job portal and the claim storage during the whole time when Inga is searching a new job position, it is worthwhile for these two systems to federate the two identities managed by the two. This will allow Inga to single-sign-on to one of these services and have access to both of them. Actually, the claim storage could be a component of the job portal, anyway. This would simplify the establishment of the business relationship needed for identity federation. Now, as the accounts of the job portal and the claim storage have been federated, and the accounts of the claim storage and the Global IdP have been mapped, the job portal can request the Global IdP for additional identity information by using the Interaction Services (7a). Additionally, the job portal requests the claim storage for claims of Inga (7a). This step also requires authorization by Inga. The Interaction Service is used here again. Hannes, a headhunter, finds Inga's profile and thinks that she is quite suitable for an open job position for which he is responsible. To get more information about her background and qualifications, he wants to get additional claims from her. Therefore, he uses the People Services to get information about the former employer of Inga (4b). Having this information, Hannes requests the discovery service where he can find the respective claim issuer (e.g. web service of former employer) (5b). In order to get access to the claims, Hannes uses the Interaction Service to get Inga's permission on this (6b). Inga classifies the job position Hannes offers as very lucrative and authorizes the access to the claims.

A couple of days later Inga is sceptical whether her eCV portfolio is enough convincing, because she received less job offers as she expected. Inga holds that filling her eCV portfolio by further signed claims could help in being more attracting for headhunters. Therefore she requests several institutions where she graduated in off-the-job trainings for such claims. For this purpose she has to authenticate herself to the respective services of the training institutions (4c). Some of the service providers already had federated the accounts of Inga with the Global IdP. This helps in identifying Inga (5c). After successful authentication and identification, the claim issuers and trainers issue claims for the trainings Inga has completed (6c). Inga wants these claims to be accessible via the claim storage; hence she uploads them to her claim storage (7c).

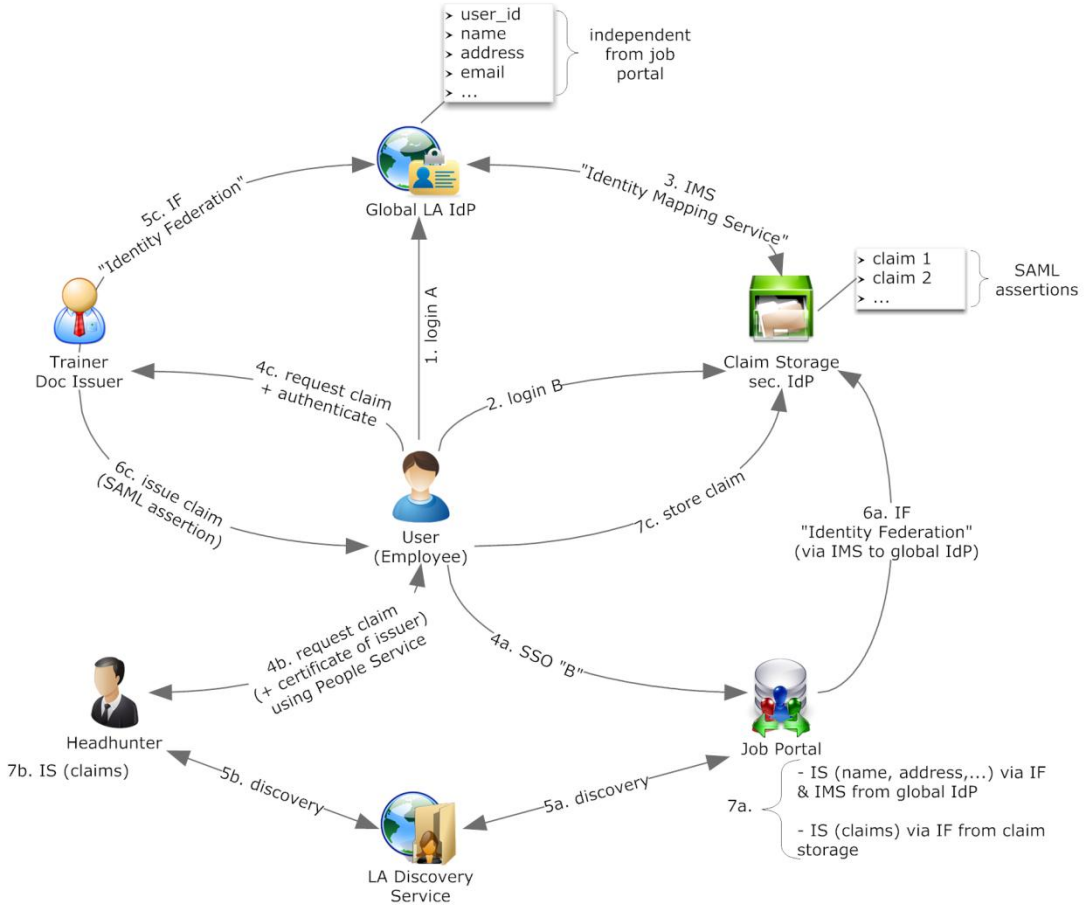

Figure 19: Mapping of Liberty Alliance services to Job Portal

5.2.2 Scenario-specific variations

The scenario described in the previous section is applicable to both the Inhouse Platform and the Open Platform solution. But in that example it is presumed that all involved parties form independent domains. For the Inhouse Platform the system could be implemented as an intraorganizational platform where all transactions take place inside that intranet domain which is not accessible by externals. For the Inhouse Platform the claim storage could be a subsystem of the job portal and the global IdP is the central identity provider of the company. As a result, there is no need for identity federation or mapping between IdP, claim storage and job portal. The IPS can be realized without a special Liberty discovery service, as one can assume that all service

providers reside within the company intranet and are easy detectable and the HR Manager of that companie knows where he can get claims of an employee. Still Liberty SSO can be used if access to the job portal and the claim storage is implemented separated from access to further intranet services of that company. One potential possibility to use identity federation is to federate the accounts at the inhouse trainer with the central IdP of the company. This can make sense when the inhouse trainer is an external and is not allowed to access all company databases.

5.3 OpenID/OAuth

In order to evaluate the applicability of OpenID/OAuth to the eCV scenarios, we first present a possible solution which maps the concepts of OpenID/OAuth to the application scenario. We can then evaluate our proposed solution along the defined evaluation criteria. While our solution aims at providing a maximum of control over the data to the user, it does not preempt other applications of the OpenID/OAuth protocol to the given scenario.

The solution tries to accomplish the following objectives:

1. The user has the ability to control to whom he grants access to (partial) information
2. The job portal should be able to make job offer/request suggestions by e.g. matching different criteria from offers and requests
3. For the job portal to conduct these operations, it will have to read user-supplied data which is probably privacy sensitive. A trust relationship must be established between the user and the job portal.

5.3.1 Mapping of OpenID/OAuth

The solution divides the job portal in 4 different subsystems. Following a separation of duties and tasks of the respective entities, the following subsystems can be identified (cf. Figure 20):

- The User Subsystem includes all functions the user must have access to. It includes mechanisms for authentication, options to submit, manage and store CV information to the eCV storage system, access control mechanisms which allow the user to define policies for the access to his stored data.
- The HR subsystem is able to authenticate HR managers and provides them with options to issue job requests and searches as well as claims for specific users.
- The Trainer subsystem is similar to the HR subsystem. Trainers are able to provide documents and claims for users.
- The Storage subsystem is the central place where all user supplied data is stored. This includes CV data as well as additional data such as certificates from trainers or HR department.
- The central job portal manages the subsystems and is the entity which is able to find matching job offers and requests such that it can make suggestions to HR or users.

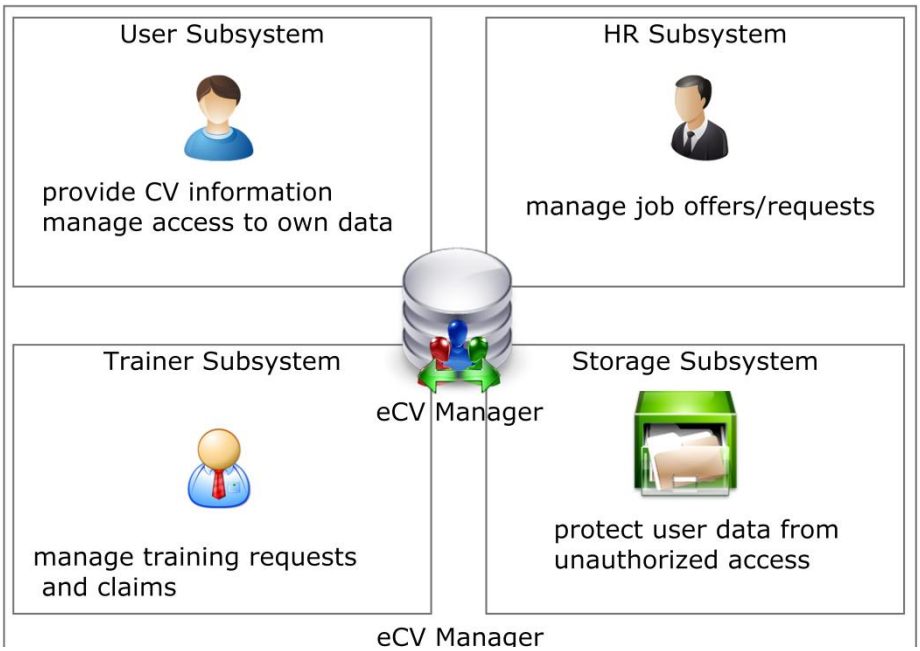

Figure 20: Job portal subsystems

Description of OpenID/OAuth application to eCV scenario (cf. Figure 21):

1. The user logs in to job portal using OpenID
2. After successful authentication, the user is allowed to submit/store/remove/change CV data in the job portal (storage subsystem)
3. The user authenticates to the job portal (HR/Trainer subsystem).

   Note: this subsystem could be a (sub-) webpage of the job portal which allows the user to select which HR managers / trainers to allow access to his data, e.g. users might want to deny access to their job requests to HR managers of the company they are currently working for.

   Note: The user will not have to authenticate again, if eCV user subsystem can pass on the authentication state to the HR/Trainer subsystem, e.g. the same logon-session to the webpage
4. After successful authentication, the user is able to add, modify, delete data in his stored eCV using the eCV user subsystem.
5. If the user wants to grant access to this data to third parties (i.e. HR or trainers), he can log on to the respective subsystems of the third party. Technically, this could for example be realized by a webpage inside the eCV platform the user visits. The webpage allows him to easily define policies by e.g. selecting to publish all or some information to all or some third parties, as for example excluding the current employer to see the request for a new job.

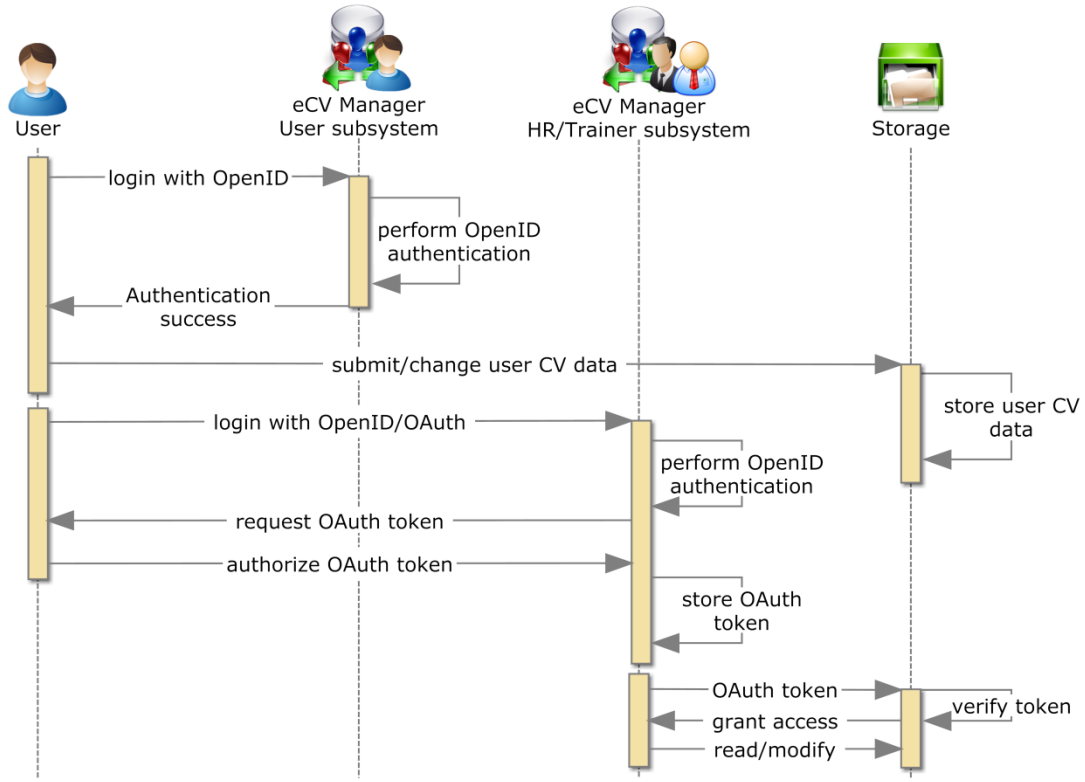

Figure 21: Subsystem interaction sequence diagram

6. The user defines/decides which groups/individuals to give access (read/write) to his documents in the eCV storage, e.g. all HR managers of a specific company. eCV HR/Trainer subsystem sends an appropriate OAuth token to the user. If the user gives his consent to the desired action, an authorized OAuth token is returned and stored for future usage in the eCV HR/Trainer subsystem.

   Note: HR managers / Trainers can logon to the subsystem to enter data access requests. These are presented to the user when he logs on to the HR/Trainer subsystem and he can allow or deny access.

   Note: such requests could be auto-generated by the job portal, if for example HR issues a job search and job portal find all matching candidates. job portal then requests OAuth tokens from the respective users on behalf of the HR.

7. HR managers/trainers can logon to their respective subsystems. They might be presented with suggested candidates they need to act on (by e.g. asking them for details, issuing a certificate, etc.). If no OAuth token is available for this user, they could trigger a request, which can be approved or declined by the user the next time the user logs on to the system. Depending on the decision the user takes, HR/Trainer will be able to see or modify the user data. This puts the user in control over his data.

8. The stored OAuth token can be used to access the specific user data. Depending on policies, tokens can be one-time access tokens or be used multiple times.

9. OAuth tokens will only be stored in the HR/Trainer subsystem and made accessible to HR managers/Trainers only after they authenticate to the HR/Trainer subsystem.

This solution requires all actors to establish a trust relationship with the job portal. Especially the storage and management of data and access tokens must be trusted. Since user stored data is accessed directly by HR/Trainers, one option is for the data to be unencrypted in job portal storage subsystem. The concept presented herein requires a strong trust relationship between user and the job portal.

5.3.2 Scenario-specific variations

The introduced application scenarios for the adoption of the given protocols for achieving a privacy enhanced eCV management system have its benefits for both IPS and OPS. In both cases the proposed OpenId/OAuth combination enhances privacy of an employee/applicant by leaving him the control over the amount of data and its details to be provided to interested parties. Nevertheless, the differences in the architectural characteristics of the two infrastructures require a differentiating view for the way of implementing the protocols. That arises from the different composition of the domains of interested parties. The consolidation of all domains of interest under the corporate domain facilitates the restriction of access to the platform to any domain but the in-house. This has for instance the (possible) implication that the eCVs published by an employee – assuming that adequate access control mechanisms are established in the company intranet – can be regarded as in-house confidential data and therefore are not accessible by any external entity. From the corporate point of view such an implementation is essential because the PI platform can be regarded as an enabler for company internal employee movements which can be categorized as a confidential process.

From the employee/applicant point of view, the privacy threats in the IPS solution are less than in the OPS solution. The reason for this is that the HR department at least possesses portions of personal data (name, address, day of birth, CV, etc.) of a current employee anyway. But still the usage of OpenId/OAuth can be used to enforce the privacy enhancing mechanisms just as described in the general scenario.

Considering the needs for concealment of company-internal information, one important facility for the PI solution is the in-house implementation and provision of the service providers. Implementing the OpenId provider as an intranet subsystem could guarantee that company internal data is never passed to external entities. The same holds for the implementation of the OAuth token provisioning, which – as described in the previous section – can be merged into the IdP.

5.4 Transport Layer Security Protocol

Mapping of the TLS protocol to both the Inhouse Platform Solution (IPS) and the Open Platform Solution (OPS) will be done in the context of the Simple TLS Handshake, Client-Authenticated TLS Handshake and Resumed TLS Handshake.

The Inhouse Platform Solution consists of a central job portal service to create and edit CVs, post job requests and offers, provide or request claims. In this solution, three roles were defined in Chapter 2, namely Employees (request job offers and claims from HR manager, send CV information and job requests to the HR Manager, request training offers and claims from Inhouse Trainer, and provide training requests and CV information to the Inhouse Trainer), HR Manager (receives job requests and CV information from Employee and provides job offers and claims to the Employee) and Inhouse Trainer (receives training requests and CV information from Employee and provides training offers and claims for the Employee).

In the Open Platform Solution, we have the following actors: the User, the Headhunter and the Claim Issuer. The User interacts with the Headhunter (request job offer, provides job requests and CV information) and the Claim Issuer (request claims for the CV-Portfolio and provides CV

information). The Headhunter interacts with the User (provides job offers, requests job requests and CV information). The Claim Issuer interacts with the User by requesting CV information and proving claims for the CV-Portfolio.

As TLS provides a means to securely communicate between peers, each actor's connection to either to the inhouse job portal (IPS) or the subsystems (OPS), will use the protocol in the same way, therefore one actor is chosen for the examples.

The analysis of the applicability of TLS does not differ between the two platform solutions as the TLS protocol does not change essentially. Furthermore, the trust relationships can be considered in the same manner since TLS is concerned with providing a secure communication between peers.

5.4.1 Simple TLS Handshake

Assumption: No client authentication is provided in this version of the TLS protocol, therefore we assume that this is provided by another protocol that supports TLS. Subsequently this part of the sequence is left out in the description. All actors logging on to the different systems will follow the same server authentication TLS process as outlined below. The protocol applied to the Employee's log-in would be as follows:

1. The Employee logs on to the Job Portal, or a subsystem as in the case of OPS, using the simple TLS protocol.
2. After the server has been successfully authenticated through the TLS process, information exchanged will be encrypted. The Employee can, at this point, securely modify or add CV data in the job portal (storage subsystem). In the case of the OPS, the user could request an authentication of the HR/Trainer subsystem through the TLS process, once this is accomplished the user could add, modify and delete data in his stored eCV using the eCV user subsystem.

5.4.2 Client-Authentication (CA) TLS Handshake

This is similar to the simple handshake however this time the server asks the client to authenticate itself (cf. Figure 22).

1. The Employee logs on to the Job Portal, or a subsystem as in the case of OPS, using the client-authenticated TLS protocol.
2. When both the client and server are authenticated the information exchanged will be encrypted.
3. The user can, at this point, securely modify or add CV data in the job portal (storage subsystem). Similarly, in the OPS, the user would like to use the HR/Trainer subsystem once this is accomplished through the TLS process, the user can add, modify and delete data in his stored eCV using the eCV user subsystem.

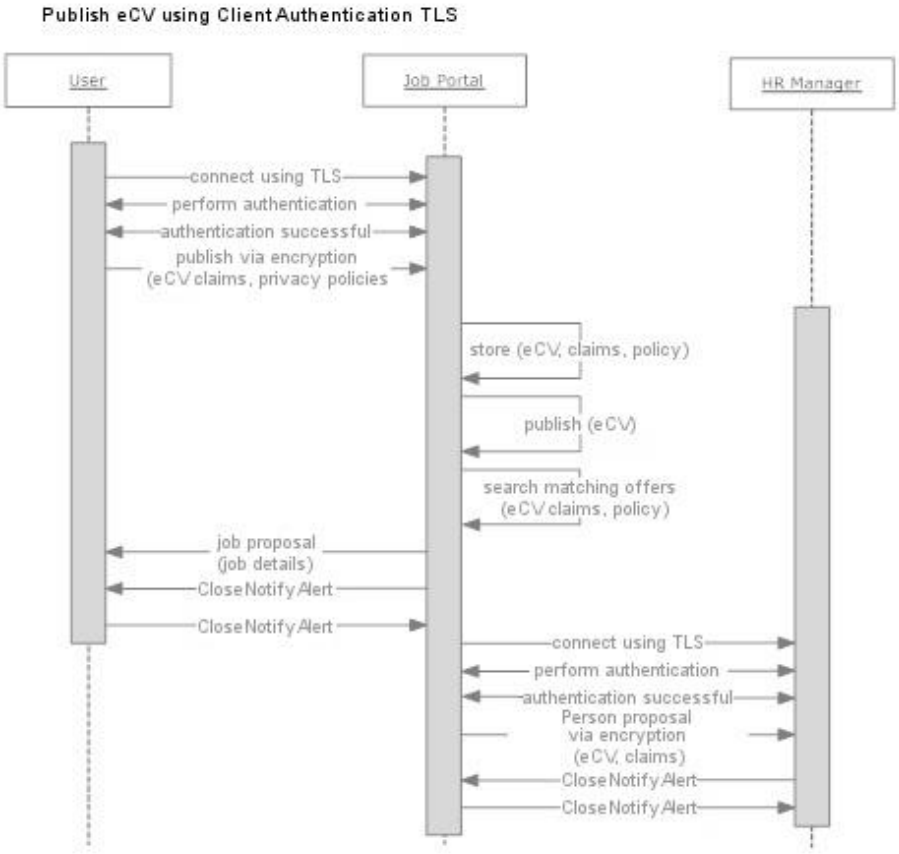

Figure 22: Publishing eCV with SSL/TLS protocol

5.4.3 Resumed TLS Handshake

This is similar to the client authentication handshake however this time the client sends a session-id from a previous session it had with the server.

1. The Employee logs on to the Job Portal and included in the message is a session-id.
2. If the server recognizes the session-id sent by the client, it responds with the same session-id. The client uses this to recognize that a resumed handshake is being performed. At this point, both the client and server have the "premaster secret" and random data to generate the key data to be used for this connection.
    a. If the server does not recognize the session-id sent by the client, it sends a different value for its session-id. This tells the client that a resumed handshake will not be performed.
3. As the authentication of both parties has already being established in the previous session, the information exchanged will now be encrypted.
4. The Employee can, at this point, securely modify or add CV data in the job portal (storage subsystem). In the OPS, the user's login to the HR/Trainer subsystem will follow the same procedure above which will subsequently permit the user to add, modify and delete data in his stored eCV using the eCV user subsystem.

Chapter 6

Evaluation of IdMIP

In this chapter we want to examine how far the protocols introduced in Chapter 4 can meet the requirements defined in Chapter 3 in the job portal scenario introduced in Chapter 2. As mentioned in Chapter 5, this evaluation will be done along the job portal scenarios introduced in Chapter 5 in conjunction with the mapping of the specific protocols to the job portal scenario. This chapter also should give a feeling for the advantages and disadvantages of the protocols in the context of a job portal scenario. Another question that should be answered here is how well each of the introduced protocols fulfils the different IdM- and privacy-specific criteria defined in Chapter 3. The evaluation will be done separately for each protocol and each scenario, if possible. Afterwards, a general evaluation of the scenario and the protocols in its entirety will be done.

6.1 Evaluation of Protocols

In this section the proposed candidate protocols of Chapter 4 will be evaluated against the criteria of Chapter 3.

6.1.1 Anonymous Credential Systems

This section evaluates the technology of Anonymous Credentials in two ways. First we compare it against the criteria defined in section 3. We will look at each criterion and give a short assessment whether anonymous credential systems contribute to the goal. Second, we assess to what extent anonymous credential systems are useful in the job portal scenario defined in section 2.1. The technical description in section 4.1 does not distinguish between inhouse solution and platform solution. Consequently, we cannot distinguish in the assessment either. Hence, all statements apply to both scenarios setups, unless we mention it explicitly.

Chapter 3 lists five domains of evaluation criteria: support for role concepts, privacy support, manageability, usability and economic viability. Looking at the support of role concepts one sees that anonymous credential systems are a perfect match to all criteria in this group. Strict role assignment (R.1) is well supported. The technology allows a binding to arbitrary attributes including roles. Anonymous credential systems even decouple the identities from the attributes so that just the attribute matters. In contrast, common attribute certificates such as X.509 support attributes but the data subject is forced to disclose its identity each time it makes use of an

attribute certificate. The criterion on strict competence clarification (R.2) is not supported per se by the technology. However, the fulfillment of R.1 helps to achieve R.2 as well. So the eCV system has to make sure that functions associated with a role do not allow gaining extra privileges. The eCV system needs a well thought role concept to achieve this. The roles can in turn then mapped to attributes represented in claims. Anonymous credential systems are made to fulfill the criteria on integrity (R.3) and verifiability (R.4) of information. They even allow disclosing a subset of attributes encoded in the credential, and still keep the integrity and verifiability.

Since we do not use anonymous credentials for authentication the respective criterion (R.5) is not fulfilled. But this is not because anonymous credential system could not help realizing this feature (in fact they are made for exactly that) but we do not use them for authentication in the eCV scenario.

The privacy criterion access control to own data (P.1) is not directly fulfilled with anonymous credentials. Rather, anonymous credential systems allow the user to address this issue by means of anonymization. Hence, the user controls what fraction of information she want to share but once disclosed she has no control who gets access.[12] However, this feature actually fulfills criterion P.2 on reduced identity information; the user can indeed control the amount and the level of detail. The remaining privacy criteria are not applicable to this technology. They are targeted at the overall eCV solution, not specific technology pieces. Hence, anonymous credentials do not solve the authorized dissemination of data (P.3), covering tracks (P.4), and confidentiality for data sent over the Internet (P.5).

From the manageability criteria, the criterion on interoperability (M.2) is the most interesting one. On one hand anonymous credential systems are not standardized, yet. On the other hand, presenting a token to a policy decision point is a procedure found in most access control systems. Using anonymous credentials for encoding the eCV data as we suggested in section 4.1 is certainly not the easiest and most interoperable way to transport this kind of information.

Data minimization (M.3) on the local scale is in the hands of the user. The technology supports the minimization as described for P.5 above. On the system scale it depends on the actual implementation where data is stored and if it is replicated in the system. Scalability, as defined in criteria M.4, is indeed an issue. We assume that it is more difficult for the user to import data if they should or even have to be encoded in anonymous credential tokens. As for the remaining manageability criteria on minimal competence (M.1) and user adoption (M.5) we cannot make any valid statements since they are targeted at the overall system again.

As for the usability criteria, the ease of use (U.2) certainly largely depends on viable user interfaces for selective disclosure. In the PrimeLife project Activity 4 is actively working on that so that we cannot make an assessment, yet. Intermediate results, namely the first HCI report D4.1.1, states that anonymous credential systems are complex technical concepts which "are unfamiliar to many end users and often do not fit to their mental picture of what belongs to an electronic identity and how it can be technically protected." [PH09]. So we cannot say that the integration of anonymous credentials in the eCV scenario setting does improve the usability. The integration (U.1) of anonymous credentials is also difficult. In case they are used for access control, both sender and receiver have to understand the cryptographic authorization tokens. In case anonymous credentials are used as data containers, what we motivate in section 5.1, both the issuer and the consumer of the tokens have to share the same semantic understanding of attribute and their values, such as "Alice has a master degree in information technology". This is difficult to achieve on a general scope, but might be feasible in a well defined scenario scope.

---

[12] Certainly, this holds true for the technology of anonymous credentials in the eCV scenario only and is no statement about the implementation of the eCV scenario as such.

We cannot make a real assessment on the economic viability of the solution E.1. The technology is either available as open source (Idemix) or will eventually become part of a bigger product (U-Prove).

6.1.2 Liberty Alliance

This section evaluates the capabilities of Liberty services that were introduced in Section 4.2 against the criteria defined in Chapter 3. As basis for the evaluation the Liberty-enabled Job Portal Scenario introduced in Section 5.2 is used.

6.1.2.1 Evaluation of Liberty-enabled scenario against criteria

The Liberty Alliance Framework provides several mechanisms to support identity related applications. The presented solution for the job portal scenario tries to include several relevant Liberty Alliance protocols to enhance the user's privacy and security while maintaining a good level of overall usability.
All claims are embodied in SAML assertions. The issuer of the SAML assertion signs the statement and thus provides integrity protection for the contained data (R.3). Furthermore, the issued claims and their issuers can be identified and verified (R.4). Every action requires the user to authenticate using either direct authentication, e.g. against the claim storage, or indirect via the LA identity federation process (R.5). However, the LA protocols as used in our description include no options for the direct assignment of roles and a strict separation of competences (R.1, R.2).
LA allows covering most of the privacy relevant requirements. With the use of IS protocols, access control to the data is established, allowing the user to decide which data to disclose. The additional option to define policies further increases the user's power to control access to stored data (P.1). The concepts of IF allow to minimize the spread of identity related information. The user will store globally, unique identity parameters such as name, address, etc. at a trusted global IdP, while the claims can be stored separately at the claim storage. This prevents privacy leakage through linking of different data subsets (P.2). LA does not provide any means to cover tracks or control the unauthorized dissemination of data. While the data is sent over the network it is desirable to protect its confidentiality when needed (P.5). LA itself does not specify how this can be achieved, however it allows for the extension to other protocols such as TLS to provide encryption for data travelling across public networks.
The LA framework is a very complex interaction scheme, which requires the introduction of multiple services at different entities. This could lead to problems in the adoption by users (M.5). The use of standard SOAP and XML structures increases interoperability as long as other, additional entities are also LA capable (M.2). Since identity information only has to be stored once, the data amount can effectively be minimized (M.3). The IMS mapping of identities between the claim storage and the global IdP has to be performed for every new user, thus leading to possible scalability issues (M.4).
Regarding usability, it has to be noticed that Liberty Alliance provides the framework for the protocols without anticipating a concrete application. Thus, the usability will depend on the actual implementation of the framework standards. By using standardized and already deployed protocols (XML, SOAP, web services), any LA based implementation will easily integrate into existing solutions (U.1). Furthermore, all standards and protocols are open, thus reducing cost for the introduction of the protocol suite.

An interesting detail is that the 3GPP TR 33.980 standard defines a simple SSO service of ID Federation, connecting the 3GPP mobile world with Liberty Alliance protocols. It is the application of Liberty Alliance ID-FF/ID-WSF standard in the wireless environment and thus could increase the user experience when accessing LA enabled services via mobile phones. The

mechanisms allow the user to authenticate using a key which is sent securely to the USIM in their mobile phone. When the user wants to access a LA enabled service, he will use his subscriber identity which is already registered with the mobile operator. The HSS (Home Subscriber Server) performs authentication via the user's mobile phone, and after successful authentication, a shared secret is exchanged with both the user's mobile phone and the service provider. The shared secret is stored securely in the user's mobile device and is then used to access the service.

6.1.2.2 Applicability of Liberty Alliance to the scenarios

As described in Chapter 4, Liberty is more a complex framework than a protocol with limited specific functionalities. The comprehensive potentials that the Liberty framework offers can be used as a basis for privacy enhanced infrastructures. We have seen that especially in the Open Platform variant of the job portal scenario there is a need for interoperable and scalable protocols, because the actors of the architecture are from different domains. This requires special care of access controls and policies for them. Exactly this property of the OPS is challenging. The challenge lies in the implementation and establishment of an architecture that can control the privacy-awareness despite this heterogeneity. Liberty seems to have suitable solutions for this problem. If the web services in the presented frameworks are used, privacy-awareness can be established for scenarios like the job portal. As seen in the previous chapter, services like identity federation, identity mapping, people services and interaction services build stable basis for that. Through the ability to define access policies based on the user requirements and the ability to easily build "circles of trusts", Liberty technologies seems to be quite adequate for such business scenarios.

On the other hand, data transfer is only performed inside the corporate domain in the inhouse scenario. Thus, we can assume that proper trust relations exist between the actors involved in the IPS. There is no need for special business agreements or circles of trust. Nevertheless, Liberty services like SSO or identity federation could be utilized in order to offer user convenience and ease of management.

6.1.3 OpenID / OAuth

The proposed solution for the application of OpenID/OAuth protocols to the eCV scenarios focused primarily on the following objectives: the user decides to whom he grants access to information, the job portal is able to provide suggestions to applicants and headhunters and therefore is able to read all supplied data to perform the search and match operations.

Another concept is to hide all privacy sensitive data from the job portal. The job portal would then have no possibility to act upon the data and provide useful job suggestions to users. This could be achieved by the encryption of all data sent to the job portal such that job portal is unable to extract valuable information. Such an encryption would have to rely on private keys for the users which are unknown to all other participants, e.g. embedded in a smart card or a trusted device. The job portal would then allow storing encrypted data which is encrypted for the target audience. As the job portal is then unable to decrypt the eCV data, its functionality is dramatically decreased to that of a storage and key exchange provider. Applicants and HR managers would use the job portal to authenticate using OpenID and retrieve the public keys of the target audience. The applicant can then encrypt the data using the retrieved public key and use the OAuth mechanisms to grant access to this data. This protection is then somehow redundant, since the encryption already guarantees that no other party will be able to decrypt the eCV. The job portal would solely be used to perform an authenticated key exchange based on the OpenID authentication.

A compromise would be to separate the data into public and private portions. Private portions are encrypted and never revealed to the job portal, but linked to a set of public information. The

public portion is less privacy sensitive and can be known by the job portal. Based on the public information, the job portal can compare job requests and offers and provide valuable suggestions to all users. The user could then decide which information (e.g. name, city, educational background) is public and which are private (e.g. gender, nationality, credentials). The trust relationship between user and job portal covers for the protection of the public data while the private data is cryptographically protected. Such a system allows the job portal to operate on the given (public) data in a limited way compared to the complete public data solution. Furthermore, a system must be established for the users to share their encrypted data stored on the job portal with third parties, e.g. HR/Trainers. Key sharing and key exchange protocols, either directly between user and the third party or managed by the job portal are to be elaborated. The implications and requirements are subject to further discussion.

Another variant stores highly sensitive data in a separate storage directly in the user's system. The job portal will store public and 'half-public' (i.e. accessible only to a selected circle of persons, as defined by the user) data. The job portal needs at least a minimum set of data to provide a matching/searching function to the different users of the service. Such public and half-public data could be stored unencrypted at the job portal, which implements the access control to this information. Any further data (e.g. high-sensitive data like references from former employers, etc.) could be stored at the user's device. The user then has full control on the access permissions to this information. This could be implemented in an additional (vertical) authentication, access request, access approval cycle between the user and HR/trainers directly. The job portal will then only be used to manage and mediate between the entities while they establish different trust relationships amongst each other, building networks of trust relationships without direct involvement of the job portal.

In the evaluation we will focus on the presented implementation variant, which requires all participants to establish and maintain a trust relationship with the job portal to store and protect the privacy sensitive data in the eCV.

6.1.3.1 Evaluation of OpenID/OAuth for Inhouse Platform Solution

The evaluation of the presented solution will be driven by the requirements defined in Chapter 3. The OpenID/OAuth protocols introduce no intrinsic support for role concepts as defined in criteria R.1 and R.2. However, OpenID provides the option for identification of the acting parties. Thus the application of specific roles to singular identities can be performed independently by the job portal user management. Authentication for the identities is required and thus R.5 can be met adequately. The integrity of the data must be assured by the job portal storage subsystem. The OpenID/OAuth protocols allow to set up secure transaction sessions to protect the eCV data during the transport from the user's computer to the job platform. While the presented solution does not require the use of signatures on the provided data, such an extension is possible to provide integrity protection (R.3) and verifiability (R.4) for the stored data. Thus requirements from criteria class R can be met by additional processes for the division and assignment of roles to the authenticated identities and by the application of additional security features such as digital signatures on the provided data.

The combination of OpenID and OAuth allows the user to define access control policies in a user-friendly way. The user can decide which persons, or groups of persons will have access to which partial information. OAuth tokens can be bound to single data objects of the eCV instead of binding them to the complete eCV. This would allow for a fine-grained access control scheme under the control of the user. Thus P.1 (Access Control) can be met in an appropriate way. The fine grained access control policies allow the definition of reduced identity information sets (P.2), depending on the target audience. While OAuth allows defining access policies, the data cannot be protected from further dissemination (P.3) if no additional protection mechanisms are applied. The

job portal could be implemented in a way such that the data is revealed after a valid access token is presented. The viewer application of the job portal would then allow viewing the document directly instead of downloading it to the client's computer. Thus having a valid OAuth access token would not allow participants to download the document and distribute it any further without user's notice. However, as a general threat, the change of media (e.g. screenshot, printing the document, etc.) can always counteract on such measures. A way to deal with this attack would be to distinguish between trusted and untrusted sources for the data. Trusted sources would include a proof of authenticity and integrity which is inherently bound to the data itself. The proof must be renewed for every issued instance of the eCV (e.g. by an extension of the OAuth protocol). Thus, when an HR manager receives an eCV, the applicant would have to sign the eCV including a statement on the freshness of the signature (e.g. a time-stamp). A media change would not be able to reproduce this signature so that the receiver of the data would be able to see privacy related information, but since this information is not signed it would be considered untrusted. As untrusted data can be produced by anyone, it can be considered less privacy sensitive since it contains no proof that the included information is indeed true. Trusted data should then be the base for essential decisions. While OpenID/OAuth allows to establish secure sessions to the job portal for the protection of confidentiality (P.5), the job portal system cannot be prevented from keeping a log of all job request searches or offers. Companies could restrict access to their published job offers in the same way in which users restrict access to their data in order to prevent profiling by rival firms.

The manageability support using OpenID/OAuth lies in its scalability (M.4). New users are easily added and users can adopt fast and without additional training to the system since standard interfaces such as web browsers and web pages are used (M.5).

The integration into existing infrastructures and protocols is straightforward, if web based solutions are already used (U.1). The management of access control policies by the user using OAuth would be via a web interface, allowing him to easily define policies on the data (U.2).

Since the protocols are open source and easy to integrate into existing infrastructures, the expected costs for the operator of the job portal are kept to a minimum. This allows the operator of the job portal to develop business models which go beyond webpage advertising and marketing. It is possible to charge a small fee for companies to present job offers or for users to publish eCVs while an additional charge could be implemented for the automatic search and match feature. However, it must be kept in mind that such a system relies on a large user base on both sides, both job requests and offers, so the fees should be kept at a minimum. Since the protocols allow for a relatively cost-effective implementation, small fees can be applied.

6.1.4 Classic Remote Login

The TLS protocol ensures that the entities communicating with each other are authenticated and furthermore it provides a means for secure communication through encryption techniques. The RBAC authentication system requires a matching combination of private and public keys (keypair) which authenticates the server to the client via the SSL/TLS protocol. Since the matching public key is available to the client, it can check to see if the server can be trusted before sending it passwords. Furthermore, when using the standard client authentication mechanism of SSL/TLS, the information in the X.509 certificate can be used to look up the user name in the RBAC database. These aspects of the protocol would enable it to fulfill two of the criteria in the evaluation section on Support of Role Concepts, namely Criteria R.5 on Authentication and Criteria R.3 on Integrity.

6.2 Suitability of the Protocols as IdMIP

So, how can the protocols and standards introduced in Chapter 4 can fill the privacy gaps raised in architectures like the job portal? The first observation that can be made about the protocols OpenID, OAuth, Liberty, Anonymous Credentials and SSL/TLS is that they all more or less have distinct focuses and complexities. The mapping of the protocols to the job portal scenario in Chapter 5 showed that all the protocols enrich the scenario in different points. None of the protocols can be claimed as the ultimate candidate for the implementation of privacy-enhanced IdM infrastructures. The evaluation in Chapter 6 showed that all the proposed candidates can enrich the architecture in different aspects. Only Liberty and OpenID/OAuth can be regarded as concurrent protocols. But still these technologies differ in their complexity and their level of abstraction. Both technologies offer means for single-sign-on and cross-domain resource sharing. But OpenID and OAuth definitely have less complexity as they combine architecture design and implementation in one solution. Compared to this, Liberty Technology has a higher complexity, as it consists of several web service architecture frameworks that build the basis for further higher-level IdM and privacy functionality. As seen in section 5.2 the Liberty Framework allow very complex scenarios with a lot of scope for development. The possibilities for building "circles of trust", which require the establishment of business agreements and access control policies for user data is very helpful in scenarios where different service providers are involved. The drawback of Liberty is obvious. The deployment of Liberty-enabled architectures can be very complex, as it consists of a set of web service frameworks. This requires that appropriate implementations have to be made. On the other hand, OpenID technology does not offer as much possibilities as Liberty. The criteria on which one can decide which of the technologies should be used should be the complexity and the dimension of the architecture that is going to be implemented.

Summarizing the statement of this work is that the proposed protocols indeed can be used as IdM infrastructure protocols. But as the evaluation of each of them showed, none of these protocols can be the single solution. Instead, suitable protocols have to be used together in order to reach a higher privacy in such IdM infrastructures. This arises from the fact that these protocols are partly positioned on different technical layers and have differing functionalities. As shown in section 5.1 the protocols can be combined. One possibility would be to use SSO, identity federation and similar services at the general architecture layer and Anonymous Credentials as assertions instead of SAML token. Solely the use of Anonymous Credentials would not have the same power as Liberty regarding the establishment and management of business relationships for an effective IdM. Additionally, SSL/TLS will not be replaceable by any of the other protocols. TLS provides confidentiality and integrity on the communication layer. This should be regarded as an obligatory element of IdMIP. Furthermore, this protocol is established very well.

The result of this evaluation is that none of the proposed protocols can fulfil all of the requirements defined in Chapter 3, but nevertheless if we regard identity management infrastructure protocols as sets of adequate IdM-, privacy- or security protocols, then privacy-enhancement can be reached for infrastructures. Possible obstacles in thate are lacks in interoperability. If several protocols shall work together, they have to be interoperable. This will only be the case if the protocols offer appropriate interfaces and are extendable. The combination of Liberty, Anonymous Credentials and SSL/TLS would be such an example.

Chapter 7

Conclusion

In this document we identified obstacles for the establishment of privacy enhancing identity management protocols in service oriented infrastructures, where processing identity related information is essential. We investigated the requirements of such architectures in Chapter 3. Another aim was to find out how far these requirements can be fulfilled by already existing privacy- or IdM-related protocols. Such an investigation is helpful in identifying key problems of current solutions and further could help in the development of new standards and protocols. Therefore, in Chapter 4 we introduced a series of such protocols. In order to be able to make statements about the applicability of these protocols to SOA, we examined them in a scenario of a job portal, which has several parties involved and raises privacy risks because of the sensitive data that is processed in such a system.

The evaluation showed that none of the proposed protocols can fulfil all of the criteria that are important for privacy-enhanced IdM infrastructures. Most of them differ in their properties, functionalities and complexities. But this fact showed that the criteria can be fulfilled if several protocols with different potentials are combined. Thus, identity management infrastructure protocols would then be defined as a set of adequate protocols that cover different aspects, quasi as a separate layer of the infrastructure. Decisions on which IdMIP to use would then be based on decisions on which protocol should be used on which layer and for what purpose. The comparison of OpenID/OAuth and Liberty showed that there also are common functionalities across different protocols. In such cases there has to be made a decision based on the requirements of the architecture to be realized. In our example we proposed OpenID/OAuth for simpler architectures and Liberty for more complex architectures where the arrangement of business agreements between several service provider is essential.

This document does not cover an investigation of the extent of interoperability possible between the different IdM protocols. Further work could try to answer this question. As a result a final statement could be made whether existing protocols are able to serve as IdMIP components or whether new protocols and standards need to be developed.

As an outlook, the job portal scenario of this paper will also provide valid reference for other applications of the various IdM solutions in markets beyond job search and headhunting. For example, the metaphor of the eCV scenario can be referred to mobile Banking and mobile Travel or to Customer Loyalty Programs and mobile Content Management. Essentially, all Service Provider – User interactions are an exchange of privacy-relevant data (as in an eCV): In any case

of service provisioning, the track record of an entire Service Provider – User relationship is a "curriculum vitae" of that interaction. Hence, the example of the eCV will surely prove applicable for the wider scope of an interaction between a user (e.g. via a privacy-enabled mobile device) and a Service Provider and Storage entity (e.g. a Trusted Third Party or a backend system of any kind). And the logics developed in this paper can lead to modelling technical and business-related scenarios in other fields as well.

References

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, EUROCRYPT 2001, volume 2045 of Lecture Notes in Computer Science, pages 93–118. Springer Verlag, 2001.

[LA03] Liberty Alliance Project, Introduction to the Liberty Alliance Identity Architecture, Revision 1.0, March 2003

[LA05] Liberty Alliance Project, Liberty ID-FF Architecture Overview, Version 1.2-errata-v1.0, January 2005
http://projectliberty.org/liberty/content/download/318/2366/file/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf

[LA07] Liberty Alliance Project, Liberty ID-WSF Web Services Framework Overview, Version 2.0, August 2006,
http://projectliberty.org/liberty/content/download/889/6243/file/liberty-idwsf-overview-v2.0.pdf

[LA08] Liberty Alliance Project, Liberty Alliance Web Services Framework: A Technical Overview, Version 1.0, February 2008,
http://www.projectliberty.org/liberty/content/download/4120/27687/file/idwsf-intro-v1.0.pdf

[LT03] Liberty Alliance Project, Liberty Specs Tutorial, Version 2, October 2003,
http://projectliberty.org/liberty/content/download/423/2832/file/tutorialv2.pdf

[PH09] PrimeLife, Heartbeat H6.3.1, Requirement for privacy-enhancing Service-oriented Architectures, February 2009

[PL08] PrimeLife, Deliverable D3.3.1 / D3.4.1, First Report on Standardisation and Interoperability, May 2008

[PL09] PrimeLife, Deliverable D2.1.1, First Report on Mechanisms, February 2009

[PH09] PrimeLife, Deliverable D4.1.1, HCI Research Report – Version 1, February 2009

[TG09] 3G Americas, Identity Management – Overview of Standards & Technologies for Mobile and Fixed Internet, January 2009,
http://www.3gamericas.org/index.cfm?fuseaction=register&goto=http://3gamericas.org/documents/3GAmericas_Unified_Identity_Management_Jan2009.pdf

Appendix A

Linked Requirements of H6.3.1

A.1 Core Policy Requirements

Requirement No. 1 Policies should be available in an unambiguous formalization. Thereby, the content of policies should be machine interpretable.

Policies are used by service providers to describe restrictions on the processing of personal data. From a privacy point of view, policies on purpose limitation, non-disclosure and data retention period are of major importance. Since policies should be available for automatic processing and comparison with user preferences, they have to be available in a machine-interpretable form. To avoid misinterpretation of policies and thus reduce legal conflicts, unambiguity in the normalisation is necessary. However, there may be cases, where policies given in natural language cannot be transformed into such strict machine-interpretable form.

Requirement No. 2 It must be ensured that communicated policies cannot be argued by the ensuring entity.

Policies must be binding, i.e. the ensuring entity must not be able to argue their existence and exact content. This requirement can be met by using electronic signatures.

Requirement No. 3 Policies must be easily accessible to users. The way of accessing the policies should be determined by a clear specification.

Potential users of a service should be able to see the policies of every service provider without troubles. A standardised means of access could be made available, but should only provide as little communication as possible, to limit the amount of data made available through this communication.

Requirement No. 4 Policies should be presented to users in an easily comprehensible manner.

As policies can be very complex, users that do not have detailed legal knowledge might not be able to understand and assess them. Thus, policies should be described in a manner that is easily comprehensible to the general public.

Requirement No. 5 It must be explicitly determined who is responsible for the policy, including a reference to the applicable jurisdiction.

This determination must be visible for users.

Requirement No. 6 It must be explicitly determined what data are covered by a policy. This determination must be visible for users.

A clear link between data and policy is needed, since different services of one provider may give varying policies, respectively for different parts of one set of data. This is an effect of data separation. It is advisable to communicate policies for each purpose separately.

Requirement No. 7 Policies should cover all aspects of data processing with regard to privacy legislation.

Policies can be of arbitrary detailedness. In order to prevent unnecessary complexity, they should not be more detailed than legally / contractually necessary.

Requirement No. 8 Recipients or categories of recipients to which the data will be passed on to, must be explicitly determined. This must include a reference to the applicable jurisdiction for the recipient.

Requirement No. 9 It should be explicitly determined under what policies data is passed on to other parties.

If personal data is passed down a service chain, the receiving service provider is legally bound in regard to what it may do with this data. As this may be only a subset of what the originating service may do, this should be reflected in a separate (derived) policy.

A.2 Privacy Logging Requirements

Requirement No. 13 The fact that data are logged must be visible to the user.

When logging the processing of personal data, these logs themselves often are personal data. This information needs to be visible to the user as part of the transparency principle. The user must be informed of the fact that logging is applied, and information on the specific logs may be in scope for subject access requests.

A.3 Requirements on access to primary information

Requirement No. 19 Access to personal information should be provided in an unambiguous formalisation. The content of the information should be machine interpretable.

If users of a service shall be able to analyse accessed information in a partly automated manner, a machine interpretable formalisation is indispensable. In order to on the one hand avoid misinterpretation of accessed information and on the other hand prevent possible legal disputes about differently interpreted information, unambiguousness of formalisation is required.

Requirement No. 21 A simple methodology with regard to request and granting of access to information should be provided to users.

Users of a service should be enabled to easily get access to the information they provided. For this a standardised procedure should be used, so the efforts for such accesses are kept low on both

sides. Through standardised clauses an automation of the process - at least partially - could be feasible.

Requirement No. 23 Accessed information should cover only contractual or further legally relevant aspects of data processing.

Service providers are legally obliged to grant users access to specific information (see requirement 17). In principle, the accessible information provided should be limited to this specific information in order to avoid too much complexity.

A.4 Cross-Domain-specific Requirements

Requirement No. 25 It must be possible to maintain communicated policies even if the Service Oriented Architecture is dynamically adapted (refers to the constellation of a SOA being established by several entities).

It may happen that a member of a Service Architecture leaves the organisation and is replaced by another entity. Dynamic changes of this kind should be possible without resulting in the need to negotiate policies once again with customers or even in the necessity to terminate contracts with customers. This requirement does not apply to the virtual organisation: The formalization of policies e.g. may not restrict replacement of enterprises and their services during their run-time.

A.5 Requirements for additional mechanisms

Requirement No. 33 It should be made easy for users to exercise their rights of correction, erasure and blocking.

As correction, erasure and blocking are instances that are initiated by the user, technical feasibility as such is not sufficient as requirement. Rather, it shall be smoothly possible for the user to exercise his rights.

Requirement No. 36 The user shall have the possibility to express her preferences in an easy manner.

As users quite often are technical and legal amateurs, tools enabling them to express their preferences in a formalized manner (as defined by requirement 1) should be made available to them. These tools should be easy to use. The preferences can be the basis of a partly automated negotiation of new policies.