User Configurable Privacy for Authentication and Authorization in a Federation

(Concept paper)

Author	Moritz Theile (moritz@melcoe.mq.edu.au)
Version	0.82
Date	20th January 2006
Reviewed by	

Abstract

The Internet was built without a way to know who and what someone is connecting to [03]. It misses something that can be used for identification like a driver license or passport in the real world. One approach to overcome this shortcoming is implemented by Shibboleth™ [04].

Shibboleth is an open source software based on the OASIS' Security Assertion Markup Language (SAML) [05] standard that gives individuals an identity on the internet which respects the individual's privacy concerns. The section in the Internet that accepts this identity is called a ***Federation****, and mainly consists of* ***Identity Providers*** *(IdP) and* ***Service Providers*** *(SP).*

When a user wants to access a resource on the internet that is provided by a SP, the SP asks an IdP for information about that user. Based on this information access to the resource is granted or denied.

This paper describes how an individual can view and control the personal information that is revealed by his Identity Provider to a Service Provider. It is important to make the handling of privacy in the Federation transparent and controllable for individuals to ensure their acceptance and therefore the usage of Federation Services.

Contents

INTRODUCTION **1**
AUTOGRAPH **3**
THE USER INTERFACE OF AUTOGRAPH 4
AUTOGRAPH IN THE SSO SEQUENCE 5
PICTURE GALLERY EXAMPLE **5**
THE 'PICTURE GALLERY' SERVICE 5
CONFIGURING THE IDCARD FOR THE 'PICTURE GALLERY' SERVICE 6
INTEGRATION WITH SHIBBOLETH **9**
EMBEDDING AUTOGRAPH IN THE SSO PROFILE 9
RELEASING THE OPTIMAL ATTRIBUTE SET 10
ARCHITECTURE **11**
REQUIRED COMPONENTS 11
OPTIONAL EXTENSIONS 13
Attribute Mapping *14*
CONCLUSION **15**
REFERENCES **16**
APPENDIX A **17**
REQUIREMENTS 17
APPENDIX B **17**

Introduction

Shibboleth in specific and SAML in general are widely used in Higher Education for extending single sign-on (SSO) to include not only their own institution's resources, but also external universities' resources. Examples are the InQueue [06] and InCommon [07] federation in the USA, Switch in Switzerland [08], and MAMS in Australia [09].

Shibboleth requires "… that control of attribute release to service providers be available to both administrators and principals. Therefore, a Shibboleth attribute authority MUST have the ability to authenticate requests and MUST implement some form of access control governing the release of specific attributes and values belonging to specific principals to specific requesting service providers. Subject to that constraint, any access control mechanism may be supported." ([2] l.153-157)

Exactly this requirement for a configurable **attribute release mechanism** will be targeted with the approach discussed in this paper; in particular the attribute release mechanism for principals. Attributes contain information about principals (**IdP members**) (E.g. givenName="Sue"). By controlling the release of this identity **attributes,** IdP members can determine how much privacy they reveal. The requirement to reveal private information at all is important, because information is often needed for authorization.

One example for this in the real world is that people have to be older than 18 to buy alcohol. If someone does not want to reveal his age by showing his driver license, he cannot buy alcohol. One says the **identity requirements** are not met.

There is one shortcoming in the way we identify ourselves in the real world. It is not possible to hide some of your identification attributes. When someone shows his driver license not only his age but also his name and address is viewable.

This paper introduces a concept that enables IdP member to create a tailor-made idCard for each resource. The **idCard** contains identity attributes that are shown to the SP that hosts the resource the IdP member wants to access. If this would be possible in the real world customers could have an idCard for liquor shops that contain only their age.

The web application which enables the user to configure his Identity Provider to consider his personal privacy concerns is called **Autograph**. It is important to inform the user about the consequences of removing identity attributes from the idCard. For this reason Autograph gives immediately feedback on the features of a Service, which are available with the revealed information. The feature 'buy alcohol' would not be available from the Service 'bottle shop' when the age is not provided. Autograph also informs the user when a service feature is not available because of some missing information. Therefore, the user can make this feature available by revealing the required information.

The concept of Autograph is based on the idea that a Service Provider protects its resources and an Identity Provider protects its member's privacy. In order not to reveal to

much privacy of its members, the Identity Provider needs to know what attributes are required by the Service Provider. This information is contained in the Service Provider description. It is used by the Identity Provider to dynamically determine an attribute set that is tailor-made for each individual (**smart attribute release**).

Current implementations of Shibboleth use Attribute Release files (**ARP** files) to configure the Identity Provider to release only useful attributes. The basic problem with ARP files is that they can't express to release attribute `x` only if attribute `y` has value `a`. But this is required to create an Autograph as introduced in this paper.

Also a Service Provider description is much more intuitive. Instead of defining what to release, administrator can define what's needed for certain resources. Based on this information the Identity Provider can figure out what to release automatically.

A real life scenario using this approach could be as follows:

1. The IdP ‚Gumtree University' makes a legal agreement with the SP ‚University of Art' that states:
    a. All members of the ‚Gumtree University' can search the ‚Picture Gallery' service of the ‚University of Art'.
    b. All staff at ‚Gumtree University' can download hight resolution pictures from the ‚Picture Gallery' service.
2. The IdP ‚Gumtree University' receives (or creates) a Service Provider description that states:
    a. In order to control if the right people are accessing our ‚Picture Gallery' service we need the attributes a, b, c from you.
3. The ‚Gumtree University' adds this Service Provider description to its Service Provider description repository.

That's all that has to happen in order to set up the IdP to release the right attributes.

The Service Provider description in this example is simplified. Normally it would also say, wich attributes with which values are required for which Service Features. This gives the IdP the possibilitiy for a privacy preserving way of releasing attributes. This example is explained in the ‚Picture Gallery Example' chapter in more detail.

Autograph

To explain the Autograph approach the following terms are used. Some of the definitions might be simplified or differ slightly from the definitions in [1] and [2].

A **Service Provider** (SP) offers Services to the members of the IdPs in the Federation. **Services** offer a set of Service Features. **Service Features** are functionalities of a Service. (E.g. search, submit, subscribe newsletter …). Service Features can have **required Attributes**. These can have **required Values**.

SPs can be described in a **Service Provider Description** (SPD) according to the schema in Figure 1.

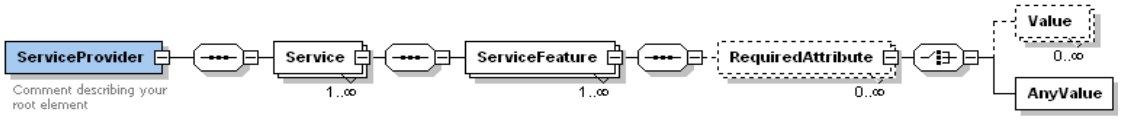

Figure 1 : A Service Provider can be described in an xml instance of this schema.

An **attribute request** is issued from the SP to the IdP to ask for the identity of an IdP member. The request also contains the information which Service the member wants to access. The response contains the identity of that member as a set of attributes. This is the **released attribute set**. The attributes in this set are **released**.

A **Service Feature access decision** can be positive or negative and decides whether a user gets access to a Service Feature. It is based on the required attributes of a Service Feature and on an attribute set. It is positive, if at least one required value of each required attribute matches the value of that attribute in the attribute set. Otherwise it is negative.

Member blocked attributes are not released. They can be unblocked by an IdP member.

The **biggest attribute set** contains all available attributes of an IdP member.

The **biggest releasable attribute set** is the biggest attribute set without member blocked attributes.

An **optimal attribute set** (OAS) is a subset of the biggest releasable attributes set, which gives an IdP member access to a maximal number of Service Features. Additional, no attribute can be removed without loosing access to a Service Feature.

An **available Service Feature** is a Service Feature for which the Service Feature access decision with the released attribute set is positive.

A **reachable Service Feature** is a Service Feature for which the Service Feature access decision with the biggest releasable attribute set is positive.

There are two questions to clarify in order to explain the integration of Autograph in the Shibboleth System from a user point of view: When and how does the IdP member interact with Autograph?

The *when* is explained in the section 'Autograph in the SSO ' and the *how* in the section 'The user interface of Autograph'. The whole process is illustrated in the section 'Picture Gallery Example'.

The user interface of Autograph

The graphical user interface of Autograph (Autograph GUI) has 3 elements.

1.) In the Service selection form a Service can be chosen by an IdP member. Configurations will only affect the released attribute set to the selected Service. The selection field in the form contains only Services with reachable or available Service Features.

2.) The released attributes box or **idCard** shows the released attributes with the values of the IdP member.

3.) The Service Feature list shows the reachable and available Service Features.

The released attributes are represented as an idCard card. This idCard metaphor is the fundamental concept of the Autograph GUI. The idCard will be shown to the Service Provider in order to get access to one or more Service Features of the selected Service. By looking at the idCard, a user is instantly informed what information will be revealed when she visits the selected Service. There is one idCard per Service.

The idCard provides the functionality for the IdP member to remove attributes. When an attribute is removed from the idCard it becomes a member blocked attribute. Therefore, an available Service Feature might change to a reachable Service Level.

The Service Feature list marks available Service Features with a green and reachable Service Features with a red background. The reachable Service Features contain also a button to add all required attributes to the idCard. Thus, clicking the button makes the Service Feature available.

The user interface in action is described with an example in section 'Picture Gallery Example'.

Autograph in the SSO sequence

Autograph can always be accessed by an IdP member at a certain URL to create her personal idCard for a Service.

However, it is important that Autograph is presented to the IdP member when she visits a Service the first time. In this way, she is informed about the attributes that will be released to the SP and she also can configure her idCard. (see requirement 8) (also see Appendix B)

To accomplish this behaviour Autograph has to be embedded in the single sign-on sequence. That is, Shibboleth starts a sequence of interactions between the SP that hosts the Service when an IdP member calls a Service and the IdP so that the SP can identify the principal. There are different scenarios for different situations like principal is or is not logged in.

So, when an IdP member calls a Service he might be forwarded to his IdP using Shibboleth WAYF service. He is prompted for username and password. After that the Autograph GUI shows up. The Service the IdP member wants to access is already preselected. The IdP member can now configure her idCard for the Service she wants to visit.

When the IdP member is already logged in, that means a Shibboleth session is established, and calls a Service, Autograph shows up on his browser immediately, omitting the WAYF and log-in process.

Picture Gallery Example

This chapter illustrates the concept with a detailed example.

The 'Picture Gallery' Service

A SP with the name 'University of Art' offers a picture gallery Service. This service has the Service Features 'search' and 'download'. To get access to the 'search' Service Feature the 'community' attribute with any value is required. (The community attribute may contain values like staff, student or physics.) The Service Feature 'download' requires the attribute 'community' with the value 'Staff' and the attributes 'givenname' and 'surname' with any value. The resulting SPD is shown in Listing 1.

Listing 1: The Service Provider Description of the University of Art. (This is a simplified xml document for demonstration purposes only.) It is an instance of the schema in **Figure 1**.

```
<ServiceProvider name="University of Art"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SPD.xsd">
     <Service name="PictureGallery">
          <ServiceFeature name="search">
               <RequiredAttribute name="community">
                    <AnyValue/>
               </RequiredAttribute>
          </ServiceFeature>
          <ServiceFeature name="download">
               <RequiredAttribute name="community">
                    <Value>Staff</Value>
               </RequiredAttribute>
               <RequiredAttribute name="givenname">
                    <AnyValue/>
               </RequiredAttribute>
               <RequiredAttribute name="surname">
                    <AnyValue/>
               </RequiredAttribute>
          </ServiceFeature>
     </Service>
</ServiceProvider>
```

For the ‘Picture Gallery’ Service the optimal attribute set to release depends on the user. For a student it would only contain the ‘community’ attribute. It is not necessary to send the name attributes, because the Service Feature ‘download’ is not available for students anyways. For staff members the optimal attribute set would contain the name attributes, too, because they get access to the ‘download’ Service Feature which requires them.

If the ‘surname’ attribute would be blocked by one specific staff member, this member couldn’t get access to the ‘download’ service level. Therefore it wouldn’t make sense to send the ‘givenname’ attribute. However, the staff member could get access to the ‘search’ Service Feature. The optimal attribute set would contain the ‘community’ attribute only.

How these facts are represented in Autograph is demonstrated in the following section with screenshots of an Autograph prototype.

Configuring the idCard for the ‘Picture Gallery’ Service

The staff member Hans configures his idCard:

Hans accesses the Service ‘Picture Gallery’ the first time. He types in the URL in his browser and hits return. He is forwarded to the WAYF Service. He chooses his IdP. The IdP prompts him for his username and password. After he commits, the Autograph GUI shows up like in Figure 2.

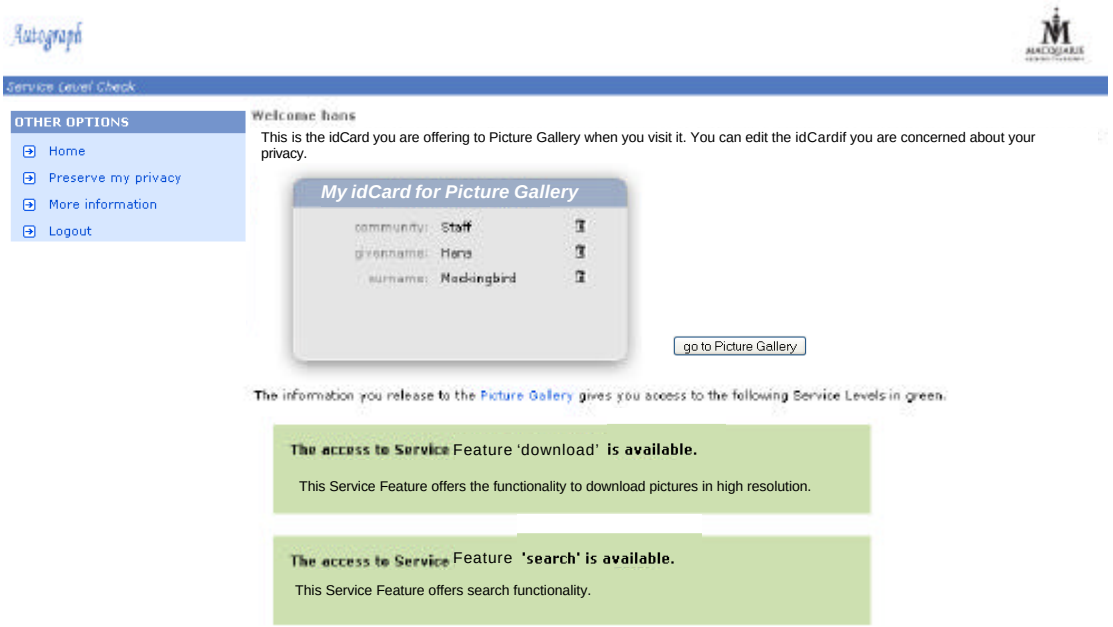

Figure 2: Hans releases his attributes 'community', 'givenname' and 'surname'. These attributes give him access the Service Features 'download' and 'search'.

Hans sees his surname on the idCard, but doesn't want it to be released. After he clicked the delete symbol in the line of the surname attribute, he sees the screen as shown in Figure 3. Besides his surname also the attribute 'givenname' disappeared from the idCard. This is because the Service Feature 'download' is without the 'surname' attribute not available any longer. And since no other Service Feature requires the given name, there is no need to release it.

In other words, the idCard shows always the optimal attribute set. According to the definition, in this set there are no attributes that don't give additional access to Service Features allowed. Consequently, Hans reveals always the minimal amount of privacy for the Service Features he wants to access.

In Figure 3 the Service Feature 'download' has a link in its red box. If Hans would click on it, all missing required attributes for this Service Feature would be added to the idCard. Thus, the Service Feature 'download' would become available again. Hans would see the window as in Figure 2 again.

Hans feels comfortable with the thought that only his community will be revealed when he visits the 'Picture Gallery' and clicks on the 'go to Picture Gallery' button. This brings him to the 'Picture Gallery'. The 'Picture Gallery' Service grants him only 'search' rights, because it doesn't know his name.

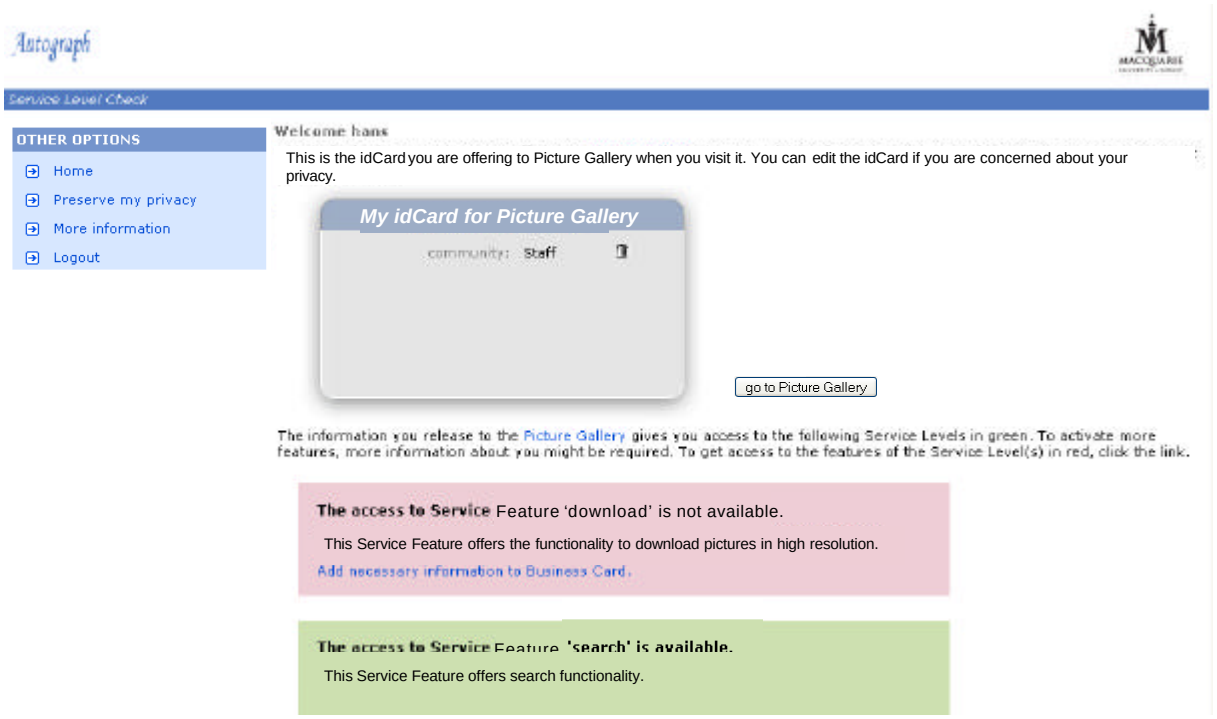

Figure 3: In this screenshot the Service Feature 'download' is not available. By clicking the link in the red box, the required attributes of this Service Feature would be added to the idCard, and the Service Feature would become available.

The student Sue configures her idCard:

Student Sue accesses Autograph by entering the url of Autograph directly. She authenticates herself and chooses the Service 'Picture Gallery', she sees on the idCard only her 'community' attribute (see Figure 4). The Service Feature list shows only the 'search' Service Feature, because the 'download' Service Feature is not reachable for her. It requires the value 'Staff' in the 'community' attribute which she hasn't.

Sue's options are limited in this case. If she removed the 'community' attribute from the idCard she wouldn't have access to any of the Service Features any more.

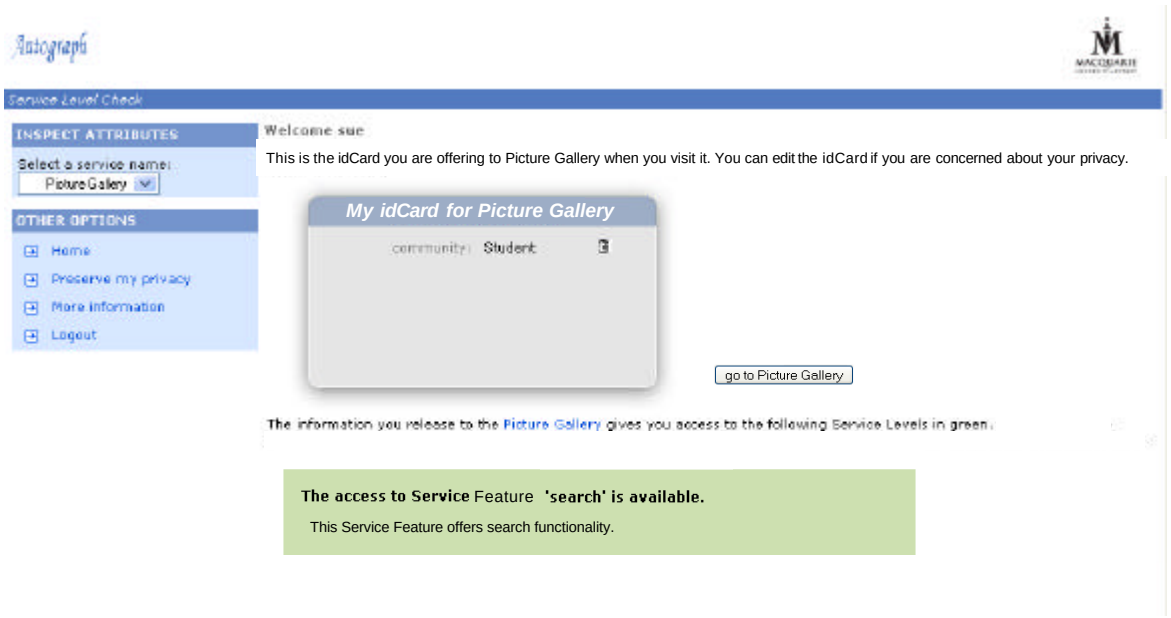

Figure 4: When the student Sue logs in, the Service Feature 'download' is not listed, because it is not reachable for her. She couldn't fulfil the requirement of the 'download' Service Feature that the community has to have the value 'Staff'.

Integration with Shibboleth

In order to implement the introduced functionality it is necessary to add the required functionality in the "right spots" of the Shibboleth System. This chapter tries to identify these spots.

Embedding Autograph in the SSO profile

The chapter 'Autograph in the SSO sequence' explains the integration of Autograph in the SSO profile from a user point of view. This section explains how this could be done technically.

Shibboleth defines a number of SSO profiles which all have to play together with Autograph. Basically there is the Browser/POST and the Browser/Artifact profile. Both of them can be with or without WAYF and Attribute Exchange [01]. Further variations may appear when a security context between IdP and browser is already established.

Important for the integration of Autograph is that all these profiles have some characteristic steps in common:

When an IdP member wants to access a SP the SP points the browser via redirect to the IdP. After the IdP executed the authentication it returns some kind of identifier for this authentication to the SP.

Exactly before the IdP returns an identifier, Autograph with pre selected SP can be presented to the IdP member to view and configure his idCard for the SP (see Figure 5). After he is finished the SSO process continues exactly the same way as specified for Shibboleth.

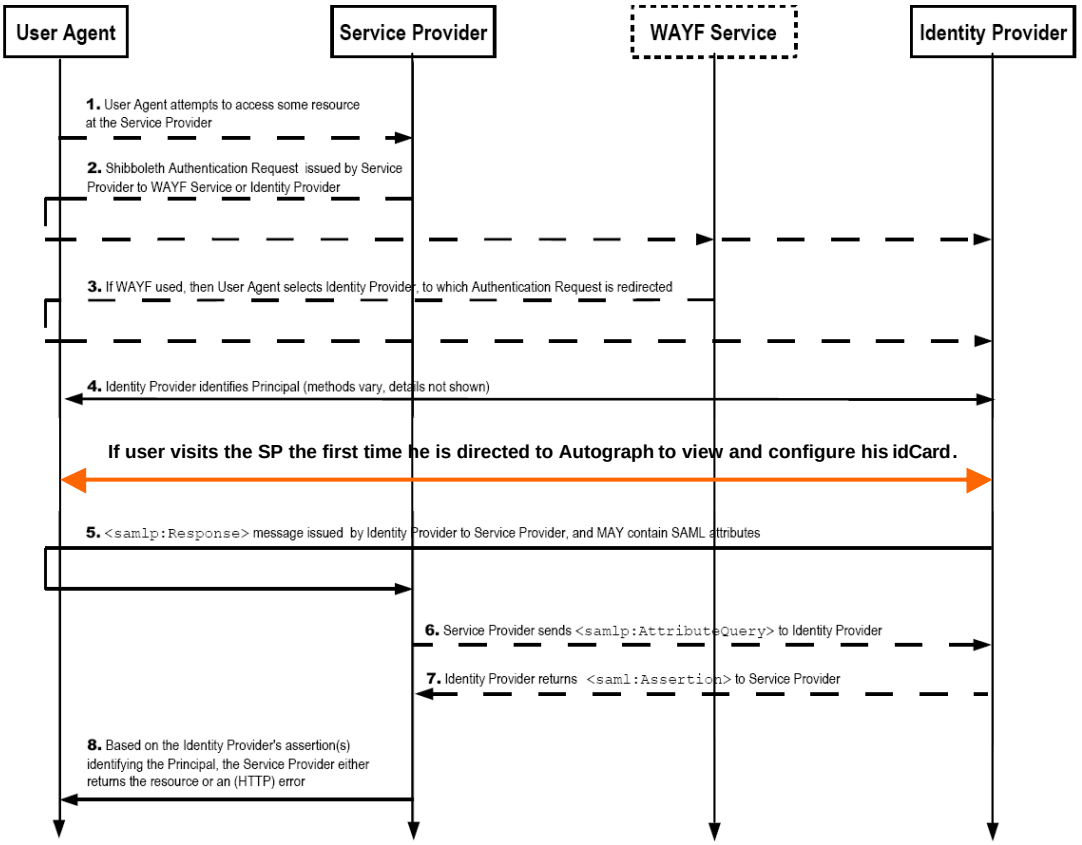

Figure 5: The idCard is viewed and configured in Autograph by the IdP member before the <samlp:Response/> is sent.

Releasing the optimal attribute set

The Attribute Authority is part of the IdP and is, among others, responsible for releasing attributes. The definition in the specification is:

"An **Attribute Authority** processes attribute requests; that is, it issues attribute assertions. The attribute authority authenticates and authorizes any request it receives." ([1] l.145-146)

The Attribute Authority provides the IdP member attributes. Between the attribute request and the response the attributes are retrieved and returned as an attribute assertion.

The attributes that are retrieved have to be the optimal attribute set in order to follow the concept shown in this paper.

The specification doesn't say which attributes to release. Thus, an IdP that releases the optimal attribute set is Shibboleth compliant.

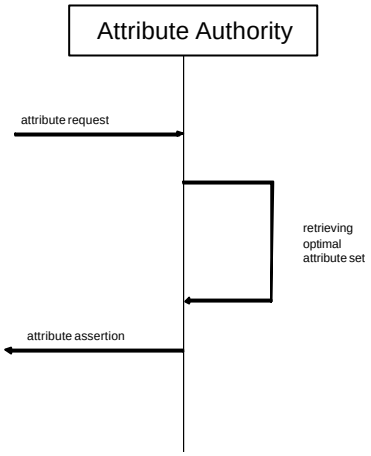

Figure 6: To follow the concept shown in this paper, the Attribute Authority has to create an attribute assertion containing the attributes of the optimal attribute set.

Architecture

This chapter is divided in two sections. The first one introduces the components that are required to implement the Autograph and optimal attribute set (OAS) concept. The second section outlines how the architecture can be extended to meet additional requirements like attribute mapping. This is shown in a very rudimentary level to demonstrate that the concept of Autograph and optimal attribute set is compliant with attribute mapping.

Required components

Figure 7 contains all required components for realizing the Autograph and the OAS concept. The components are described in the following section.

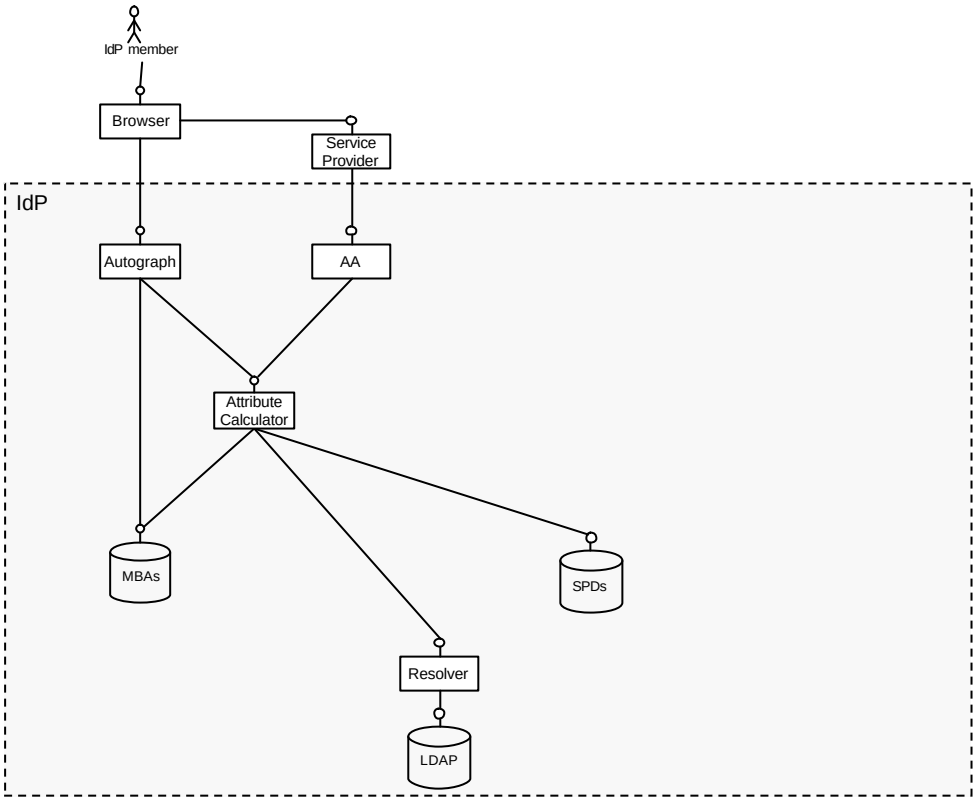

Figure 7

AA

The AA is the Attribute Authority and the only Shibboleth component (besides the IdP) shown in Figure 7 that is described in [1] or [2].

Attribute Calculator

The Attribute Calculator is the connecting element between SPDs, UBAs and the IdP member attributes. It provides the functionalities as shown in Listing 2.

Listing 2

```
public interface AttributeCalculator {

//    Retrieving the optimal attribute set
      Set getOptimalAttributeSet(String service, String member);

//    retrieving the reachable Service Features
      Set getReachableServiceFeatures(String service, String member);
```

```
//    retrieving the available Service Features
      Set getAvailableServiceFeatures(String service, String member);

//    retrieving the Services with available or reachable
//    Service Features for this member.
      Set getServices(String member);
}
```

MBA Repository

MBA Repository stands for member blocked attributes repository. It contains information on which member blocked which attribute for which service. It can be described with the schema in Figure 8.

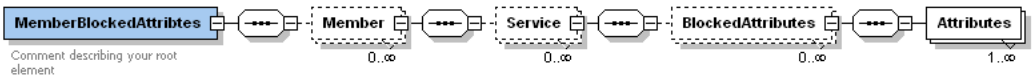

Figure 8: This schema is supposed to describe the content of a MBA Repository. It shall not imply that the repository is realized as a XML file.

SPD Repository

The SPD Repository contains all Service Provider Descriptions. SPDs are needed to retrieve the available and reachable Service Features to display in Autograph. It is also needed to find the optimal attribute set. If a SPD is not available for a SP a default SPD is generated.

Resolver and LDAP Repository

The Resolver and LDAP Repository is the access point to user specific attributes. The Resolver contains the information where to look for certain attributes. The LDAP Repository contains the attribute values for the IdP members.

Optional extensions

The architectural overview in Figure 9 contains two extensions of the IdP. The SPD updater is triggered in a certain time interval to update the local SPD repository.

The idea of attribute mapping is more complex and therefore explained in its own section.

Attribute Mapping

Data of institution members is normally saved in a LDAP database. This database contains for each member an object with different attributes. The naming and meaning of those attributes is specified in so called directory schemas. Unfortunately there are different directory schemas in use. This fact makes mapping necessary if institutions with different directory schemas want to exchange information of their members.

Attribute Mapping is the process of creating a value for an attribute based on local available information.

To implement a mapping mechanism the following components can be added:

Mapping Repository

The Mapping Repository contains the information which Mapping is to be used for which attribute and for which Service Provider.

Mapping Editor

With the Mapping Editor the Mappings in the Mappings Repository can be edited.

Mapper

The Mapper creates the attribute value for a certain Service Provider, IdP member and attribute:

```
String getValue(String attributeId, String memberId, String
serviceProviderId)
```

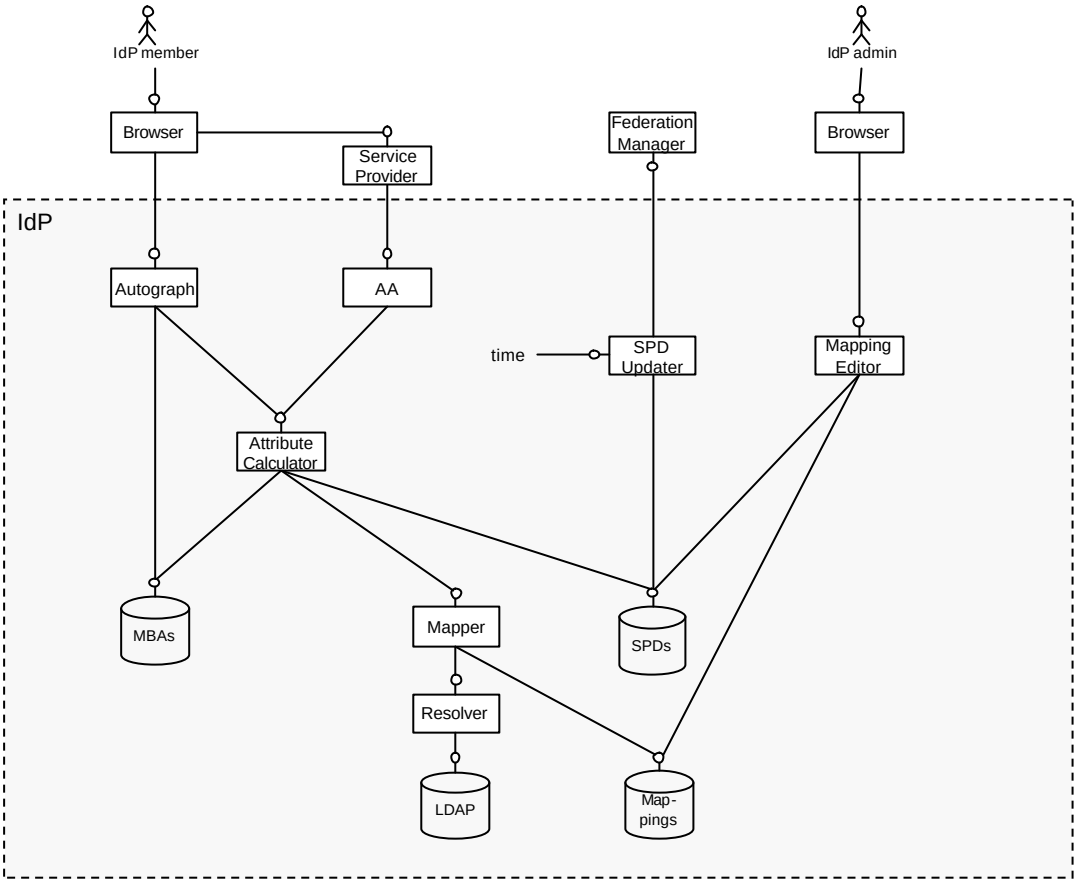

Figure 9

With these components the IdP admins are able to match their local directory schemas to directory schemas used by Service Provider.

Conclusion

The Autograph and the optimal attribute set concept is compliant to the Shibboleth specification. Autograph provides IdP members an intuitive way to protect their privacy. The OAS concept guarantees that only required attributes are released. The administrative work can be reduced to a minimum by retrieving Service Provider descriptions from a central repository in the Federation.

References

[01] Shibboleth Architecture, Technical Overview, Working Draft 02, 8 June 2005

[02] Shibboleth Architecture, Protocols and Profiles, 10 September 2005

[03] Kim Cameron, Architect of Identity, Microsoft Corporation

[04] http://www. shibboleth.internet2.edu

[05] http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[06] http://inqueue.internet2.edu/

[07] http://www.incommonfederation.org/

[08] http://www.switch.ch/edu/misc.html

[09] https://mams.melcoe.mq.edu.au/zope/mams

[10] Microsoft, A Guido to Integrating with InfoCard v1.0

Appendix A

Requirements

Autograph fulfills the following requirements:

1.) The IdP shall protect the privacy of its members as good as possible. That means it shall always release the optimal attribute set.

2.) Autograph shall enable IdP members to view the attributes released to a specific SP.

3.) Autograph shall enable IdP members to block attributes for a specific SP.

4.) Autograph shall enable IdP members to view the reachable Service Features of a Service.

5.) Autograph shall enable IdP members to only view the available Service Features of a Service.

6.) Autograph shall enable IdP members to make reachable Service Features available.

7.) Autograph shall provide a self explaining, intuitive user interface.

8.) The IdP member is directed to the Autograph application before he visits the Service Provider the first time.

9.) The IdP member is able to access Autograph independently from visiting SPs.

Appendix B

The question this appendix takes care of is: Which attributes are released by default? There might be IdPs that don't want to have the Autograph GUI in the login process, so the IdP member won't configure the idCard that is shown to the SP. In this case a default configuration is very important.

There are two possibilities that are easy to accomplish: Either releasing all attributes or blocking all attributes. Blocking all attributes without using Autograph makes no sense because no Service Feature would be accessible. Releasing all attributes on the other hand would be very careless.

It would be nice to have a possibility to have the functionality to assign attributes to certain privacy levels. They could be low, medium and high. With this information the IdP could be configured to release only attributes up to a certain privacy level. When a Service Feature requires an attribute with high privacy level, but the default privacy level

is set to medium, the attribute wouldn't be released without manually changing the idCard.

With this privacy level information for each attribute it would also be able for an IdP member to set his personal default privacy level.

Appendix C

Integrating Autograph with ARP files

Current implementations of Shibboleth are using ARP files for defining which attributes to release to which Services. It might be inevitable to deal with this legacy mechanism instead of redesigning the affected modules in the Identity Provider for a proper solution of integrating Autograph. This section discusses if this could be done and how it could be done.

Site ARP

All available SP descriptions are mapped to the site ARP that means:

All attributes that could ever be released to a Service Provider are released in the site ARP to this Service Provider (all Service Features enabling attribute set).

User ARPs

If Autograph wouldn't be used, all attributes would be released because there exists no user ARP, but if a configuration session with Autograph for a specific Service was made a User ARP is created.

The user ARP blocks all attributes that do not belong to the optimal attribute set but would be released to the Service Provider.

The idCard configuration Session

An idCard configuration Session is a process that involves an IdP member, a Service Provider description and a Service.

The main steps in the idCard configuration Session:

1. Optional: The Service Selection field is shown.
    a. Choosing Service
2. The idCard view of the Autograph GUI is shown
    a. Displaying the idCard with all attributes that make all of the default Service Features available.
    b. Displaying the reachable Service Features
    c. Displaying the available Service Features
3. IdP member configures his idCard by
    a. Deleting attributes from the idCard
    b. Making reachable Service Levels available
4. IdP member leaves Autograph.

Under the hood:

Session initialization:

- Map serviceId2serviceDescription = getServices(directory).
- Map attribute2value = SharpeCore.getAttributeMap(user)
- Map serviceId2memberBlockedAttributes = getMemberBlockedAttributes(memberId)

During 1:

- allServiceFeaturesEnablingAttributesSet = getAllServiceFeaturesEnablingAttributeSet(serviceDescription)
- getReleasedAttributeSet

During 2:

- getReachableServiceFeature(serviceDescription, biggestAttributeSet)
- getAvailableServiceFeature(serviceDescription, releasedAttributeSet)
- getAttributesOfServiceFeatures(availableServiceFeatures)

During 3a:

- ?

Conclusion

It is possible to achieve exactly the same attribute release behavior under one condition: For each IdP member that uses a Service has a user ARP to be existent. Otherwise the all Service Feature enabling attribute set is released.