An Architecture for Privacy-Aware Inter-domain Identity Management

Wolfgang Hommel

Munich Network Management Team,
Leibniz Supercomputing Center Munich
hommel@lrz.de

Abstract. The management of service oriented architectures demands an efficient control of service users and their authorizations. Similar to structured cabling in LANs, Identity & Access Management systems have proven to be important components of organizations' IT infrastructures. Yet, due to new management challenges such as virtual organizations, on-demand computing and the integration of third party services through composition, identity information has to be passed to external service providers; this decentralization inherently leads to interoperability and privacy issues, which existing management standards are not dealing with appropriately yet. We present an architecture, based on SAML, XACML and XSLT, which provides a tight integration of cross-organizational identity data transfer into the local provisioning business processes along with a policy-driven inter-domain privacy management system, and its implementation.

1 Introduction and Problem Statement

Besides network components and systems, the operation of complex IT infrastructures more and more has to focus on the management of application-level services offered to end users. An essential part of service provisioning is the setup, configuration, maintenance and deletion of user accounts, also known as Identity & Access Management (I&AM). The I&AM paradigm demands to provide a holistic view of a user instead of administrating each account on each service independently. Typically, a central identity repository, such as an LDAP-based enterprise directory, provides the user data required for authentication, authorization and accounting, as well as for service personalization. I&AM systems are usually fed by an organization's human resources (HR) system and customer relationship management (CRM) database; they thus contain sensitive data, which must be protected due to privacy and governance aspects.

However, in an increasing number of scenarios, cross-organizational identity data transfer is required. If, for example, IT services are outsourced to third parties, personalization and accounting data must be made available to the service provider (**SP**), as they are required for service provisioning and billing. Similarly, aligning with other organizations to form a virtual organization, e.g. in Grid projects, requires to pool together parts of the resources and user data

J. Schönwälder and J. Serrat (Eds.): DSOM 2005, LNCS 3775, pp. 49–60, 2005.

alike. To avoid redundant and inconsistent storage of identity information, as well as the administrative overhead to acquire and maintain this data multiple times, dedicated languages and web services based management protocols exist for the exchange of identity information. Standards like the Security Assertion Markup Language (SAML, [1]), the Liberty Alliance specifications [2] and the Web Services Federation Language (WS-Federation, [3]) provide methods which allow an SP to retrieve information about a user from the user's so-called Identity Provider (**IDP**). Especially SAML is in wide-spread use, as it has served as basis for Liberty Alliance and because WS-Federation has adopted SAML support meanwhile. The application of these standards to inter-domain service provisioning is also known as Federated Identity Management (**FIM**).

While those standards provide a lot of much needed inter-domain provisioning functionality, we have shown in previous work that they have several deficiencies in common [4]. In this paper, we present solutions for two of the most urgent problems of the current FIM standards and their existing implementations. First, the demand for an identity federation wide common data schema is not considerate of the syntax and semantics of local I&AM solutions and thus makes the seamless integration of FIM into existing provisioning business processes next to impossible in practice. Second, none of the standards specifies how administrators and users can control and restrict which information about a user is allowed to be sent to which provider, as is urgently required to protect the user's privacy.

We address these issues in this paper by extending the standard SAML architecture by two IDP-side components, while still maintaining full SAML compliance. First, we introduce an Attribute Converter component. It translates incoming requests from the federation-wide data schema into the one used by the local I&AM solution; then, it converts outgoing responses back into the federation's data schema. Second, we demonstrate how the eXtensible Access Control Markup Language (XACML, [5]) can be used efficiently to specify which service providers can access which identity information. We have implemented both components prototypically as extensions to the well-known Shibboleth software [6].

After discussing the state of the art in section 2, we present the concept of our SAML architecture extensions in section 3. We focus on implementational aspects and the introduction of our prototype in section 4 and give an outlook to our further research in section 5.

2 Towards Federated Identity Management

Because more and more services and applications supported the LDAP protocol for both authentication and storage of configuration data, LDAP-based enterprise directories have been widely adopted as basis for intra-organizational I&AM solutions, which focus on the integration and centralized management of an organization's employees, customers and users and their access rights to the local services. Unfortunately, as in many other management areas, no single

data schema standard exists, and thus default vendor configurations, such as those found in Microsoft Active Directory or Novell eDirectory, compete with non-proprietary LDAP schema definitions such as inetOrgPerson [7]. In practice, many organizations even create their own LDAP schema to cover their individual needs.

To facilitate cross-organizational identity data exchange, early attempts to grant other organizations access to own enterprise directories quickly turned out to be tedious and suffer from bad scalability. Having to set up accounts for users from other organizations and getting applications to work with different schemas leads to massive administrative overhead and is impractical when more than a handful of organizations is involved.

Thus, dedicated management standards were created, out of which SAML [1] has found wide-spread adopters and is supported by the recent versions of identity management solutions by most big vendors, including HP, IBM, Novell, and Sun. SAML establishes a web services based back channel between the service provider (SP) and the user's home organization, which is called Identity Provider (IDP). Over this back channel, the SP can request information about the user, as shown in figure 1:

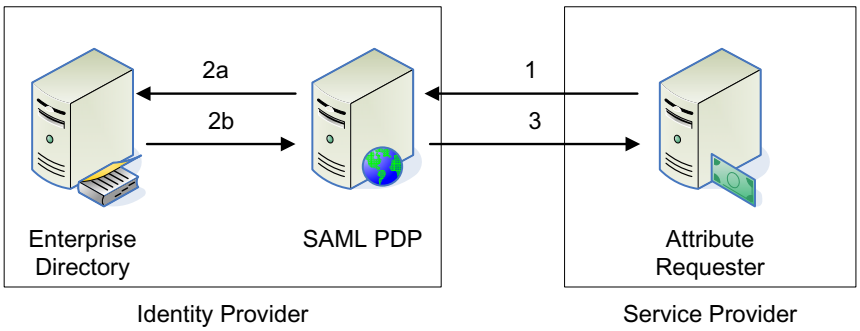

Fig. 1. Identity data exchange through a standard SAML back channel

1. The SP sends a SAML request to the IDP's SAML Policy Decision Point (**PDP**), e.g. it queries the user's billing address. The user is identified by a handle known to both providers, and the billing address consists of *attributes* such as `name`, `street`, `postal code` and `city`, which must have been defined in a federation-wide data schema a priori.
2. Those attributes are looked up in the IDP's local identity repository, which is typically the enterprise directory also used by the local I&AM solution. The result is returned to the IDP's SAML component.
3. The data is wrapped into a SAML attribute assertion and sent back to the SP.

Having to use a federation-wide data schema in a SAML architecture raises two problems: First, finding a common data schema for all involved parties is a non-trivial task due to different technical demands, e.g. different syntactical requirements of applications, and each involved organization's political goals. Second, as an IDP's SAML component must be able to look up the attributes in this schema, the organization either has to use this schema internally as well or provide an extra repository which is synchronized with the local I&AM solution regularly; either way, this causes costs for the extra hardware, synchronization software and operation. Two solution attempts are presently available:

1. The Liberty Alliance provides two standardized schemas, called employee and personal profile ([8], [9]). However, these schemas provide only the greatest common divisor of potentially required identity information, and thus are per se insufficient and have to be extended by other required attributes, similar to their LDAP counterparts.
2. Most vendors support a technique called *attribute mapping*. For example, if an SP requests the `dateOfBirth` attribute, it could be mapped to the `DOB` attribute in the local repository. However, more complex transformations than just renaming an attribute, such as changing the date format from `YYYY-MM-DD` to `DD.MM.YY`, or composing the result from three separate attributes `day`, `month` and `year`, are not possible.

As can also be seen from figure 1, there is no filtering mechanism in place that restricts which attributes are allowed to be sent to the SP. Yet, it is crucial to protect the users' privacy and empower each user to control and restrict which SP has access to which attributes. This issue is also dealt with insufficiently:

- Of the three FIM standards, only Liberty Alliance introduces the idea of *Attribute Release Policies* (**ARP**s). However, it does neither specify the content of such ARPs nor how they should be implemented.
- Only Shibboleth [6], a SAML-based open source FIM software, which is the de-facto standard among higher education institutions, supports ARPs, but in a proprietary format and with rather limited functionality, i.e. the release of each attribute to each SP can be restricted only based on this attribute's current value.

More complex conditions, such as granting access to one's credit card data only if one is actually buying something from a shop and not just browsing for information, cannot be modelled with current ARP concepts and implementations. Also, no obligations can be specified, such as informing a user whenever an SP accesses certain attributes, e.g. by means of an e-mail or a log file.

As both an integration of FIM into existing local business processes and an enhanced privacy protection are urgently required to achieve a smooth setup of identity federations and earning of user acceptance, we have extended the SAML architecture by schema conversion and privacy management components, which are described in the next sections.

3 An Extended SAML Architecture with Schema Conversion and Privacy Management Support

Figure 2 shows our extended SAML architecture; as only the IDP-internal workflow has been modified and the SAML PDP is still the only point of contact to the outside world, we preserve full SAML compliance. We now describe the overall workflow and then go into conceptual details of the attribute conversion component in section 3.1 and specify the privacy management mechanism in section 3.2:

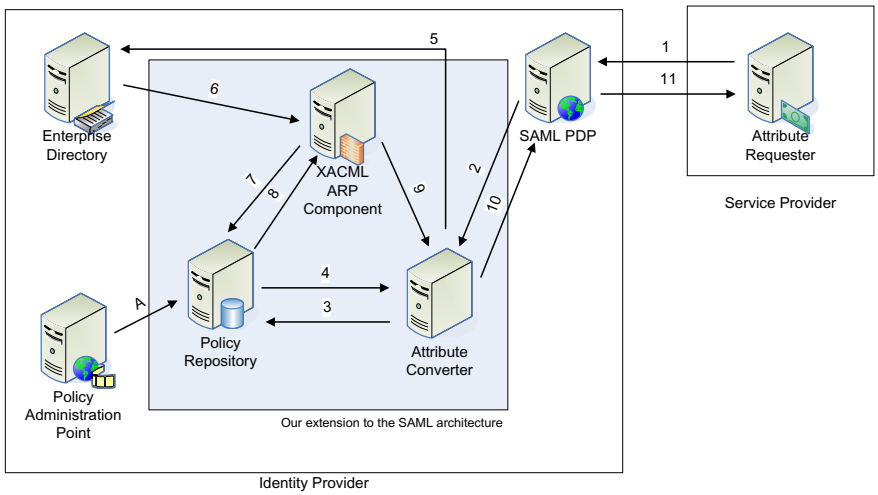

Fig. 2. Extended SAML architecture and workflow

1. The SAML attribute request is sent by the service provider (SP) to the IDP's SAML PDP as before. The data schema used in the request is the federation-wide.
2. The IDP's SAML PDP extracts the list of the wanted attributes from the SAML request; instead of looking them up in the enterprise directory directly, it forwards this list to our attribute converter, which is described in more detail in section 3.1. It also passes information about the requesting SP and the affected user, which are required later to pick the appropriate policies.
3. The attribute converter contacts the IDP's policy repository. It stores pairs of conversion rules which are used to first convert incoming requests from the federation-wide into the locally used data schema, and then convert the results from the locally used to the federation-wide data schema. The rules can be administrated via the policy administration point (see arrow A).
4. The rules required to convert the actually requested attributes to and from the locally used data schema are returned to the attribute converter.

5. The attribute converter translates the list of originally requested attributes into the list of attributes which need to be looked up in the local enterprise directory. By doing so, the names of the attributes can be changed, but attributes may also be added to or deleted from the list, depending on which attributes are required in the local schema to provide the content for the requested attributes in the federation-wide schema. This list of attributes is then looked up in the enterprise directory.
6. Before those attributes can be returned to the SP, Attribute Release Policies (ARPs) are used to protect the user's privacy. We are using the eXtensible Access Control Markup Language (XACML, [5]) to model and enforce ARPs as described in section 3.2. The attributes and their values, which have been retrieved from the enterprise directory, are forwarded to our XACML component.
7. The XACML component looks up the applicable ARPs in the IDP's policy repository. The selection of ARPs depends on various factors, such as the requesting SP, the affected user and the attributes which have been looked up.
8. The relevant ARPs, which typically include IDP-wide ARPs defined by an administrator and user-specific ARPs, are returned to the XACML component. Details are provided in section 4.
9. The XACML ARP component filters the list based on the rules specified by XACML policies as described below and returns only those attributes whose release is allowed back to the attribute converter.
10. This time, the attribute converter has to convert the attributes from the locally used schema back to the federation-wide schema. The necessary rules have already been fetched in step 3. The final result is returned to the IDP's SAML PDP.
11. The IDP's SAML PDP wraps the result in a SAML assertion, which is finally sent to the SP as result of the original attribute request.

The following sections describe the internals of the attribute converter and the XACML ARP component in detail.

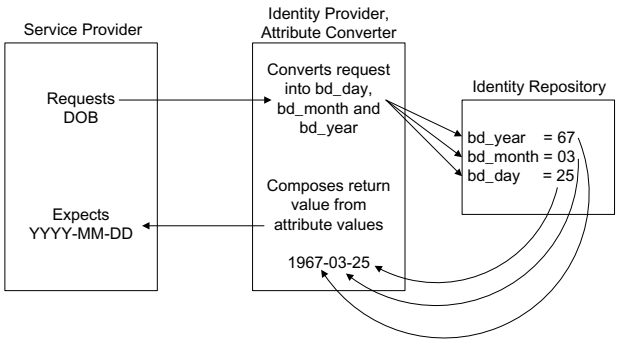

Fig. 3. Solution workflow for a simple schema mismatch example

3.1 Cross-Organizational Identity Schema Conversion

The attribute converter's purpose is to enhance the interoperability of FIM solutions with existing I&AM systems by letting all IDP components work with the IDP's locally established data schema and nevertheless communicate with federation partners transparently.

An example of its use is shown in figure 3. It assumes that a federation has defined a `DOB` attribute which holds the user's date of birth in the format `YYYY-MM-DD`; however, the IDP, which receives the request, stores the user's date of birth in three separate attributes for day, month and year of birth. Even such seemingly simple problems cannot be solved with the existing FIM standards.

The figure also shows the attribute conversion relevant steps of the workflow described above. First, the incoming request for one attribute is modified, so the three locally required attributes are looked up. Then, before returning the result to the SP, the return value for the originally requested attribute is composed from the three separate attributes.

Three kinds of conversions can be made:

1. The names of the requested attributes can be changed, e.g. from `DOB` to `dateOfBirth`. This is equivalent to the attribute mapping approach described in section 2.
2. The attribute's value can be text-processed, in order to fulfill syntactical requirements of the target data schema, or to compose or split up attributes.
3. The attribute's value can be modified to adapt different semantics; for example, the value of a user's `nationality` attribute may have to be `German` in the local schema but `DE` in the federation-wide. Such semantical conversions are eased in practice because many attributes can only have discrete values.

The actual conversion rules are specified as XSLT [10] stylesheets, i.e. XML transformations are performed; an example is given in section 4. XSLT is an obvious choice, as all FIM standards are XML-based and most of their implementations use XML internally as well. Furthermore, also well-established products for local I&AM, such as Novell's Nsure Identity Manager 2, are using XSL transformations for intra-organizational data conversions, so existing programming experience and code can be reused efficiently in the federated case.

3.2 Inter-domain Policy-Based Privacy Management

Privacy management is a well-studied field; standards such as P3P [11] and EPAL [12] have been widely adopted. However, they are intended to specify, publish and enforce privacy policies on the service provider side; they do not specify how the user's preferences shall be stored on the client or identity provider side. Existing implementations, such as Shibboleth [6], use proprietary privacy policy formats on the IDP side; thus, users cannot reuse their policies at other IDPs if they use a different implementation.

We have chosen the eXtensible Access Control Markup Language (XACML, [5]) as basis for the implementation of Attribute Release Policies (ARPs) for the following reasons:

- XACML and SAML have been paired before to achieve fine-grained inter-domain access control, e.g. in well-known systems such as PERMIS [13] and Cardea [14]; a detailed overview can be found in [15].
- XACML is a OASIS standard with a reference implementation available as open source [16]. As XACML is a generic access control language, the policy format can be tailored to individual needs and still be evaluated by any standard compliant XACML PDP; this ensures interoperability and eliminates the need to implement a dedicated PDP. XACML's relationship to P3P, which has been outlined in [17], allows us to leverage a proven privacy standard to FIM applications.
- XACML already provides functionality which is required for advanced ARPs but not available in current proprietary implementations, e.g.
  - Support for multiple roles of a user, e.g. one used at work and one used in spare time.
  - Grouping of attributes, i.e. release rules do not have to be specified for each attribute separately, e.g. as in Shibboleth.
  - Arbitrarily decentralized management, i.e. multiple XACML policies can be combined to form the effective policy. Typically, an IDP administrator will specify default policies which each user can override individually on demand.
  - Very flexible condition formulation; for example, certain attributes may only be released for a certain purpose which the SP has to disclose. Conditions may also contain environmental data such as the current date and time.
  - Formulation of obligations. Logging an SP's access to selected attributes and informing the user by e-mail are two popular obligations which are already part of the XACML standard; arbitrary other obligations can be implemented through XACML's extension mechanisms.
  - Policy protection, i.e. an existing public key infrastructure (PKI) can be used to sign and encrypt the ARPs to prevent unauthorized modification and disclosure.

The XACML PDP component shown in figure 2 consists of an XACML policy enforcement point (PEP) and a standard XACML PDP. The PEP creates XACML requests based on the attributes which are passed in from the enterprise directory (see step 6 of the workflow on page 53). It then fulfills any obligations returned by the XACML PDP and returns the attribute values to the converter component if their release was allowed. An example can be found in the next section.

4 Implementation Details

We will now describe our implementation of the SAML architecture extension and the integration of its components into Shibboleth.

As described in section 3.1, the attribute converter uses XSLT stylesheets to transform incoming requests and outgoing responses. We have implemented this

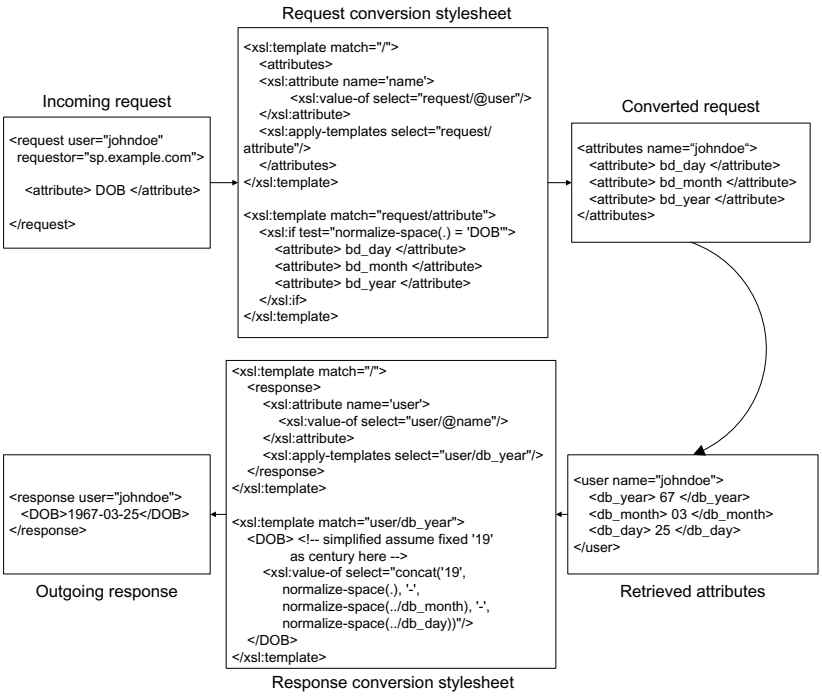

Fig. 4. An example for request and attribute conversions using XSLT

functionality using Xalan [18] as XSLT processor through the standard JAXP Java API. Our prototype uses the local file system as XSLT stylesheet repository; additionally, a web server can be used as policy administration point (PAP) to upload new stylesheets and edit them online (see arrow A in figure 2). Example stylesheets which solve the date of birth schema problem discussed in section 3.1 are shown in figure 4.

Shibboleth provides data connectors for relational databases, LDAP servers and flat text files; it also has an extension mechanism which can be used to hook in custom connectors. By implementing such a custom connector, attribute lookups can be redirected to the attribute converter, which in turn uses the Java JNDI API to retrieve the attributes from the enterprise directory in the local data schema. It passes their values and the meta-data about the service provider, which is delivered by Shibboleth, on to the XACML ARP component.

The XACML ARP component is also implemented in Java, facilitating Sun's XACML PDP implementation [16]. The attribute values and meta-data are received by a custom XACML PEP, which assembles an appropriate XACML request; this request is then evaluated by the PDP, which returns the decision whether the attribute may be released to the given service provider under the given conditions, along with optional obligations. The following example shows an XACML policy which grants access to the user's credit card number to an

online shop only if an actual order is placed; an obligation specifies that each allowed release must be logged:

```
1  <Policy id="xacmlARP1" RuleCombiningAlg="first-applicable">
2    <CombinerParameters>
3      <CombinerParameter ParameterName="ARPpriority">
4        100
5      </CombinerParameter>
6    </CombinerParameters>
7    <Description> ARP by user John Doe </Description>
8    <Rule id="CreditCardToBookShop" effect="permit">
9      <Description> Release credit card number to bookshop </Description>
10     <Target>
11       <Resources>
12         <Resource>
13           <ResourceMatch MatchId="string-equal">
14             <AttributeValue>
15               idp.example.com/johndoe/defaultrole/creditCardNumber
16             </AttributeValue>
17             <ResourceAttributeDesignator AttributeId="resource-id" />
18           </ResourceMatch>
19         </Resource>
20       </Resources>
21       <Subjects>
22         <Subject>
23           <SubjectMatch MatchId="string-equal" AttributeValue="shop.example.com">
24             <SubjectAttributeDesignator AttributeId="service_provider" />
25           </SubjectMatch>
26           <SubjectMatch MatchId="string-equal" AttributeValue="bookshop">
27             <SubjectAttributeDesignator AttributeId="service" />
28           </SubjectMatch>
29           <SubjectMatch MatchId="string-equal" AttributeValue="purchase">
30             <SubjectAttributeDesignator AttributeId="purpose" />
31           </SubjectMatch>
32         </Subject>
33       </Subjects>
34       <Actions>
35         <Action>
36           <ActionMatch MatchId="string-equal" AttributeValue="read">
37             <ActionAttributeDesignator AttributeId="action-id" />
38           </ActionMatch>
39         </Action>
40       </Actions>
41     </Target>
42     <Obligations>
43       <Obligation Id="Log" FulfillOn="Permit">
44         <AttributeAssignment Id="text">
45           Your credit card number has been released to:
46           <SubjectAttributeDesignator AttributeId="service_provider" />
47         </AttributeAssignment>
48       </Obligation>
49     </Obligations>
50   </Rule>
51   <Rule id="DoNotReleaseAnythingElse" effect="deny"/>
52 </Policy>
```

As can be seen from lines 2–6 of the example, we are using a priority based policy combining algorithm which composes the effective policy out of an arbitrary number of optionally distributed ARPs. In practice, ARPs created through the user's PAP typically have a higher priority than the administrator-specified ARPs, so users can override the IDP's defaults. The attributes to which access is controlled are specified as XACML resources, as shown in lines 11–20 of the example; each attribute is identified globally by its name, which is a URN composed of the IDP identifier, the person, its role and the attribute name as specified in the federation-wide data schema. Three consecutive XACML subject matches control which service provider is actually requesting the attributes for the provisioning of which service and which purpose (see lines 21–33).

An integration into Shibboleth's IDP component can be achieved by first adapting the `listPossibleReleaseAttributes()` method, which must return the names of the user attributes which should be retrieved; second, `filterAttributes()` has to remove all attributes whose release is not permitted by the ARPs. The user's and service provider's ids are passed to both methods,

which provides sufficient information for the XACML PEP to identify, combine and let the PDP evaluate the relevant XACML-based ARPs.

Shibboleth's proprietary ARPs can be lossless converted to XACML ARPs. Basically, Shibboleth ARP `targets` become XACML `subjects` and Shibboleth ARP `attribute` elements are converted to XACML `resources`. As release decisions are made on `attribute` and not on `rule` level in Shibboleth ARPs, each Shibboleth `attribute` has to be converted into a dedicated XACML `rule`. We have automated this transformation by also using an XSLT stylesheet.

5 Summary and Outlook

In this paper, we presented a SAML-based architecture for privacy-aware distributed service provisioning, which allows a tight integration of inter-domain provisioning workflows into the individual local identity management business processes. Two urgent problems of current standards have been addressed while still maintaining full compliance. First, we added an attribute converter to the standard SAML architecture; it utilizes XSLT stylesheets to convert incoming SAML attribute requests and outgoing responses from the federation-wide data schema to the locally used one and vice versa, so SAML can be integrated into the local I&AM infrastructure transparently and at minimum cost. Second, to protect each user's privacy across administrative domains, we specified Attribute Release Policies based on XACML, a generic access control language, which has been successfully paired up with SAML for various other purposes before. We demonstrated our implementation and its use on simple real-world problems.

Our further research will focus on improving other weak spots of current FIM standards; in particular, we will study the use of data pushing mechanisms to complement the current pull-only SAML protocol bindings.

Acknowledgment

The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful discussions and valuable comments on previous versions of the paper. The MNM Team directed by Prof. Dr. Heinz-Gerd Hegering is a group of researchers of the University of Munich, the Munich University of Technology, and the Leibniz Supercomputing Center of the Bavarian Academy of Sciences. The web server of the MNM Team is located at `http://www.mnm-team.org/`.

References

1. Cantor, S., Kemp, J., Philpott, R., Maler, E., (Eds.): Security Assertion Markup Language v2.0. OASIS Security Services Technical Committee Standard (2005)
2. Wason, T., Cantor, S., Hodges, J., Kemp, J., Thompson, P., (Eds.): Liberty Alliance ID-FF Architecture Overview. `http://www.projectliberty.org/resources/specifications.php` (2004)

3. Kaler, C., Nadalin, A., (Eds.): Web Services Federation Language (WS-Federation). http://www-106.ibm.com/developerworks/webservices/library/ws-fed/ (2003)
4. Hommel, W., Reiser, H.: Federated Identity Management: Shortcomings of existing standards. In: Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Management (IM 2005), Nice, France (2005)
5. Moses, T.: OASIS eXtensible Access Control Markup Language 2.0, core specification. OASIS XACML Technical Committee Standard (2005)
6. Cantor, S., Carmody, S., Erdos, M., Hazelton, K., Hoehn, W., Morgan, B.: Shibboleth Architecture, working draft 09. http://shibboleth.internet2.edu/docs/ (2005)
7. Smith, M.: Definition of the inetOrgPerson LDAP Object Class. IETF Proposed Standard, RFC 2798 (2000)
8. Kellomki, S.: Liberty ID-SIS Employee Profile Service Specification. http://project-liberty.org/specs/liberty-idsis-ep-v1.0.pdf (2003)
9. Kellomki, S.: Liberty ID-SIS Personal Profile Service Specification. http://project-liberty.org/specs/liberty-idsis-pp-v1.0.pdf (2003)
10. Clark, J.: XSL Transformations (XSLT), Version 1.0. W3C Recommendation, http://www.w3.org/TR/xslt/ (1999)
11. Reagle, J., Cranor, L.F.: The Platform for Privacy Preferences. In: Communications of the ACM. Volume 42., ACM Press (1999) 48–55
12. Powers, C., Schunter, M.: Enterprise Privacy Authorization Language, W3C member submission. http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/ (2003)
13. Chadwick, D., Otenko, A.: The PERMIS X.509 Role Based Privilege Management Infrastructure. In: Proceedings of the 7th ACM Symposium on Access Control Models and Technologies. SACMAT, ACM Press (2002) 135–140
14. Lepro, R.: Cardea: Dynamic Access Control in Distributed Systems. Technical Report TR NAS–03–020, NASA Advanced Supercomputing Division, Ames (2003)
15. Lorch, M., Proctor, S., Lepro, R., Kafura, D., Shah, S.: First Experiences Using XACML for Access Control in Distributed Systems. In: Proceedings of the ACM Workshop on XML Security, ACM Press (2003)
16. Proctor, S.: Sun's XACML implementation. http://sunxacml.sf.net/ (2004)
17. Anderson, A.H.: The Relationship Between XACML and P3P Privacy Policies. http://research.sun.com/projects/xacml/ (2004)
18. Apache Software Foundation: Xalan XSLT Processor. http://xml.apache.org/xalan-j/ (2005)