

Policy-Based Integration of User and Provider-Sided Identity Management

Wolfgang Hommel

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

Abstract. Depending on whether the users or the providers are performing it, Identity Management (IM) traditionally has different meanings. For users, IM means to choose between one's own identities and roles, in order to make selected personal information available to providers under privacy aspects. For providers, IM typically consists of centralized identity data repositories and their use by the offered services. Methods and tools for both aspects of IM have developed almost orthogonally, failing to consider their interoperability and complementary purposes. We analyze the similarities between both IM aspects and demonstrate how both sides can benefit from the use of a common policy language for personal information release and service provisioning. We derive criteria for this common policy language, demonstrate XACML's suitability and discuss our prototype for the Shibboleth IM system.

1 Introduction and Problem Statement

Identity management (IM) has turned into a double-edged term over the past few years. From the user's point of view, managing her identities is comprised of registering different subsets of her personal information at service providers and subsequently selecting and using these different profiles appropriately [1, 2]. Also, many research projects focus on the privacy issues of this aspect of IM, often closely related to anonymous service usage [3, 4]. On the other hand, major software vendors sell *identity & access management* software to enterprises; its purpose is to centrally acquire the relevant identity information and make it available to the local systems and services, which is referred to as *provisioning*. Concerning privacy, typically the conformity with laws, user acceptance, long-term data retention and other obligations are the primary issues [5, 6, 7].

Methods and tools for these related, yet different aspects of IM have been developed independent of each other; thus, many of them are suffering from interoperability deficiencies and the lack of fulfilling some of other side's elementary requirements. For some branches, approaches to privacy-aware IM have been mutually agreed on by both sides, e.g. regarding e-commerce [8, 9, 10]. However, standards such as P3P are limited to their respective domain and cannot be used in other types of identity federations in general. For application areas such as Grid computing [11], e-learning [12] and business-to-business (B2B) outsourcing [13], dedicated solutions are required and actually being developed.

In this paper, we demonstrate the role of user-sided Attribute Release Policies and provider-sided Attribute Acceptance Policies for privacy-aware IM and present an integrated approach, which applies the policy language XACML successfully for both purposes. In section 2, we analyze the requirements on both ends and give an overview of the currently existing solutions. We discuss in section 3 how ARPs and AAPs fit together and why a common policy language should be used. The use of XACML for this approach is specified in section 4, which focusses on the service provider side, complementing our earlier work regarding the user side. In section 5, we share our preliminary experiences with this approach, and also discuss its present limitations. An outlook to our next research activities concludes this paper.

Throughout this paper, we use the following terms and acronyms:

Service Provider (SP). A real or virtual organization which offers one or more services to users based on arbitrary conditions; the primary condition we consider in this paper is that the SP acquires sufficient information about a potential user from a *trusted* data source.

Identity Provider (IDP). For our discussion, we define an IDP as entity which stores personal information about one or more users. Atomic elements of this information are called **attributes**. The two most common IDP types are a) a software or hardware device which the user runs locally, which we call a **local wallet**, and b) an organization which offers this functionality as a service. Typically, in the latter case, the organization will verify the data before storing it, and vouch for its correctness afterwards.

Identity Federation (IF). A set of SPs and IDPs with established trust relationships; thus, SPs can rely on the correctness of the data retrieved from IDPs in the same IF. Examples for IFs are the *Circles of Trust* as suggested by the Liberty Alliance [14]. The term **Federated Identity Management (FIM)** refers to technical and organizational measures as well as legal aspects within IFs; its most widely adopted industrial and academic approaches are SAML, Liberty Alliance, WS-Federation and Shibboleth.

Attribute Release Policies (ARPs). As detailed in section 2, releasing personal information to SPs is based on rules and criteria specified by each user, which we model and enforce as policies; the term ARP for these policies has been coined by Shibboleth [15], which is based on SAML.

Attribute Acceptance Policies (AAPs). SPs will accept users for each service only if sufficient information about them is available under acceptable conditions, e.g. regarding data handling and retention limitations, and the values of the retrieved attributes fulfill arbitrary criteria, e.g. for access control. These criteria are formulated as AAPs.

2 Requirements Analysis and State of the Art

Obviously, in order to successfully build an integrated, common solution for both sides, the requirements and goals of the users as well as the SPs must be considered. We analyze these requirements and refer to previous approaches on

the user side in section 2.1 and on the provider side in section 2.2; we then discuss their similarities and derive consequences for an integrated approach in section 2.3.

2.1 User-Sided Requirements and Solutions

From the user perspective, we have to distinguish four aspects of identity management, which are also shown in figure 1:

1. Which identity data does the SP need? Three main types of services are presently being offered:
 - Services that can be used anonymously, i.e. do not require any information about the user at all, or only collect data, which does not make the user distinguishable from a sufficiently large set of other users (called *anonymity set* [1]).
 - Services that can be used pseudonymously, i.e. they utilize profiles to identify their users individually, without necessarily acquiring any personally identifying information (PII), e.g. internet discussion forums.
 - Services that users must be personally known to, e.g. online banking.

These categories are also known as “nymity levels” [16]: Services may allow users to select their preferred level, depending on how much information they want to reveal, e.g. in order to benefit from website personalization [17].
2. Which identity and role is the user acting in? Users may want to use different identities, e.g. for professional and private activities, which in turn may be divided into several roles, e.g. being a lecturer, researcher and dean at a university. For example, using different email addresses for each of these roles will result in different attribute values stored in each profile.
3. Where is the identity data stored? As a general principle of identity federations and single sign-on systems, manually registering the data at each SP is not desired, which basically leaves two options [18]:
 - The data is stored locally on the user side, i.e. in a software or hardware device, which, for example, automatically fills out the SP’s HTML formulars and manages the user’s login names and passwords. While such a *local wallet* approach gives the user full control over her data [19], additional measures are necessary in order to enable the SP to verify the authenticity of this data. Attribute certificates, which are signed by a trusted authority, could be used for this purpose, but turned out to be hard to handle on the user side in practice [20].
 - The data is stored by an organization, which is trusted by both the user and the SPs which request the data from it. As centralized approaches, such as Microsoft Passport, have failed, modern FIM architectures are decentralized and allow an arbitrary number of IDPs within IFs. However, this complicates the trust relationships between the providers, which today are mostly based on contracts that have been made out-of-band a priori.

4. How is the identity data being used? For example, is it only being used locally by the SP for service provisioning, or will the data be released to third parties, and how long will it be retained?

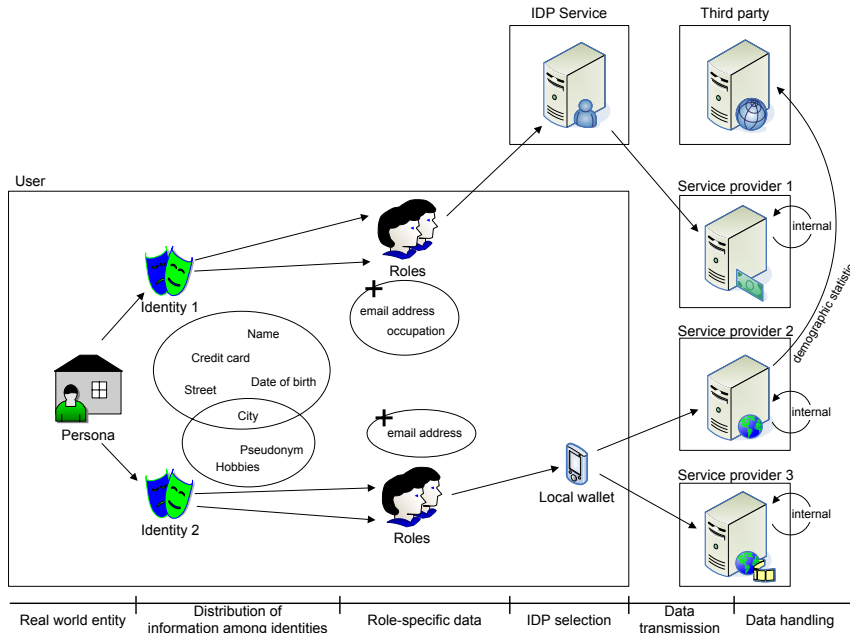


Fig. 1. User-sided aspects of Identity Management

Traditionally, users had to study the services’ terms and conditions and privacy statements in detail before deciding which information they sign up with. Thus, any automation of this process must be flexible enough to capture and express the rationale behind these decisions.

For mere web sites, various *local wallet* implementations exist, with P3P [8] being a profound standard with the highest adoption rate: P3P-enabled browsers fetch the web sites’ P3P privacy statements, parse and compare them to the user’s preferences and can fill out HTML forms automatically or warn the user about conflicts.

However, P3P and the other implementations are limited to web sites, a pre-defined user data schema, and are not interoperable with the modern FIM protocols, as we have discussed in previous work [21].

Consequently, users need a more generic way to specify

- which identity and role to use for
- which service provider and service,
- which information (attributes) to release
- under which conditions (e.g., release the credit card number only if actually placing an order, not when just browsing for information), and
- under which obligations (e.g., delete the data after 90 days of inactivity).

Additionally, in many situations, such as B2B cooperations, users cannot freely choose which information may be released about them; thus, there must be a way for administrators to specify defaults, which may either be refined by the users or override the users' settings [13].

In the terms of Shibboleth, such specifications are called Attribute Release Policies (ARPs). While there is no policy language standard for ARPs yet, various prototypes for the diverse FIM approaches exist [22, 23, 24, 15, 25, 26].

2.2 Provider-Sided Requirements and Solutions

Obviously, IM requirements increase with the types and number of services offered by an SP. In our analysis we assume that the SP is an enterprise offering multiple services, which require different subsets of the user's identity data over a longer period of time, as this imposes the hardest requirements on the implemented IM system.

Historically, the services or their administrators had to acquire and maintain the identity data themselves, which has led to redundant work, data inconsistencies within the enterprise, and thus a higher total cost of ownership.

Identity & Access Management (I&AM) systems, as presently available from several major software vendors, recentralize an enterprise's identity data management processes, typically by storing all user information in a central relational database management system or an LDAP-based directory service. The services then either directly access this central identity repository or are being fed by it.

Dedicated management frontends for administrators and self service web interfaces for users exist to maintain the data. However, the current FIM protocols, such as SAML, are not sufficiently integrated with I&AM software yet; instead, they are intended to directly feed the services with identity data, effectively bypassing the I&AM software and its central identity repository. While this may be sufficient for SPs which only offer a single service, e.g. an e-commerce website, it is counterproductive for larger enterprises which already have an I&AM infrastructure deployed.

Concerning the information, which is requested about users, SPs distinguish between

- identity data that must be known for service provisioning, e.g. for accounting and billing purposes. Unless this data is available and valid, the service cannot be used.
- additional data that can improve the user experience, e.g. through personalization.
- additional data that can be used for other purposes, e.g. data mining for long-term service improvement.

Additionally, parts of this data must be or could be given to third parties; for example, an e-commerce web site might use an external service for credit card billing and has to give the user's shipping address to a courier or postal service.

Regarding privacy, enterprises are bound to laws as well as to user acceptance issues; we distinguish between

- the acquisition or collection of data, which can be based on
 - information provided by the user or IDP, e.g. during signing up for the service, and
 - information collected during service usage, e.g. analyzing the user’s click-stream on web sites [24],
- data handling policies, e.g. making data available to third parties [7].
- data protection measures, e.g. encryption or auditing to prevent internal abuse, as well as the enforcement of the stated privacy practices [5].
- data retention policies, e.g. logfiles are deleted after a fixed period of time [27].

In this paper, we focus on the acquisition and maintenance of information provided by the user’s IDP. To this extent, SPs have to specify which data they acquire for which purposes, e.g. in their privacy statements; while P3P can be used for single web sites, no such standards exist yet for IFs and generic FIM.

Some implementations, such as Shibboleth [28], support the specification of AAPs based on simple proprietary policy languages. They can be used by the service administrators to specify which user attributes shall be requested from the IDP, and to formulate simple conditions on their values. In practice, only simple access control mechanisms are implemented this way, which have to be maintained separately and are not integrated into the SP’s I&AM system.

The use of AAPs in IFs and their integration into I&AM systems has not been researched in detail yet; in section 4 we demonstrate the suitability of the policy language XACML for AAPs.

2.3 Similarities and Synergies

ARPs and AAPs have been treated isolated of each other in most previous research and implementations. In the above sections, we summarized the necessity of both types of policies, making obvious how they complement each other. Consequently, we believe that ARPs and AAPs must be discussed in unity to achieve interoperability in FIM and a seamless integration into the SP’s I&AM system.

Policy-based management provides useful formal methods and tools to model and enforce these information release and acceptance decisions, and the similar structure and content of ARPs and AAPs, which we detail in the next section, can be exploited by the use of a common policy language for both types.

3 Policies for Privacy Management and Service Provisioning

We subsequently demonstrate that a policy based management approach fulfills the requirements discussed above. Again, we first discuss the user and the SP sides separately and then point out their commonalities. This section is explicitly independent of any concrete policy language; whereas several existing

policy languages could be used for the purposes described below, or a new dedicated language could be created, our prototype is based on XACML as described in section 4.

3.1 User or IDP-Sided Attribute Release Policies

In order to use policies as a formal base of automated attribute release decisions, we define the content of each policy and discuss how multiple policies can be combined and work together.

To meet the criteria laid out in section 2.1, an ARP must provide syntactical elements for the following tasks:

- Naming the user attributes, e.g. the user's given name, which are stored in an IDP database. A data schema, such as P3P's, defines which attributes a user object may have; in general, not all SPs and IDPs will use the same data schema, but on-the-fly conversion mechanisms that work with FIM protocols exist [13].

As the values of attributes may vary with the selected identity and role of the user (which are known to the IDP, and optionally to the SP), we use URIs in the format `https://address.of.idp/identity/role/attributename` to name attributes. Furthermore, attributes may be grouped for user convenience, so for example the shipping address of a user, consisting of full name, street, ZIP code and city name, can be referred to as a single attribute when specifying release policies.

- Naming SPs, services and request purposes. This distinction is necessarily hierarchical, as one SP may offer multiple services and each service may request information for various purposes (e.g., personalized browsing vs. handling an actual order). Again, URIs in the format `https://address.of.sp/service` can be used as identifiers for tuples of SPs and services, and a common terminology for purpose specifications must be established in the IF (see also [22]).
- Specifying the type of access allowed on the data. Presently, we distinguish between a one-time read access to the data and subscribing to the attribute; in the latter case, changes in the attribute may be made available to the SP after service usage, in order to avoid data inconsistencies between IDP and SP. In the future, also write access may be grantable to SPs; however, current FIM protocols do not support this operation yet. Each request must provide the tuple (*access type, purpose*).
- Formulation of conditions. For example, the release of an attribute may depend on its current value, the release of other attributes, or the presence of certain criteria in the SP's privacy statement.
- Specification of obligations on both the IDP and the SP side. As a typical obligation that has to be fulfilled by the IDP, users want to be notified when an SP attempts to access certain sensitive attributes. SP-sided obligations may include restrictions on the maximum data retention time or the opt-out of giving the data to third parties.

For each user, an arbitrary number ARPs can be specified; typically, there will be:

- One default ARP, which is used if no other ARP matches a request. Usually, this default ARP will deny the release of any attribute.
- Any number of SP-specific ARPs set by the user according to her preferences, either a priori, or at run-time if the used FIM protocol allows suitable interaction mechanisms (such as BBAE [29] or Liberty [30]).
- Any number of SP-specific ARPs set by the IDP administrators, e.g. according to B2B contracts.

Policies, which refer to the same SP, service, and purpose, may cause policy evaluation conflicts, e.g. if one policy allows the release of an attribute while the other policy denies it. While a lot of research on policy based management suggests various sophisticated conflict avoidance, detection and resolving techniques, we presently make use of a simple priority-based policy combination workflow: Depending on which priorities users are allowed to assign to their policies, user settings may override administrator settings or vice versa.

3.2 Attribute Acceptance Policies for Service Providers

In analogy to ARPs on the user side, SPs specify in an AAP:

- Which services the AAP applies to, also identified by URIs as in the ARPs described above. This allows to manage multiple services through a single AAP, in contrast to existing implementations, in which each service requires a dedicated AAP, whose content then is possibly redundant.
- Which IDPs and requested user attributes this policy applies to (policy target specification). The list of trusted IDPs is typically derived from federation meta-data; the data schema and conversion issues discussed in the previous section also apply here. The currently selected identity and role of the user do not have to be known.
- Which purpose the data is being requested for. This field is intended to hold the identifier of a machine or human readable section of a privacy statement in which the data handling, protection and retention practices are stated; it thus complements, and does not replace, existing mechanisms like P3P.
- Conditions on the availability and values of attributes. For example, a credit card number might not be accepted unless also the expiry date is provided, or the number of an already expired credit card may be rejected.
- Obligations for SP-sided policy evaluation, such as keeping logfiles of denied transactions. Other obligations, e.g. for data handling, protection, and retention, could be specified here as well; however, in this paper we focus on the data acquisition only and will investigate the integration of complementary methods, such as EPAL [5], in our further research.

An arbitrary number of AAPs can be combined with each other, e.g. in order to specify enterprise-wide defaults and service-specific refinements thereof. In

analogy to ARPs, using a priority-based policy combination algorithm to resolve potential policy conflicts allows a fine-grained identity data provisioning configuration for each service and enables decentralized service-specific administration.

3.3 Using a Common Policy Language

Unlike existing approaches to ARPs and AAPs [15, 25, 31, 10], we suggest to use a common language for both types of policies due to their structural similarity and their complementary contents.

Furthermore, using a common language for both ARPs and AAPs brings several practical advantages, such as lower implementation overhead for both IDP and SP software, as well as the possibility to use the same or at least similar tools and graphical user interfaces to maintain the policies.

A policy language must be able to specify the following elements to be suitable for the purposes discussed above, which are also shown in figure 2:

- Subjects, i.e. services which request attributes (in ARPs), and users which want to use services (in AAPs).
- Objects, such as the protected user attributes (in ARPs) and services (in AAPs).
- Actions, such as reading an attribute for a certain purpose (in ARPs) or using a certain functionality of a service, which relates to the purpose of an attribute request (in AAPs).
- Conditions, which decide whether the subjects are allowed to perform the actions on the objects.
- Obligations, which have to be fulfilled or queued for later fulfillment during policy enforcement.

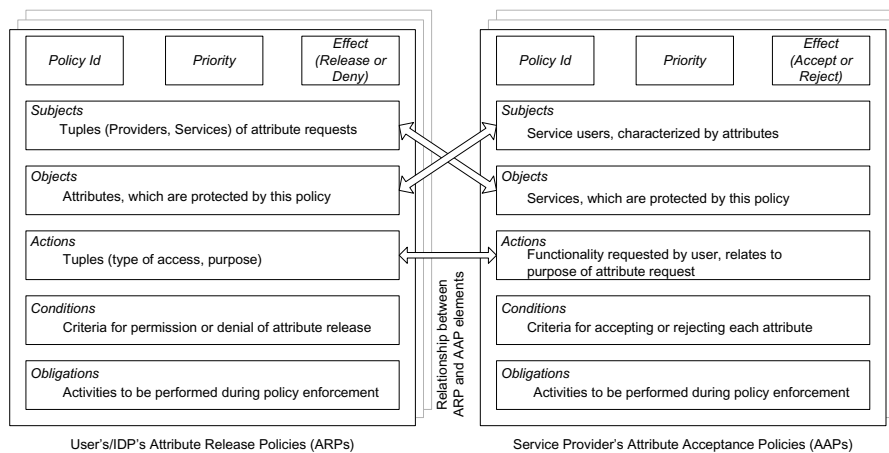


Fig. 2. Common structure of ARPs and AAPs

As further requirements, it must be possible to combine multiple distributed policies, e.g. ARPs specified by the administrator and by the user, and to efficiently decide whether a policy is relevant for an actual request or can be safely ignored. The necessary policy elements lead to access control specific policy languages. We subsequently discuss the policy evaluation workflow and present the use of XACML as policy language for ARPs and AAPs in section 4.

3.4 Policy Evaluation Workflow

A Policy Decision Point (PDP) is used to evaluate the policies, whereas the consequences are taken by the Policy Enforcement Point (PEP), which is

- the FIM component, which replies to attribute requests, in the case of ARPs on the IDP side
- the service, which the user wants to use, or a component delivering the data to it, in the AAP case.

As the PDP may need to evaluate multiple policies in order to decide, and the policies may be arbitrarily distributed, there must be a database or registry service which lists all policies that have been set up for a user (IDP-sided) or service (SP-sided). Out of all policies related to a user or service, the ones, which are relevant for a request, can be selected based on their target specifications:

- For ARPs, all policies that match the $(SP, service)$ and $(access\ type, purpose)$ tuples of the attribute request must be considered.
- For AAPs, those policies must be combined, which match the $(IDP, user\ attributes, purpose)$ triple of an attribute response.

After the relevant policies have been identified this way, they need to be combined into a single so-called policy set, i.e. a sequence of policies, which the PDP can evaluate in one logical step.

As discussed in the previous section, we are presently using a simple priority-based policy combination algorithm, i.e. a different priority is assigned to each policy, and the policy set is ordered by the descending priority of the contained policies. While this is sufficient for the small amount of policies we are currently working with, clearly more sophisticated policy conflict detection and avoidance approaches will have to be used in larger scale environments.

The created policy set is then evaluated by a PDP to decide

- whether a user attribute may be released to an SP (on the IDP side)
- whether the attribute, its origin, value and associated obligations are acceptable for the service (on the SP side).

The PDP's decision is sent to the PEP, along with the list of optional obligations. The PEP has to fulfill these obligations, or trigger their later fulfillment, e.g. by setting a date for data deletion to enforce retention obligations.

4 XACML as ARP and AAP Policy Language

We have analyzed existing FIM-specific approaches and concluded that most of them fail short of fulfilling the core requirements of ARPs in previous work [32], thus they are also unsuitable for the modelling and enforcement of AAPs, especially as they have not been designed for this purpose.

However, various more generic policy languages exist and are suitable for both ARPs and AAPs, e.g. PERMIS' policy language [33], Ponder [34], XrML [35] and XACML [36]. Out of those, XACML excels as most frequently used policy language in the areas of FIM and distributed access control (for an overview, see [37]; details can be found in [38, 39, 40, 41]).

We have detailed how XACML can be used for ARPs in previous work [26]; subsequently, we specify how AAPs can be modeled in XACML, and discuss the evaluation and enforcement workflows.

4.1 Modeling AAPs with XACML

In analogy to the specification in section 3.2, we use XACML policy elements as follows for AAPs:

- The services, which the policy applies to, are the **Resources** in XACML terms. Multiple services can be specified by URIs, optionally using wildcards, regular expressions or the XACML **AnyResource** element.
- The URIs of the requested user attributes and optionally the trusted IDPs are specified as XACML **Subjects**, i.e. consecutive XACML **SubjectMatches**. The restriction to certain IDPs, which could for example be derived from a federation member list, is only necessary if the used FIM software does not communicate with trusted IDPs only anyway, or if different trust levels have to be supported.
- Request purposes are related to the service functionality, which the user has requested, and they point to sections of privacy statements and policies. We store identifiers of the different functionalities as XACML **Actions** and the pointers as XML attributes of **Action** elements in the URI format. Multiple actions can be specified in a single AAP if the same user attributes are requested for different purposes, depending on the used service functionality.
- A XACML **Condition** element can be used for the formulation of acceptance conditions; XACML offers a variety of statements and functions, which can be used to create arbitrary complex conditions, e.g. based on the user attributes' values or environmental data such as the current date and time.
- XACML **Obligation** elements are used to specify short- or long-term obligations. Currently, XACML only supports sending emails and writing to log-files as standardized obligations, but arbitrary obligations can be specified as long as the PEP knows how to interpret them.

The above elements are used within XACML **Rules**; each AAP may contain any number of rules and must specify a rule combination algorithm, such as “first-applicable” or “deny-overrides”, and optionally have a description, which can be

used to document its purpose. Additionally, we assign a priority to each AAP to control the policy set combination.

4.2 Evaluation and Enforcement of XACML AAPs

To create the XACML `PolicySet`, the relevant AAPs are chosen by matching their subjects, actions and objects against the current user attribute response sent by the IDP and the context of the current request. We had to implement the priority-based policy combination algorithm as it is not part of the XACML standard yet, which however allows the definition of custom policy combiners.

The resulting `PolicySet` can be evaluated by any standard-compliant XACML PDP, such as Sun's reference implementation [42]. Presently, we have to trigger the PDP for each user attribute separately and cannot evaluate all user attributes at once, because XACML results can contain only one decision, but we need to be able to accept some user attributes, even if others are not accepted. The performance drawback is discussed in section 5.

The PDP's decision is enforced by the PEP, which is either the service itself or a component which delivers the user attributes to the service; it also has to fulfill the obligations, which are part of the PDP's decision. Guaranteeing the enforcement of obligations, which have been stated in human- or machine-readable privacy policies external to the AAPs, is outside the scope of our work, but our AAPs provide the interface to complementary components such as those described in EPAL and TPM-related work [43].

5 Preliminary Experiences

We have extended our XACML ARP prototype for Shibboleth, which we described in previous work [26], to also support XACML AAPs on the SP side. Although this prototype is not yet made available to our users, implementing and testing it revealed promising preliminary experiences and leads to further tasks:

- Adapting Shibboleth to XACML AAPs by using Sun's reference XACML PDP implementation is straight-forward, as only the AAP fetching and evaluation Java methods have to be replaced.
- Existing Shibboleth AAPs, which are based on a simple XML files, can automatically be converted to XACML AAPs via an XSLT stylesheet.
- Evaluating and enforcing XACML AAPs has a noticeable performance impact; basic instrumentation of our code revealed that evaluating the AAPs for each requested attribute leads to overhead, which we will investigate to eliminate by caching the compiled XACML `PolicySets` in our future work.
- Due to the lack of a dedicated graphical user interface, we currently have to set up and maintain the XACML AAPs manually, which is tedious and error-prone. A web-based GUI for intuitive ARP and AAP configuration will be developed as part of a student project. Clearly, such an interface must be available to hide the underlying complexity from the users before we can switch to production use.

Furthermore, our web portal already allowed to specify basic ARPs, which were not based on XACML yet. The usage of this functionality has shown that less than ten percent of all users have modified their ARP settings. As a consequence, we believe that

- users have to be sensitized for privacy aspects and made aware of their rights and options.
- defaults, which are suitable for most users, have to be set up.
- intuitive and well-documented GUIs must be available, which allow setting up, maintaining, test and comprehend these policies.

We will contribute future versions of our implementation and tools to the Shibboleth project.

6 Conclusion and Outlook

In this paper, we have argued for an integrated approach to privacy-aware identity management on both the user and the service provider side. Based on the need to automate the decision making about information release and acceptance, we derived the suitability of policy-based management from a discussion of requirements on both sides. The novelty of our approach lies in the use of a common policy language for both Attribute Release and Attribute Acceptance Policies. We specified the required policy elements and the evaluation workflow in general and extended a prototype developed in previous work, which now allows us to use the standard XACML policy language for ARPs and AAPs within the Shibboleth software framework. Finally, we shared some preliminary experiences and outlined our next steps concerning the enhancement of performance and usability aspects.

Our future research will focus on the Service Provider side; especially, we will analyze the interface of AAPs to identity & access management systems in order to seamlessly integrate FIM workflows into the local business processes, and strive for a tighter coupling of AAPs with privacy policy enforcement frameworks presented in related work.

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. Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, pseudonymity, and identity management – a proposal for terminology. In: *Lecture Notes in Computer Science*, Volume 2009, Springer (2000) 1–9
2. Bonatti, P.A., Samarati, P.: Regulating Service Access and Information Release on the Web. In: *Proceedings of CCS 2000*, Athens, ACM Press (2000)
3. Camenisch, J., Shelat, A., Sommer, D., Fischer-Hübner, S., Hansen, M., Krase-mann, H., Lacoste, G., Leenes, R., Tseng, J.: Privacy and identity management for everyone. In: *1st conference on Digital Identity Management*, ACM Press (2005)
4. Bhargav-Spantzel, A., Squicciarini, A., Bertino, E.: Establishing and protecting digital identity in federation systems. TR 2005-48, Purdue University (2005)
5. Powers, C., Schunter, M.: Enterprise Privacy Authorization Language, W3C submission. <http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/> (2003)
6. Karjoth, G., Schunter, M., Waidner, M.: The Platform for Enterprise Privacy Practices — Privacy-enabled Management of Customer Data. In: *Proceedings of the Workshop on Privacy Enhancing Technologies*, Springer (2002)
7. Mont, M.: Dealing with privacy obligations in enterprises. Technical Report HPL-2004-109, HP Laboratories Bristol (2004)
8. Reagle, J., Cranor, L.F.: The Platform for Privacy Preferences. In: *Communications of the ACM*. Volume 42., ACM Press (1999) 48–55
9. Langheinrich, M. (Ed.): A P3P Preference Exchange Language — APPEL 1.0. <http://www.w3.org/TR/P3P-preferences/> (2002)
10. Damiani, E., di Vimercati, S.D.C., Fugazza, C., Samarati, P.: Semantics-aware privacy and access control: Motivation and preliminary results. In: *Proceedings of 1st Italian Semantic Web Workshop*. (2004)
11. Baker, M., Apon, A., Ferner, C., Brown, J.: Emerging grid standards. *IEEE Computer Journal* (2005) 43–50
12. Allison, C., et al.: Integrated user management in the european learning grid. <http://www.hlr.de/publications/> (2005)
13. Hommel, W., Reiser, H.: Federated Identity Management in B2B Outsourcing. In: *Proceedings of the 12th Annual Workshop of the HP OpenView University Association (HPOVUA 2005)*, Porto, Portugal, ISBN 972-9171-48-3 (2005)
14. Linn, J. (Ed.): *Liberty Trust Models Guidelines* (2003)
15. Cantor, S.: Shibboleth v1.2 Attribute Release Policies. <http://shibboleth.internet2.edu/guides/deploy-guide-origin1.2.html#2.e.> (2004)
16. Goldberg, I.: A Pseudonymous Communications Infrastructure for the Internet. PhD thesis, University of California, Berkeley (2000)
17. Koch, M.: Global identity management to boost personalization. In: *9th Research Symposium on Emerging Electronic Markets*. (2002) 137–147
18. Pashalidis, A., Mitchell, C.: A taxonomy of single sign-on systems. In: *Lecture Notes in Computer Science*, Volume 2727, Springer (2003)
19. Pfitzmann, B.: Privacy in browser-based attribute exchange. In: *ACM Workshop on Privacy in Electronic Society (WPES 2002)*, ACM Press (2002) 52–62
20. Josang, A., Pope, S.: User Centric Identity Management. In: *Proceedings of AusCERT 2005*. (2005)
21. Hommel, W.: An Architecture for Privacy-aware Inter-domain Identity Management. In: *Proceedings of the 16th IFIP/IEEE Distributed Systems: Operations and Management (DSOM 2005)*, Barcelona, Spain (2005)

22. Aarts, R., et al.: Liberty architecture framework for supporting Privacy Preference Expression Languages (PPELs). Liberty Alliance White Paper (2003)
23. Ahn, G.J., Lam, J.: Managing Privacy Preferences for Federated Identity Management. In: 1st Workshop on Digital Identity Management, ACM Press (2005)
24. Koch, M., Möslin, K.: Identities management for e-commerce and collaboration applications. International Journal of Electronic Commerce (IJEC) (2005)
25. Nazareth, S., Smith, S.: Using SPKI/SDSI for Distributed Maintenance of Attribute Release Policies in Shibboleth. Technical Report TR2004-485, Department of Computer Science, Dartmouth College, Hanover, HN 03744 USA (2004)
26. Hommel, W.: Using XACML for Privacy Control in SAML-based Identity Federations. In: Proceedings of the 9th Conference on Communications and Multimedia Security (CMS 2005), Salzburg, Austria (2005)
27. Mont, M., Thyne, R., Bramhall, P.: Privacy Enforcement with HP Select Access for Regulatory Compliance. Technical Report HPL-2005-10, HP Bristol (2005)
28. Cantor, S., Carmody, S., Erdos, M., Hazelton, K., Hoehn, W., Morgan, B.: Shibboleth Architecture, working draft 09. <http://shibboleth.internet2.edu/> (2005)
29. Pfitzmann, B., Waidner, M.: BBAE — a general protocol for browser-based attribute exchange. Technical Report RZ 3455, IBM Research, Zürich (2002)
30. Aarts, R. (Ed.): Liberty ID-WSF Interaction Service Specification (2004)
31. Choi, H.C., et al.: A privacy protection model in id management using access control. In: Proceedings of ICCSA 2005, Springer (2005) 82–91
32. 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, IEEE Press (2005)
33. Chadwick, D., Otenko, A.: The PERMIS X.509 Role Based Privilege Management Infrastructure. In: 7th ACM SACMAT, ACM Press (2002)
34. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The ponder policy specification language. In: Lecture Notes in Computer Science, Volume 1995. (2001)
35. ContentGuard Holdings Inc.: XrML 2.0 Technical Overview. <http://www.xrml.org/reference/XrMLTechnicalOverviewV1.pdf> (2002)
36. Moses, T. (Ed.): OASIS eXtensible Access Control Markup Language 2.0, core specification. OASIS XACML Technical Committee Standard (2005)
37. 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)
38. Lorch, M., Kafura, D., Shah, S.: An XACML-based Policy Management and Authorization Service for Globus Research Resources Work in Progress Draft Paper. Department of Computer Science, Virginia Tech (2004)
39. Wu, J., Periorellis, P.: Authorization-Authentication Using XACML and SAML. TR CS-TR-907, University of Newcastle, UK (2005)
40. Vullings, E., Buchhorn, M., Dalziel, J.: Secure Federated Access to GRID applications using SAML/XACML. Tr, Macquarie University, Sydney (2005)
41. Lopez, G., Gomez, A., Marin, R., Canovas, O.: A Network Access Control Approach Based on the AAA Architecture and Authorization Attributes. In: 19th IEEE Int. Parallel and Distributed Processing Symposium, IEEE Press (2005)
42. Proctor, S.: Sun's XACML implementation. <http://sunxacml.sf.net/> (2004)
43. Crane, S., Mont, M., Pearson, S.: On helping individuals to manage privacy and trust. Technical Report HPL-2005-53, HP Laboratories Bristol (2005)