Policy-based Service Provisioning and dynamic Trust Management in Identity Federations

Wolfgang Hommel
Leibniz Supercomputing Center Munich
hommel@lrz.de

Latifa Boursas
Technical University of Munich
boursas@tum.de

Abstract— **In Federated Identity Management (FIM), user administration is decentralized: Service Providers (SPs) can request information about the users from their respective Identity Providers (IDPs). The subsequent processing of this data with respect to service provisioning and various privacy aspects are open research issues. We first specify how SPs can use provider-wide and service-specific XACML policies to enforce the required quality for the data delivered by the IDPs. Then, we demonstrate how aspects of trust and reputation management can improve the dynamics of Identity Federations and enhance the end users' privacy. We also extend the identity-centric request-response model of today's FIM protocols by group queries and demonstrate their application. Finally, we introduce our prototype and its integration into the Shibboleth FIM software.**

ICC 2006 Category: Information Assurance Track

I. Introduction

Many of the challenges of authentication and authorization in large distributed systems, such as for supply chains in industry and Grid projects in academia, have been successfully met by using *Federated Identity Management (FIM)* techniques. The basic idea of FIM is to distinguish between *Identity Providers (IDPs)*, which make information about their local users available, and *Service Providers (SPs)*, which can request this information and deduce authorization decisions from it. Typically, users are modelled as objects with various attributes, which are then exchanged between IDP and SP; this is known as *Attribute Based Access Control (ABAC)*.

ABAC is successful because of its universality: First, exchanged attributes may, but need not, personally identify the user. Thus, systems, in which it is too complex to manage all users and their rights individually, can rely on Attribute Certificates (ACs), which state certain rights and are signed by a trusted Attribute Authority (AA). This also allows, if desired, the anonymous usage of services while still guaranteeing that the users have been authenticated and authorized. Second, attributes may also be used to express the *roles* in which users are acting; under the assumption of an a priori arranged mutual understanding of the semantics of those attributes, this results in an efficient way of setting up the *Role Based Access Control (RBAC)* model in large distributed systems.

Various approaches to distributed authentication, typically resulting in a *single sign-on* experience for the end users, and distributed authorization have been discussed in the past few years. While industrial standarization efforts, such as the *Security Assertion Markup Language (SAML, [1])* and the *Liberty Alliance Project (LA, [2])*, often focus on e-commerce and business-to-business scenarios in which so-called Identity Federations are set up, academic research has concentrated on the application of well-established privilege management infrastructures, such as PERMIS [3], in the area of Grid computing and Virtual Organizations. Both SAML and the *eXtensible Access Control Markup Language (XACML, [4])*, an XML-based access control policy language, have been widely adopted in both industrial and academic research and projects (for a survey, see [5]; details can be found in [6], [7], [8]).

Research has also started to investigate privacy aspects of FIM and ABAC. Obviously it is crucial for legal conformity and user acceptance to provide methods to control and restrict which information about users is given out to which service providers by their identity provider ([9], [10], [11], [12]). The Liberty Alliance and the SAML-based Shibboleth software [13] have coined the term *Attribute Release Policies (ARPs)* for these mechanisms, and policy-based management is an acknowledged choice in this area ([14], [15]). In previous work we have shown deficiencies of current ARP implementations ([16], [17]) and why XACML is an excellent choice for the modelling and enforcement of ARPs ([18]).

In this paper, we report further results of our ongoing research in the area of FIM and ABAC. Based on a scenario presented in section II, the use of XACML in Attribute Acceptance Policies (AAPs), which are the counterpart to ARPs on the service provider side, is investigated in section III. An integration of trust and reputation management techniques in order to enable a more dynamic setup of federations and virtual organisations is presented in section IV. In section V, we discuss the necessity of FIM group queries, i.e. the request of information not only about an individual user, but also the groups of which she is a member as well. Our prototypical implementation and its integration into the Shibboleth [13] architecture is introduced in section VI. Finally, section VII outlines our next steps and further research.

II. Identity Federation Scenario

The scenario presented here is used to illustrate the application of the techniques discussed in the subsequent sections. As can be seen in figure 1, our Identity Federation is based on a Virtual University (VU) that provides e-learning materials to the students of the member universities (U1–U4), which contribute to the VU. The VU operates several content and streaming servers of its own, but also has contracts with third party content providers (CP1, CP2), as well as with online bookstores (B1, B2), which offer literature relevant to the offered e-learning courses at a special discount rate.

In FIM terms, the member universities are the students' identity providers (IDPs). By using a FIM technology, such as SAML, a student can log into her home university's web portal, typically by providing a username and password, and then use the services of the federated service providers (SPs) without having to reauthenticate there—the IDP vouches for the user and the SP trusts the IDP to have authenticated the user properly, resulting in cross-organizational single sign-on.

Besides this authentication information, further attributes such as the user's study course could be requested by the VU from the respective member university. The authorization to sign up for certain e-learning courses could be deduced, e.g. from the user's study course. In this case, the VU would be the *policy decision point (PDP)*. Alternatively, the IDP, i.e. the user's home university, could act as PDP and decide which of its students is entitled to take which VU courses, so the policy decision can be arbitrarily distributed and finally the VU is the *policy enforcement point (PEP)* which grants or denies the access.

The trust relationship between the VU and the member universities is extended to the third party content providers and the bookstores. If the user wants to access material which is stored at one of the external content providers, FIM protocols again take care of authentication and authorization: The user does not have to reauthenticate at the content provider's site, because the VU vouches that the user is entitled to access the material and the content provider trusts the VU regarding this assertion. Thus, the VU is an *Attribute Authority (AA)*, i.e. it is the authoritative data source for additional information besides the information about the user which can be requested from its IDP. Regarding the online bookstores, the VU is an AA which asserts that the user is really taking a certain e-learning course and thus entitled to buy the literature at the discount rate.

Obviously, the different SPs in this scenario require different subsets of the user's personal information. For example, the VU needs to know the user's study course, but not her credit card number or address. On the other hand, if the user is buying the recommended literature at the discount rate, the online bookstore is entitled to receive such payment and shipping details. Attribute Release Policies (ARPs) are used to control this flow of personal information, and although ARP support is not mandatory, e.g. for SAML conformance, various implementations exist ([19], [20], [18]).

However, current ARPs are static; users set them up a priori or on demand, i.e. they are informed online about SPs attempting to access their information, and then the ARPs will be rarely changed. Furthermore, the trust relationships described above were strictly between organizations, such as the universities and bookstores in the scenario, and relatively static, because they are typically based on formal contracts and service level agreements. We want to extend this trust management to become more dynamic and also include the end users of the distributed system. For example, if there are multiple online bookstores, which one should the user choose? Can this decision, for example, be based on the choices the user's friends have made before at the same IDP? Also, does an SP have to accept all users from the federation's IDPs? How can fraudulent transactions be avoided to prevent financial harm?

In the next section, we define technical requirements for SP-side Attribute Acceptance Policies (AAPs) and their realization using XACML policies. Then, in section IV, we describe how trust and reputation management can be incorporated into both ARPs and AAPs.

III. Using XACML for Attribute Acceptance Policies

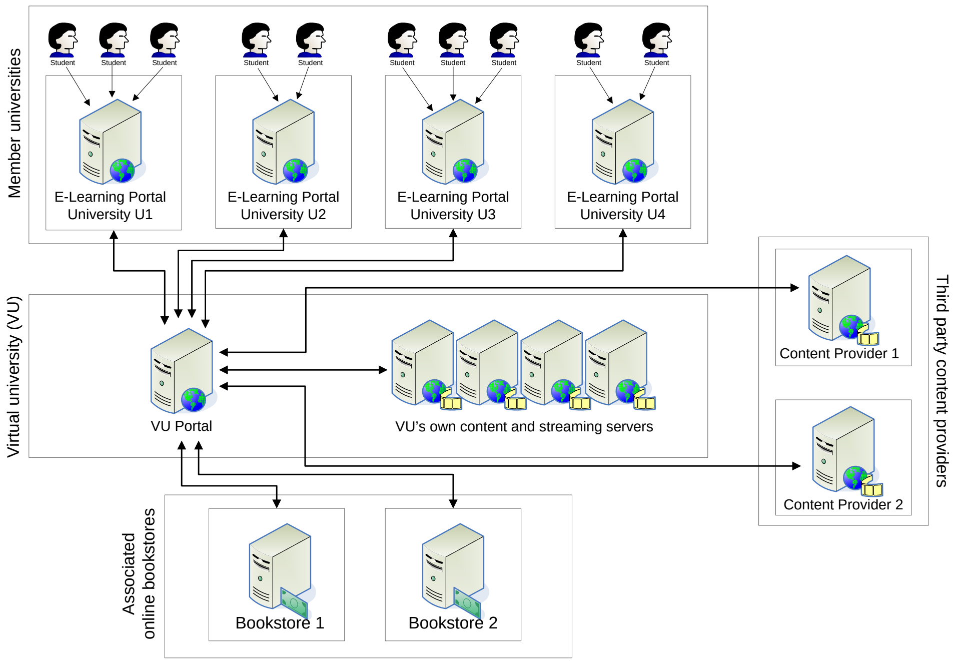
In general, providing services requires information about one's customers and end users; we are intentionally omitting services for anonymous usage in this section. By using FIM protocols, such as SAML, arbitrary user attributes can be requested. This request is successful if a) the attribute is known to the identity provider, b) the user object actually has set a value for this attribute and c) the Attribute Release Policies permit the transmission of the attribute to the service provider.

Still, receiving an attribute's value might not be sufficient for the provisioning of a service. In FIM terms, *provisioning* refers to the delivery of user information to the service for account creation and maintenance purposes. The service provider typically has to check first a) whether all the required attributes have been received and b) whether each attribute has a valid value. These checks are done by enforcing so-called Attribute Acceptance Policies (AAPs).

AAPs are not mandatory components of standards compliant implementations; thus, only few implementations exist. Shibboleth's built-in AAP language supports the formulation of rules; each rule can specify the name of an attribute, the IDP which the rule applies to and the decision to accept or deny the attribute based on its value; the value must be specified either literally or as regular expression.

However, we are striving for a more general approach, which especially supports the provisioning of several services with different requirements on the service provider side better. As can be seen from figure 2, we are going to provide identity information to multiple services through a single FIM channel instead of having to FIM-enable each service individually. We also want to be able to group services, e.g. based on their internal dependencies (such as a web server which needs an FTP server so users can upload their content) or on combinations of services which customers are frequently using.

We thus apply both site-wide and service-specific AAPs on the service provider side in analogy to site-wide and user-specific ARPs on the identity provider side. The site-wide AAPs specify which attributes must be presented to each of the provider's services as well as the criteria they must fulfill.



Member universities
Student
E-Learning Portal University U1
E-Learning Portal University U2
E-Learning Portal University U3
E-Learning Portal University U4
Virtual university (VU)
VU Portal
VU's own content and streaming servers
Third party content providers
Content Provider 1
Content Provider 2
Associated online bookstores
Bookstore 1
Bookstore 2

Fig. 1. Virtual University Identity Federation Scenario

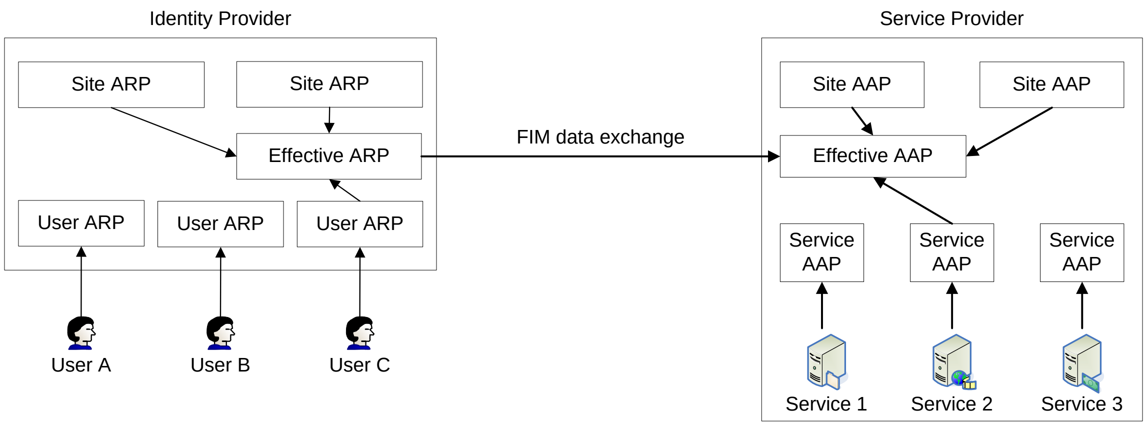


Fig. 2. Attribute Release and Acceptance Policies on Identity and Service Provider Side

These default settings can then be customized by service-group and service specific AAPs. The centralization of AAP checking for all services provided by the SP has several advantages in the practical setup and operation of FIM-enabled services:

1) From a management perspective, splitting the AAPs into site-wide and service-specific policies allows decentralizing the management of the centrally enforced policies and thus enables the delegation of administrative rights. This reflects the typical situation in which there are organization-wide defaults, division-specific refinements thereof and finally service-specific settings made by the local administrators. Most modern policy languages support decentralized administration, so this approach does not suffer from a high implementation overhead.
2) The integration into local business processes, such as for accounting, is smoother because it is not necessary to implement these checks in each service separately. Instead, the service can rely on the presence of all required attributes, just as if they were provided by a local identity management system and not through FIM. This is especially important when legacy, i.e. non-FIM-enabled, services shall be used in identity federations.
3) Besides guaranteeing the provisioning of the minimum required information to operate the service, AAPs can also be used to filter attributes, e.g. for privacy protection purposes. For example, if certain personally identifiable information is delivered by the identity provider, al-

though it is not required for using the service, this data can be discarded on its way to the service.
4) Missing attributes can be immediately reported back to the identity provider by the service provider. As a consequence, the identity provider could ask the user interactively to agree to the release of these attributes; this would prevent error messages from the service directly to the user. Such a procedure has been suggested before by B. Pfitzmann for situations in which setting up Attribute Release Policies a priori is too complex for the user [11].

A policy language for AAPs must fulfill various criteria similar to those for ARPs; we have discussed them in detail in previous work [18]. While several policy languages, such as Ponder [21] and XrML [22], could also be used for this task, we want to carry the successful application of XACML in various Grid, privilege management and identity management projects over to AAPs. We thus specified the semantics of XACML-based AAPs and demonstrate them based on the scenario described in section II:

- The subjects in XACML policies are the end users who want to use a service. They can be identified individually, which typically leads to too much administrative overhead, based on their affiliation with identity providers or depending on the values of some of their attributes, e.g. all students pursuing certain study courses.
- The XACML resources are the services, or groups of services, which are to be provisioned. They are typically identified by a federation-wide unique id.
- XACML actions specify the various possibilities to use a service; for many services this will typically be one, constant value, although arbitrary distinctions can be made, e.g. read and write access to the database which stores the e-learning material.
- XACML obligations can be used to inform the service administrators about special events via email or logfiles; for example, the online bookstore in our scenario might want to keep a logfile which contains information about how many of the books were sold at the discount rate.
- XACML conditions can be used to filter attributes, based on both their names and their values. For example, the user's study course might be discarded for privacy reasons and not be delivered to the service itself, even if the IDP-side ARPs were set up incorrectly, because this data should not have been sent to the SP in the first place.

Such policies must be defined for each group of services or each individual service. Figure 3 shows an example of an XACML-based AAP; it is used by the SP to reject credit card numbers whose expiry date has been exceeded.

As both Shibboleth's built-in and XACML-based AAPs are stored in XML format, a lossless conversion of Shibboleth's AAPs to XACML-based AAPs can be implemented and automated, e.g. using an XSLT stylesheet.

```
<Policy id="SiteARP_CC" RuleCombiningAlg="first-applicable" prio="100">
  <Description> Reject expired credit cards </Description>
  <Rule id="Check_CC_expiry" effect="deny">
    <Target>
         <AnyResource/> <!-- apply to all services -->
         <AnyAction/>   <!-- apply to all actions  -->
         <!-- apply to all users with specified credit card number -->
         <SubjectMatch AttributeValue="\d+">
           <SubjectAttributeDesignator AttributeId="CreditCardNumber"/>
         </SubjectMatch>
    </Target>
    <Condition FunctionId="time-greater">
         <EnvironmentAttributeSelector AttributeId="current-time">
         <SubjectAttributeDesignator AttributeId="CreditCardExpiry"/>
    </Condition>
    <Obligation Id="Log" FulfillOn="Deny">
        <AttributeAssignment Id="text">
          Expired credit card number rejected, user:
          <SubjectAttributeDesignator AttributeId="subject-id" />
        </AttributeAssignment>
    </Obligation>
  </Rule>
  <Rule id="acceptOtherwise" effect="permit"/>
</Policy>
```

Fig. 3. Example XACML Attribute Acceptance Policy

IV. Incorporating Trust and Reputation Management

Trust is obviously the root of identity federations, because service providers (SPs) rely on the information delivered by the identity providers (IDPs). In today's federations, trust between organizations is typically established on a contractual basis and thus quite static. Many technical aspects of trust management, such as reliably authenticating and authorizing users and machines, are handled by FIM protocols; for example, SAML assertions certify a user's attributes and SSL as well as XML signatures are used for mutual authentication and integrity checking.

However, we want to achieve more dynamic trust relationships and especially also involve the users more directly. For example, a student might not be willing to entrust her credit card data to a certain online shop just because it is member in the virtual university's federation; then again, she might do so if at least five of her friends already did. On the other hand, while the online shop agrees to deliver certain books at a discount rate to authorized students, it is not obliged to accept a new customer which is suspected to be fraudulent, e.g. based on consumer credit rating data reported by a third party scoring service.

We distinguish between two kinds of trust which complement each other:

1) Static trust, which defines a basic trust level and is typically based on a contract between organizations or on a recommendation made by an already trusted authority.
2) Dynamic trust, which primarily reflects one's own experience and secondarily also incorporates the experience of peers with the other party; this is known as reputation management.

Various algorithms exist for the calculation of dynamic trust and the determination of the total trust. These are based on the static and dynamic components. For our purpose here, it is sufficient to assume that an individual *trust level* can be specified for each ordered tuple of principals which are involved in a federation. The user's trust in a service is therefore not necessarily the same as this SP's trust in the user. In our scenario,

- the student's dynamic trust in a service will depend, for example, on
  - her acceptance of the service's terms of use, privacy policies and shipping conditions as stated on its web site,
  - her previous experience with this service and other services of the same SP,
  - her friends' and fellow students' trust in this service.
- the SP's dynamic trust in a user will usually depend on
  - its previous experience with the user, such as unpaid bills or delivery rejections,
  - its previous experience with the user's identity provider, such as low data quality or a high number of fraudulent transactions,
  - consumer ratings by third party scoring services.

We assume that the trust level is expressed as an integer value between -100 and 100, with negative values meaning a lack of trust and 0 being neutral. We can then use *thresholds* in both user ARPs and service AAPs:

- Users can specify a default ARP which, for example, permits the release of their credit card data and shipping address, if the purpose of the incoming attribute request is the handling of an actual product order and the trust level for the requesting service is $\geq n$.
- SPs can restrict untrusted users' actions by appending a term to the XACML condition which compares the trust level with the threshold value; for example, browsing the online bookstore may be unrestricted, but ordering books is limited to users with a non-negative trust level.

The user's trust in the services can be stored as a multi-valued attribute at the IDP; frontends which allow the user to edit these trust settings must be provided and could be integrated with ARP editors. The actual trust level, which is calculated from the various static and dynamic trust components, must be calculated by the IDP's engine dynamically at runtime. Decisions made on such calculation results and their varying input values should be logged for auditing purposes.

Similarly, the SP will store its experience with the user in its customer records; further data, such as consumer ratings, are usually retrieved online from third party service providers.

V. Extending FIM Requests to Group Objects

Presently, FIM protocols allow service providers (SPs) to request user attributes from the identity provider (IDP) based on a handle; this handle is typically opaque, meaning that no personally identifiable information can be derived from the handle itself, while it still is an identifier for exactly one user.

Yet, various purposes do not require only user individual attributes, but also information about the groups a user is in. Consider the following examples:

- Non-public information about a new drug for a severe disease shall be made available to patients *and* their relatives. While the patient record specifies the disease in an attribute which can be used to control the patient's access, typically no suitable attributes exist for the patient's relatives. Instead, a group could be specified in which the patient and her relatives are members, and access to the drug information could be given to anyone in such a group.
- E-commerce sites, such as the bookstores in our scenario in section II, typically attempt to increase their attractiveness to users by means of personalization. If the teacher of an e-learning class is browsing a bookstore which knows the number of students in this class, it could recommend didactics textbooks for teachers, which are appropriate to the size of the class.
- A group's accounting and shipping information may be different from that of the individual members. For example, if a purchase is made on behalf of a group, the group's (not the buyer's) information should be used.

On the IDP side, a group is typically an object, whose attributes are the list of group members and metadata, which can be explicit, such as who created the group and who is allowed to administer it, or implicit, e.g. the number of members the group has. Futhermore, each user object has a multi-valued attribute which lists the groups this user is a member of. This sort of double-chaining is not unusual for local identity management systems and is required to answer the following questions efficiently:

- In which groups is user u member?
- Which members does group G have?

The group names, as stored in user objects, become the handles for the SP to request information about these groups from the IDP.

Access to the group object's attributes is again controlled by Attribute Release Policies, which can be maintained by the group administrators. Optionally, users may have to express their consent to being listed as group members, depending on the federation scenario. If, for example, groups are mapped to roles, and some roles are mutually exclusive, users must not suppress being listed in the groups. This allows the SP to enforce the *separation of duties*. For example, someone who is responsible for granting loans at a bank must not be allowed to request a loan from himself.

VI. Implementation and practical aspects

We are using Sun's open source XACML PDP [23] for our implementation of XACML-based Attribute Acceptance Policies (AAPs) as described in section III. Our prototype resembles an XACML PEP which creates XACML requests from incoming SAML assertions, lets the PDP evaluate each asserted attribute against the AAPs and takes care of the post-processing of the PDP's XACML response, such as handling XACML obligations and returning the accepted attributes to the SAML PDP for service provisioning.

We plan to use the command-line based version of our prototype for web-based XACML policy administration points, which the service provider administrators can use to maintain their AAPs. Implementing XACML AAPs for Shibboleth service providers (called Shibboleth Targets in version 1.2) requires modifications to the source code; unlike for custom identity repository data connectors, Shibboleth does not offer extension or hook mechanisms for third party AAP implementations. Thus, the AAP implementation, a part of Shibboleth's `Metadata` component, has to be replaced. Fortunately, the API is straightforward, so replacing the `AAP::apply` method with our XACML PEP implementation is the central integration task. We plan to contribute an extension to the upcoming release of Shibboleth 2.0 which allows for switching between multiple AAP implementations and includes our XACML policy-based one.

From a practical point of view, while the enhanced functionality we have introduced certainly allows a better and more fine-grained access control and privacy protection on both service and identity provider side, its drawback is yet another increase in the complexity of security systems. To avoid system failure due to unusability, a suitable default configuration must first be set up by the IDP administrators in order to protect the privacy of those users who do not invest time to do so on their own. Second, intuitive graphical user interfaces must be provided which make policies easy to create, maintain, test, comprehend and debug for both users and service administrators. Finally, errors, such as too restrictive attribute release policies which obviate service usage, must be intercepted online and presented to the user in an interactive dialog which allows them to solve the problem.

VII. Summary and Outlook

In this paper, we first investigated the requirements of provider-wide and service-specific Attribute Acceptance Policies (AAPs) in Federated Identity Management (FIM); we provided arguments for XACML's suitability for the modelling and enforcement of AAPs and specified the AAP-relevant semantics of XACML policies. Second, we have shown how trust and reputation management aspects can be incorporated into both indentity provider (IDP) side Attribute Release Policies (ARPs) and service provider (SP) side AAPs. We have demonstrated the application of *trust levels* in FIM and suggested extensions to both IDP and SP data models to realize them. Third, we introduced the concept of FIM Group Queries; these extend the user-centric attribute request workflow to group objects, which allows enhanced functionality in various FIM applications without requiring modifications of existing FIM protocols and languages. Finally, we have shown the feasibility of our approach by means of a prototype and reviewed our work under complexity and usability aspects.

In the future, we are first going to implement graphical user interfaces, on the one hand for AAP administration on the SP side, on the other hand for end user trust level management on the IDP side. Second, we plan to contribute our XACML AAP implementation to the upcoming Shibboleth 2.0 release and will gather feedback for our approach in a real world project. Finally, we are going to analyze the use of attribute-conversion techniques to protect the user's privacy by means of depersonalizing transformations.

References

[1] S. Cantor, J. Kemp, R. Philpott, and E. Maler (Eds.), "Assertions and Protocols for the Security Assertion Markup Language (SAML) V2.0," OASIS Security Services Technical Committee Standard, Mar. 2005.

[2] T. Wason (Ed.), "Liberty ID-FF Architecture Overview v1.2," Liberty Alliance Specification, 2004.

[3] D. Chadwick and A. Otenko, "The PERMIS X.509 Role Based Privilege Management Infrastructure," in *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT. ACM Press, 2002, pp. 135–140.

[4] T. Moses (Ed.), "eXtensible Access Control Markup Language 2.0, core specification," OASIS XACML Technical Committee Standard, 2005.

[5] M. Lorch, S. Proctor, R. Lepro, D. Kafura, and S. Shah, "First Experiences Using XACML for Access Control in Distributed Systems," in *ACM Workshop on XML Security*. ACM Press, 2003.

[6] R. Lepro, "Cardea: Dynamic Access Control in Distributed Systems," NASA Advanced Supercomputing Division, Ames, Tech. Rep. NAS-03-020, 2003.

[7] P. Mazzuca, "Access Control in a Distributed Decentralized Network: An XML Approach to Network Security," Honors Thesis, Dartmouth College, 2004.

[8] M. Lorch, D. Kafura, and S. Shah, "An XACML-based Policy Management and Authorization Service for Globus Research Resources (draft)," Department of Computer Science, Virginia Tech, 2004.

[9] K. Bohrer, X. Liu, D. Kesdogan, E. Schonberg, M. Singh, and S. Spraragen, "Personal Information Management and Distribution," in *4th International Conference on Electronic Commerce Research ICECR-4*, 2001.

[10] P. A. Bonatti and P. Samarati, "Regulating Service Access and Information Release on the Web," in *CCS 2000, Athens*. ACM Press, 2000.

[11] B. Pfitzmann, "Privacy in browser-based attribute exchange," in *Proceedings of the ACM Workshop on Privacy in Electronic Society (WPES 2002)*. ACM Press, 2002, pp. 52–62.

[12] M. Mont, "Dealing with privacy obligations in enterprises," HP Laboratories Bristol, Tech. Rep. HPL-2004-109, 2004.

[13] S. Cantor, S. Carmody, M. Erdos, K. Hazelton, W. Hoehn, and B. Morgan, "Shibboleth Architecture, working draft 09," http://shibboleth.internet2.edu/docs/, 2005.

[14] K. Bohrer, S. Levy, X. Liu, and E. Schonberg, "Individualized Privacy Policy Based Access Control," in *Proceedings 6th International Conference on Electronic Commerce Research*, ser. ICECR, 2003.

[15] G. Karjoth, M. Schunter, and M. Waidner, "The Platform for Enterprise Privacy Practices — Privacy-enabled Management of Customer Data," in *Proceedings of the Workshop on Privacy Enhancing Technologies, PET 2002*. Springer Verlag, 2002.

[16] W. Hommel and H. Reiser, "Federated Identity Management: Shortcomings of existing standards," in *Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Management (IM 2005)*, Nice, France, May 2005.

[17] W. Hommel, "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, October 2005, in press.

[18] ——, "Using XACML for Privacy Control in SAML-based Identity Federations," in *Proceedings of the 9th Conference on Communications and Multimedia Security (CMS 2005)*, Salzburg, September 2005.

[19] S. Cantor, "Shibboleth v1.2 Attribute Release Policies," http://shibboleth.internet2.edu/guides/deploy-guide-origin1.2.html#2.e., 2004.

[20] S. Nazareth and S. Smith, "Using SPKI/SDSI for Distributed Maintenance of Attribute Release Policies in Shibboleth," Department of Computer Science, Dartmouth College, Hanover, HN 03744 USA, Tech. Rep. TR2004-485, 2004.

[21] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The Ponder Policy Specification Language," *Lecture Notes in Computer Science*, vol. 1995, 2001.

[22] ContentGuard Holdings Inc., "XrML 2.0 Technical Overview," http://www.xrml.org/reference/XrMLTechnicalOverviewV1.pdf, 2002.

[23] S. Proctor, "Sun's XACML PDP," http://sunxacml.sf.net/, 2004.