User-centric Privacy Management for Federated Identity Management

Gail-Joon Ahn and Moonam Ko
College of Computing and Informatics
The University of North Carolina at Charlotte
{gahn,mnko}@uncc.edu

***Abstract*—We have witnessed that the Internet is now a prime vehicle for business, community, and personal interactions. The notion of identity is the important component of this vehicle. Identity management has been recently considered to be a viable solution for simplifying user management across enterprise applications. The network identity of each user is the global set of personal credentials and preferences constituting the various accounts. The prevalence of business alliances or coalitions necessitates the further evolution of identity management, named federated identity management (FIM). The main motivation of FIM is to facilitate the federation of identities among business partners emphasizing on ease of user management. In this paper, we propose systematic mechanisms to specify privacy preferences in FIM, attempting to help users facilitate preferences for managing their private information across domains.**

I. Introduction

As enterprises have changed their business operation paradigm from brick-and-mortar to click-and-mortar, they have embraced a variety of enterprise applications for streamlining business operations such as emailing systems, customer relationship management systems, enterprise resource planning systems, supply chain management systems, and so on. However, a non-trivial problem has been compounded by this reinforcing line of enterprise applications, *the problem of managing user profiles*. The addition of such applications has proved to be subject to bringing in a new database for storing user profiles and it was quite costly and complex to manage all those profiles, which were often redundant. Considering business-to-business environments, where a set of users consists of not only their employees or customers but also those of their partners, the above-mentioned problem became even worse. As a set of underlying technologies and processes overarching the creation, maintenance, and termination of user identities, identity management (IM) has been recently considered to be a viable solution for resolving such issues.

Furthermore, the prevalence of business alliances or coalitions necessitates the further evolution of IM, so called federated identity management (FIM). The main motivation of FIM is to enhance user convenience and privacy as well as to decentralize user management tasks through the federation of identities among business partners. As a consequence, a cost-effective and interoperable technology is strongly required in the process of federation. Web Services (WS) can be a good candidate for such requirement as it has served to provide the standard way for enabling the communication and composition of various enterprise applications over distributed and heterogeneous networks [1], [2].

Since identity federation is likely to go along with the exchange of sensitive user information in a highly insecure online environment, security and privacy issues associated with such exchanges are key concerns in FIM. The concept of federated identities provides the consumers with a convenient way to create identities and move among various business nexus. Apart from all the simplicity and convenience that it provides the businesses with, the management of these federated identities becomes a crucial task since it needs to take into consideration various threats against the vulnerable and confidential user data. Any identity management framework must adequately protect sensitive user information and must adhere to important elements of privacy policy. In this paper, we propose systematic mechanisms to specify privacy preferences in FIM, attempting to help users facilitate preferences for managing their private information across domains.

The rest of this paper is organized as follows. Section II overviews three approaches involved in IM and discusses the prior research works in IM followed by an overview of FIM models. Section III articulates business scenarios for FIM and relevant privacy requirements. In addition, we discuss our approach to support multi-level privacy policy framework using privacy labels and proposes languages for privacy policy and privacy preference expression along with the related works. Section IV concludes this paper.

II. Identity Management

In this section, we first start with the discussion of IM approaches. We categorize IM approaches into the following three styles: *isolated IM*, *centralized IM*, and *distributed IM*. Thereafter, we discuss the related research works followed by FIM.

The isolated IM model is the most conservative approach of the three models. Each business forms its own identity management domain (IMD) and has its own way of maintaining the identities of users including employees, customers, and partners. Hence, this model is simple to implement and has a tight control over user profiles. However, it is hard to achieve user convenience with this model since different IMDs are likely to have different authentication processes or mechanisms for their users and corresponding authentication policies may vary between players.

The centralized IM model has a single identity provider (IDP) that brokers trust to other participating members or service providers (SP) in a Circle of Trust (CoT). IDP being a sole authenticator has a centralized control over the identity management task, providing easy access to all SP domains with simplicity of management and control. The drawback of this approach is a single point of failure within a CoT infrastructure in case that IDP fails to provide authentication service. User convenience can be also achieved partially in case where the single sign-on (SSO) for users is only effective within SPs which belong to the same CoT.

The distributed IM model provides a frictionless IM solution by forming a federation and making authentication a distributed task. Every member agrees to trust user identities *vouched for* by other members of the federation. This helps users maintain their segregated identities, making them portable across autonomous policy domains. It also facilitates SSO and trust, thereby allowing businesses to share the identity management cost with its partners. Microsoft Passport is based on the centralized IM model, while Liberty Alliance aims to be the distributed IM model.

Earlier works related to user identity management were mostly focused on a user-centric approach [11], where users have control over IM functions. A simple idea of managing user identities is described in [7]. They proposed the use of personal card computers to handle all payments of a user, thereby ensuring the privacy and security of the user's identity on the Web. Hagel and Singer [14] discussed the concept of *infomediaries* where users have to trust and rely on a third party to aggregate their information and perform IM tasks on their behalf while protecting the privacy of their information. The Novell digitalme technology [10] allows users to create various identity cards that can be shared on the Internet according to users' preferences. Users can control both what information is stored in each card and conditions under which it may be shared.

Federated identity gives the ability to securely recognize and leverage user identities owned by trusted organizations within or across CoTs, and identity federation allows organizations to securely share confidential user identities with trusted ones, without requiring users to re-enter their name and password when they access their network resources. Additionally, identity federation provides the ability to optionally and securely share user information such as their profiles or other data between various trusted applications which is subject to user consent and organizational requirements. There are two well-known FIM solutions, Liberty Alliance and Microsoft Passport. These solutions have fundamentally the same goal of managing web-based identification and authentication. Both enable organizations to build IM systems that can federate across many disparate sources. Therefore, each user can have a single network identity that provides SSO to the web sites that have implemented either or both of the systems. In this paper, we mainly focus on Liberty Alliance since Microsoft Passport is no longer available as FIM approach. Liberty Alliance is a consortium of more than 150 companies working together towards developing an open, interoperable standard for FIM [15], [23]. It is aimed towards realizing the notion of a cohesive, tangible network identity, which can facilitate SSO and frictionless business operations. It is a distributed IM model, relying on the notion of IDP and SP, as we discussed earlier. IDP is responsible for carrying out identity federation. Authentication messages or authentication requests are passed between IDP and SP. IDP and SP in Liberty Alliance Model actually facilitate WS to discover service locations and handle incoming messages from other IDP and SP as shown in Figures 1.

III. User-centric Privacy Management

Privacy is a growing concern with FIM models due to the voluminous exchange of sensitive information that occurs across enterprises. Securing communication channels and encrypting messages may help preserve the privacy of relevant information only up to some extent. The security concerns that we discussed in [3], [21], [22] are obviously applicable to privacy as well. In WS-enabled FIM where the receiver of a message may not be its ultimate destination, improper security measures may result in unauthorized access to user's personal information which leads to violation of privacy [16].

Protection of user identities and personal information can be achieved by using the principle of pseudonymity. Obfuscating message payloads can also preserve their privacy by making them accessible only through authorized parties having proper credentials or keys [20]. Privacy enhancing technologies like Platform for Privacy Preference (P3P) [9] provide a solution for point-to-point privacy protection based on user preferences. However, such solutions do not scale for a more open, interoperable WS architecture.

Liberty Alliance's SAML implementation uses pseudonyms constructed using pseudo-random values that have no discernable correspondence with users' identifiers at IDP or SP. The pseudonym has a meaning only in the context of the relationship between the two communicating parties. The intent is to create a non-public pseudonym so as to contravene the linkability to users' identities or activities, thereby maintaining the privacy.

Organizations using FIM models are required to follow four key principles of fair information practices which are discussed in [12] and partially in [5]:

- *Notice*: Users should receive prior notice of the information practices.
- *Choice*: Users have a choice to specify what information will be used and the purpose for which the information is collected.
- *Access*: Users should be able to access and modify their personal information if necessary and when needed.
- *Security*: Users should be assured that the organizational system is capable of securing their personal information.

Liberty Alliance specifications have proposed an approach to sharing user attributes on the basis of user's permission [15],

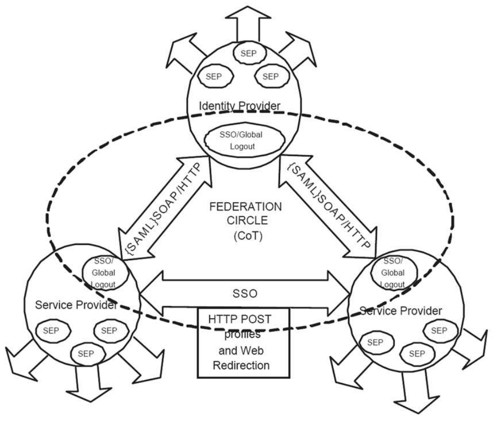


Fig. 1. Liberty Alliance model

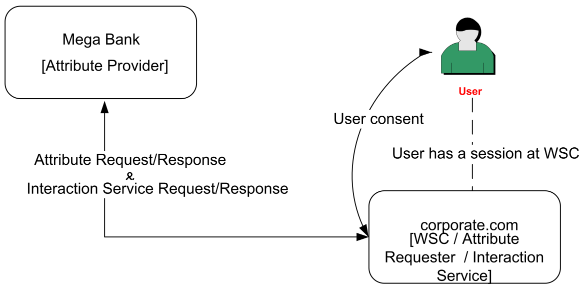

Mega Bank
[Attribute Provider]
User
User consent
User has a session at WSC
Attribute Request/Response
&
Interaction Service Request/Response
corporate.com
[WSC / Attribute Requester / Interaction Service]

Fig. 2. Indirect Interaction Case

[17]. The specifications also provide a set of guidelines that will help businesses adhere to these principles. Microsoft Passport's approach to online privacy is also based on adherence to these aforementioned principles.

Now we describe business scenarios that we utilize to articulate necessary elements in dealing with privacy issues for federated identity management, focusing on Liberty Alliance specifications. Our scenarios have two hypothetical entities: a financial service institution *Mega Bank* that has online banking services and an online stock trading and brokerage company *Corporate.com*. We assume that *Mega Bank* and other Web Services Consumer (WSC) are Liberty enabled entities and recognize each other as the member of their CoT. Also a WSC has the ability to request one or many attributes which may or may not contain Personally Identifiable Information (PII). In addition, all WSCs and *Mega Bank* have a central policy in the Usage Directives and the user has read and agreed to the posted privacy policies at each service provider before signing up with them. Finally, the user has stored her privacy preferences at the Attribute Provider for some or all of her PII.

We now categorize our scenarios from *Mega Bank* perspectives. *Mega Bank* can act as either attribute provider or attribute requester.

1) *Mega Bank* acting as an Attribute Provider [1]
Under this scenario, we identify three different cases based on the interaction service (IS) patterns that initiate a communication channel with a user to obtain the user's consent.

a. *Direct Interaction with the user for obtaining consent*: Mega Bank can initiate an IS for obtaining user consent before actually releasing the attribute to the WSC. The IS instance is initiated in case of a policy level mismatch between user's stored preferences and the policy level for the intended usage.
b. *Indirect Interaction through another WSC*: Mega Bank serves only the attribute request without invoking an IS by itself. The interaction service is invoked by the same WSC who has requested a user attribute. In this case, Mega Bank (Attribute Provider) does not have a direct interaction with the user.
c. *Indirect Interaction through a third-party IS on behalf of the user*: Mega Bank communicates with a third-party IS for obtaining user consent. The third-party IS for the user is discovered using the ID-WSF Discovery service [17]. Mega Bank and the WSC do not have any direct interaction with the user.

[1] *Mega Bank* can also serve as an IDP or can have another IDP in the CoT. However, since the role of an IDP is limited in our scenarios, we omit such cases in this paper.

2) *Mega Bank* acting as an Attribute Requester
   As an Attribute Requestor, Mega Bank sends attribute requests to Corporate.com which provides PII as an Attribute Provider. For this case, Mega Bank invokes an IS to establish a direct interaction with the user for obtaining a consent.

Our study is conducted with actual experimentations of each case. For brevity, this paper focuses on Indirect Interaction case as shown in Figure 2.

A. *Privacy Labels*

Liberty Alliance specifications also address multi-level privacy policy approach [18]. We introduce the concept of *Privacy Labels* to meet such requirement. Privacy labels are somewhat similar to security labels in mandatory access control (MAC). In MAC, every resource or object is tagged with a security label representing the sensitivity of each resource. A subject needs a legitimate security clearance to access resources. Similarly, every CoT needs to create privacy labels based on the nature of its business [2]. The privacy labels are used both in service provider's privacy policy and user's privacy preferences. We first identify the requirements for privacy labels as follows:

- *Privacy labels are hierarchical and comparable.*
  We need to evaluate each attribute request comparing privacy labels of a service provider with those of a user. The requested information is released based on the evaluation result.
- *Privacy labels are user-friendly.*
  We observed that people tend to use labels to represent abstract concept like the levels of seriousness or completeness. One example is that the United States government uses the similar concept to lower the national threat level from "Code Orange" to "Code Yellow." People may not fully understand the exact definitions of those labels. However, through the comparable hierarchical structure, people can understand the ideas behind the labels much easier.
- *Privacy labels work with a policy engine.*
  Instead of evaluating every element in privacy policies and user preferences, the policy engine needs to compare privacy label for each attribute in privacy policies and user preferences. It greatly reduces the cost for processing all requests as well as for invoking the interaction service.

In order to address the above-mentioned requirements, we need two policy schemas, PrivacyLabel and PrivacyHierarchy for the content of the privacy label and the hierarchical structure, respectively. The privacy label includes the following elements and such elements of privacy label are illustrated in Figures 3(a).

- Name: represents the privacy label.

[2]Liberty Alliance suggests only five privacy levels: Strict, Cautious, Moderate, Flexible and Casual [18].

- Purpose: describes the intention of data collection or usage of data.
- Access: indicates whether the SP provides access to the collected data.
- Recipient: describes all intended recipients of the collected data.
- Retention: indicates the retention policy applied to the data.
- Remedies: specifies the possible remedies for the policy mismatch.
- Disputes: indicates dispute resolution procedures.

B. *Managing Privacy Preferences*

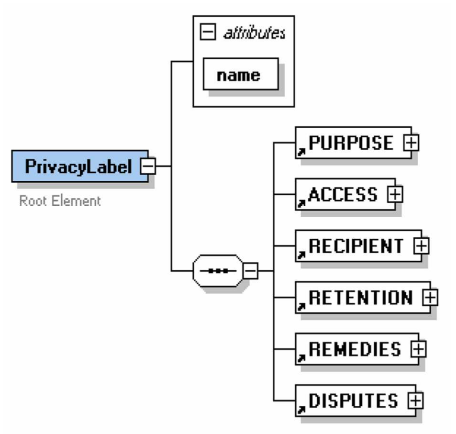
Our investigation indicated that the current FIM practices lack a well-defined and standardized structure for privacy policies to support identified practical scenarios addressed in section III. In addition, there is no systematic protocol for obtaining and storing users' preferences. Another important component to match the privacy policies with users' consent has not been fully discussed in the literature. In order to provide user-centric management, each user should be able to specify their privacy preferences, even though the ID-WSF [17] architecture provides a protocol for transferring the privacy related information in the request and responses[3]. In this section, we propose a preference specification language called PREP, which stands for PReference Expression for Privacy, to allow users to have control of the release and usage of their information stored at the attribute providers.

We now introduce an example to elaborate our approach and to highlight the need for PREP:

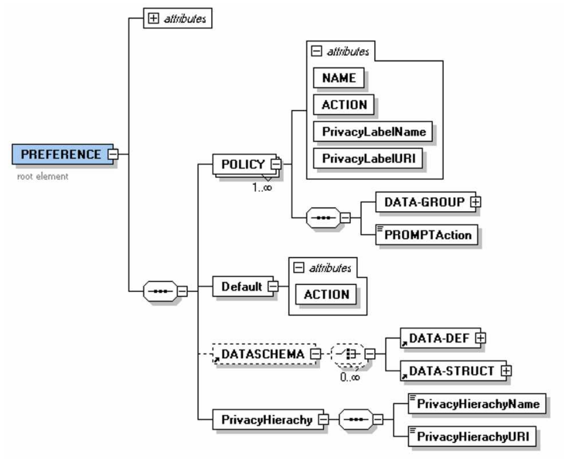
Consider that a user Cathy has requested a transaction at one of the SPs in a CoT. We assume that Cathy has been authenticated by the WSC at this point. The WSC may require some information regarding Cathy in order to complete the transaction. As a result the WSC makes an attribute request to Cathy's designated WSP which for simplicity in our case would be the IDP. For preserving privacy of the user information, which is the main goal of Liberty Alliance specifications, the IDP should release the requested attribute information with a proper user consent. IDP has already stored the user's preference regarding the release of information based on a multi-level policy approach, meaning that the user has categorized her personal information to be released with different levels of strictness. These strictness levels are directly pointed to the levels of standardized policies defined in the CoT. In such a case, WSPs just need to compare the privacy policy level in the request with the level in the preference and release the information to attribute requester(s) accordingly. In case of a mismatch, WSP can take appropriate actions preferred by the user and already stored in some form at the WSP.

It is obvious that we need to allow SPs and principals to precisely specify the different aspects of their privacy policies,

[3]More detailed explanation can be found in [15], [17].



(a) PrivacyLabel Schema



(b) PREP Schema

Fig. 3. Privacy Label and PREP

```
<?xml version="1.0" encoding="UTF-8"?>
<PREFERENCE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="
    http://www.megabank.com/Privacy/PREPv1.xsd" ExpirationDate="2007-08-13" NAME="ID000001">
 <POLICY ACTION="ALLOW" PrivacyLabelURI="http://www.megabank.com/Privacy/Cautious.xml" NAME="Cautious" PrivacyLabelName="Cautious">
 <DATA-GROUP base="http://www.megabank.com/Privacy/base.xml">
  <DATA ref="#user.name.given" />
  <DATA ref="#user.name.family" />
  <DATA ref="#user.home-info.telecom.telephone.loccode"/>
  <DATA ref="#user.home-info.telecom.telephone.number"/>
  <DATA ref="#user.home-info.telecom.moblie.loccode"/>
  <DATA ref="#user.home-info.telecom.moblie.number"/>
  <DATA ref="#user.home-info.online.email"/>
  <DATA ref="#user.home-info.postal.street"/>
  <DATA ref="#user.home-info.postal.city"/>
  <DATA ref="#user.home-info.postal.stateprov"/>
  <DATA ref="#user.home-info.postal.postalcode"/>
  <DATA ref="#user.home-info.postal.country"/>
  </DATA-GROUP>
  <PROMPTAction>Mismatch</PROMPTAction>
 </POLICY>
 <Default ACTION="Deny"/>
 <PrivacyHierachy>
  <PrivacyHierachyName>CoT1PrivacyHierachy</PrivacyHierachyName>
  <PrivacyHierachyURI>http://www.megabank.com/Privacy/PrivacyHierachy.xml</PrivacyHierachyURI>
</PrivacyHierachy>
</PREFERENCE>
```

Fig. 4. PREP Example

respectively. The various approaches can be considered to support the above scenarios. We may consider P3P [9] as a privacy framework for our scenarios. The major drawback for adopting the P3P based approach is the complexity in determining an intersection of the attribute requestor's privacy policy and the user's privacy preference policy in an automated fashion.

Using a P3P based approach would require a language like APPEL [8] for the WSPs to collect and store the user preferences. APPEL is a privacy preference expression language for P3P but it is very hard to understand and needs a special engine for a browser agent. According to P3P specifications, a single policy can have multiple statements covering different purposes for data collection. In an environment like the one we mentioned in our example, it would be a tough job for WSPs to evaluate all the permutations and combinations between the WSCs policies and the user's set of preferences in APPEL.

There are other related approaches. EPAL [20] is a privacy authorization language that can support authorization and is more stringent for Liberty Alliance requirements. Also, the enhanced SAML [13] can be considered as a way to support user friendly privacy/preference expressions.

Considering all the issues we discussed above, there is a clear need for languages to specify standardized privacy policies and to store the user preferences for corresponding such policies. The multi-level policy approach in Liberty Alliance specifications addresses the purpose of defining a set of standardizes policies for the CoT that both the users and WSCs may refer to. However, it does not propose any specification or rules for storing user preferences in a way that would facilitate the WSPs in matching the privacy policy levels in the attribute request with the levels in the user preferences.

Our work partially adopts P3P to contain various elements that define the web sites privacy policies regarding the purpose of information gathering, release procedures of information and various other factors such as access control to the col-

Policy Match (True or False)	Prompt Action: Always	Prompt Action: Mismatch	Prompt Action: Never	Return Value
True	True	False	False	1100 (Policy match, Prompt user)
True	False	True	False	1010 (Policy match, Do not prompt)
True	False	False	True	1001 (Policy match, Never prompt)
False	True	False	False	0100 (Policy mismatch, Prompt user)
False	False	True	False	0010 (Policy mismatch, Prompt user)
False	False	False	True	0001 (Policy mismatch, Never prompt)
True	True	True	True	1111 (No Operation)
False	False	False	False	0000 (Missing Attribute)
-	-	-	-	Other cases (Invalid Preference)

TABLE I
DECISION MATRIX

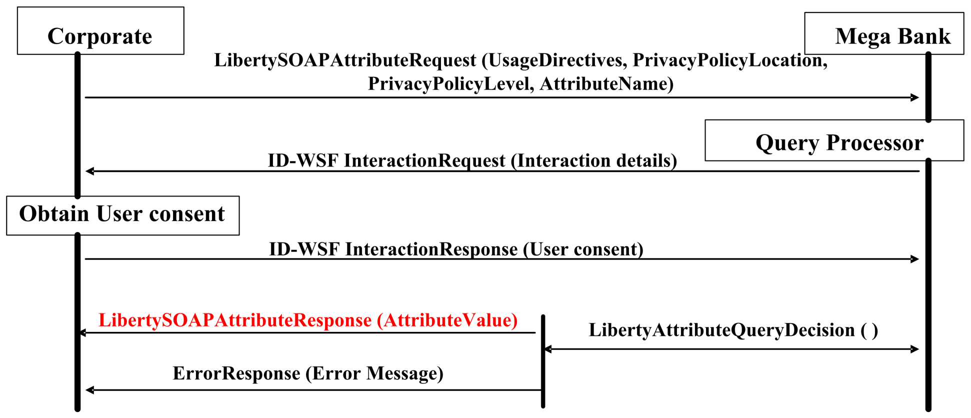

Corporate
Mega Bank
LibertySOAPAttributeRequest (UsageDirectives, PrivacyPolicyLocation, PrivacyPolicyLevel, AttributeName)
Query Processor
ID-WSF InteractionRequest (Interaction details)
Obtain User consent
ID-WSF InteractionResponse (User consent)
LibertySOAPAttributeResponse (AttributeValue)
LibertyAttributeQueryDecision ( )
ErrorResponse (Error Message)

Fig. 5. Message Flows

lected data, dispute resolution methods, and so on. We name our privacy policy framework as P3P*Lite*. For the privacy preference, we use the preference language called PREP that is much similar to APPEL. PREP shares the same extension formats of APPEL but is more restrictive than APPEL. The important drawback of APPEL in our work is that it cannot be customized to fit our scenarios because it does not support a multi-level policy approach suggested in Liberty Alliance specifications.

C. *Elements, Operations and Semantics in PREP*

PREP is a language used by the attribute provider to collect and store the user's privacy preferences. It further facilitates the decision process in legitimately releasing attributes at the attribute provider by comparing the policy level in the request with the level in PREP at the attribute provider's site. Also, PREP supports multi-level privacy policy approach in Liberty Alliance specifications. The set of standardized privacy policies should be formalized by a mutual agreement between all the entities of the CoT. There are a couple of assumptions that PREP inherits directly from Liberty Alliance specifications [18]:

- The WSP has previously collected a principal's consent, access and privacy preferences/policies for the attributes in question.
- The CoT has a web site of its own, or uses an external "Policy Broker" web site, where the privacy policies are available online.
- The SP/WSC sets the Privacy Policy and the principal specifies the Usage Policy (such as preferences).
- The consistent naming is used to indicate *who* decides *which* policy is applied to *what* attributes. In other words, the following hypothetical relationship should be supported: PrivacyPolicy$_X$ = UsagePolicy$_X$.

The WSP collects the user's privacy preference at the time of sign-up. Irrespective of the methods used for collecting the user's preferences, the preference should be stored in the format specified in the PREP structure. Just like other standardized protocols proposed by Liberty Alliance, all entities in the CoT should be mandated to follow the PREP structure for managing privacy preferences, as each entity acts as an attribute provider.

The PREP contains a set of elements that help the attribute provider store privacy preferences provided by the user into a standardized machine readable XML format. The conversion of the user preferences from a high level to XML is done by the PREP generator. The PREP generator is a program that takes the input from the user and converts it into an XML file satisfying the PREP structure for the user preferences. The PREP elements and an example are illustrated in Figures 3(b) and 4.

D. Parsing PREP: PREParser

Based on the proposed structure, we also developed a mechanism, named PREParser, to process user preferences specified in PREP. PREParser is an XML based rule engine for PREP. It evaluates the user preferences upon receiving an attribute request message. We utilize the decision matrix to expedite the evaluation process. The matrix includes all possible policy levels and preference types. PREParser firstly checks whether the incoming policy level matches with the user defined privacy preferences using this matrix then returns a corresponding decision. The matrix scales down the number of expected outputs from the parser as shown in Table I. The returned decision value eventually triggers the interaction service based on the specified prompt action. PREParser processes the PREP rule set according to the following guidelines:

- A PREP rule set should start with a <leppl: Preference> tag and should contain the `xmlns` extension that specifies the namespace for the XML Schema for PREP. The absence of <leppl:Preference> tag invalidates the rule set and any further processing should be aborted.
- Every PREP rule set can have only one <leppl: Preference> element but can have multiple <leppl: Policy> elements.
- Every <leppl: Policy> element should include mandatory extensions, containing the name of the policy in the CoT. The policy contained in `ref` follows the same nomenclature or is similar to the one that is in the attribute request.
- There can be only one <leppl: DataGroup> element in a single <leppl: Policy> element. A <leppl: DataGroup> should have at least one <pp: Data> element depending upon the type of data contained in it.

In order to validate the feasibility of the proposed P3P*Lite* and PREP languages, we have implemented prototypes for the scenarios identified in Section III. We developed the meta-information of the APIs for the basic functions derived from our approach [4]. Meta-information consists of the class name, the class description and the signature of the corresponding methods under each class. The method signature includes method name, return type, the number and order of the parameters, and the types of the parameters. The purpose of this approach is to provide a blueprint to the developers so that they can follow, modify or enhance each class based on their own needs. This approach is also platform- and program language independent. Figure 5 illustrates the message flows in our prototype of Figure 2. We also developed several classes to support our approach and the PREParser as follows:

- *UserPreferences Class* This class represents the PREP user's preferences XML document object. When an instance of UserPreferences is created, this class parses the user's preferences automatically.
- *PolicyUtils Class* This class is a collection of methods to handle privacy policy. Obviously, the service provider has more chance to use this class because the methods provided are mainly the preparation of the request SOAP message to and the handle of response SOAP message from the attribute provider.
- *PreferenceUtils Class* This class is a collection of methods to handle user's preference. The attribute provider has more chance to use this class since the methods provided are mainly the preparation of the response SOAP message to and the handle of request SOAP message from the service provider.
- *PolicyEngineUtils Class* This class is a collection of methods to handle intersection of the privacy policy and user's preferences. The attribute provider uses this class for determining whether the information is released or not.

[4]In Appendix, we include the meta-information of such APIs.

IV. Conclusion and Future Works

Information security and privacy issues are the key concerns in FIM because identity federation requires the exchange of sensitive user information in a highly insecure and open network. In this paper, we have focused on Liberty Alliance approach along with privacy issues in FIM through possible business scenarios. In addition, we have proposed a user preference expression language and a simplified privacy policy language that are crucial to manage users' PII in FIM. We believe our work can be leveraged by the research and industry communities working on privacy issues in identity management.

Our future work will focus on an enhanced privacy attribute management framework which can provide users with a high level of confidence in protecting and controlling their personal data. Developing appropriate information assurance (IA) metrics for user-centric identity management is another issue that we intend to work on in the near future. It is generally believed that no single perfect set of IA metrics can be applied to all systems [4]. Thus, we would attempt to investigate IA metrics specifically designed for user-centric identity systems. In addition, we would apply the proposed framework to our on-going project on Microsoft's CardSpace [6], evaluating and validating the IA metrics.

Acknowledgements

This work was partially supported at the Laboratory of Information of Integration, Security and Privacy at the University of North Carolina at Charlotte by the grants from e-Business Technology Institute, National Science Foundation (NSF-IIS-0242393), and Department of Energy Early Career Principal Investigator Award (DE-FG02-03ER25565). We are grateful to Todd Inskeep, Sam Phillips, and Larry Hollowood for their support and encouragement in making this work possible. The opinions expressed in this paper are of course our own and should not be taken to represent the views of these individuals.

References

[1] W3C Note: Simple object access protocol v 1.1. Technical report, Available at www.w3.org, 2000.

[2] W3C note: Web services description language (WSDL) v 1.1. Technical report, Available at www.w3.org/, 2001.

[3] G.-J. Ahn, D. Shin, and S.-P. Hong. Information assurance in federated identity management: Experimentations and issues. In *Proceedings of 5th Web Information Systems Engineering Conference, Lecture Notes in Computer Science (LNCS3306)*, pages 79–90, Brisbane, Australia, November 2004.

[4] A. Bhargav-Spantzel, J. Camenisch , T. Gross and D.Sommer. User Centricity: A Taxonomy and Open Issues. In *Proc. of ACM workshop on Digital Identity Management*, November 2006.

[5] K. Cameron. The Laws Of Identity. Microsoft Corporation, White Paper, May 2005.

[6] K. Cameron. Microsoft Identity Metasystem. Available at www.identityblog.com, 2006.

[7] D. Chaum. Security without identification: Card computers to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[8] L. Cranor, M. Langheinrich, and M. Marchiori. A P3P preference exchange language 1.0 (APPEL1.0). Technical report, Available at www.w3.org, 2002.

[9] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification. Technical report, Available at www.w3.org, 2002.

[10] L. F. Cranor. Agents of choice: Tools that facilitate notice and choice about web site data practices. *Proceedings of the 21st International Conference on Privacy and Personal Data Protection*, September, 1999, pp 19-25, Hong Kong SAR, China.

[11] H. Damker, U. Pordesch, and M. Reichenbach. Personal reach ability and security management - negotiation of multilateral security. In *Proceedings of Multilateral Security in Communications*, Stuttgart, Germany, 1999.

[12] Federal Trade Commission. Online Profiling - A Report to Congress, part 2. Technical report, 2002.

[13] P. Hallam-Baker and E. Maler. Assertions and protocols for OASIS SAML. Technical report, Available at www.oasis-open.org, 2002.

[14] J. Hegel and M. Singer, editors. *Net Worth: Shaping Market When Customers Make the Rule*. Harvard Business School Press, 1999.

[15] J. Hodges and T. Watson. Liberty architecture overview v 1.2-03. Technical report, Available at www.sourceid.org, 2003.

[16] IBM. Web services security (WSS) specifications 1.0.05. Technical report, Available at www-106.ibm.com, 2002.

[17] Liberty Alliance. ID-WSF security and privacy best practices. Technical report, Available at www.projectliberty.org.

[18] Liberty Alliance. Privacy preference expression languages. White report, Available at www.projectliberty.org.

[19] Mircrosoft Corporations. Microsoft .Net Passport Review Guide. Technical report, Available at www.microsoft.com, 2003.

[20] M. C. Mont, S. Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. Technical report, Available at www.hpl.hp.com, 2003.

[21] P. Shenoy, D. Shin, and G.-J. Ahn. Towards IA-Aware web services for federated identity management. In *Proceedings of IASTED International Conference on Communication, Network, and Information Security*, pages 10–15, New York, USA, December 2003.

[22] D. Shin, G.-J. Ahn, and P. Shenoy. Ensuring information assurance in federated identity management. In *Proc. of the 23rd IEEE International Performance Computing and Communications Conference (IPCCC)*, Phoenix, Arizona, April 2004.

[23] T. Watson. Liberty ID-FF implementation guidlines v 1.2.02. Technical report, Liberty Alliance Project, 2003.

Appendix: Classes

UserPreferences Class

This class represents the PREP user's preferences XML document object. When an instance of UserPreferences is created, this class parses the user's preferences automatically. The typical initiation is as follows:

```
UserPreferences userpreferences
   = new UserPreferences ("c:/temp_prep")
```

Constuctor

There is a constructor in this class.

```
UserPreferences(String PreferencesFile){
           parseUserPreferences(PreferencesFile);}
```

Methods

```
String getDefaultAction()
//return the default action of the user's preferences

String getPrivacyHierachyURI()
//return the URI of the Privacy Hierarchy definition
//of the user's preferences

ArrayList getAttributes()
//return an array list of the attributes inside
//the user's preferences

ArrayList getAttributesAll()
//return an array list of five objects related to
//attributes
//1st object is an array list of the attributes
//inside the user's preferences
//2nd object is an array list of the URIs of the
//Privacy labels in 1st object
//3rd object is an array list of the URIs of the
//base data schema in 1st object
//4th object is an array list of the action of the
//each attribute in 1st object
//5th object is an array list of the prompt action
//in 1st object

String getExpirationDate()
//return the expiration date of the user's
//preferences
```

PolicyUtils Class

This class is a collection of methods to handle privacy policy. Obliviously, the service provider has more chance to use this class because the methods provided are mainly the preparation of the request SOAP message to and the handle of response SOAP message from the attribute provider.

Constuctor

No constructor in this class.

Methods

```
ArrayList getAllRequestedAttributes(
                          String PrivacyPolicyURI)
//return an array list of attributes from
//the privacy policy with given URI

ArrayList GetDataGroupHierachy(
                       String BaseURI, String ref)
//return an array list of all attributes under
//the attribute group ref with reference to the
//base data schema with given URI

String getAttributeProvider(
              String attribute, String pseudonym)
//return the AP's endpoint for a specific user
//with the given pseudonym and a specific attribute

SOAPMessage CreateSoapMessage(
             String pseunym,
             ArrayList all_attributes,
             String PrivacyPolicyURI,
             String messageID)
//return a soap message which is going to send to
//the AP. The inputs are: a specific user with the
```

```
//given pseudonym, an array list of all_attributes
//to that AP, the privacy policy with given URI,
//and the message ID

String genMessageID()
//return a random number for the message ID

String getRedirectRequestURL(SOAPMessage msg)
//return the redirect URL from the soap message msg

String getMessageID(SOAPMessage msg)
//return the message ID from the soap message msg

SOAPMessage sendSoapMessageAndgetResponse(
                            SOAPMessage attrQuery,
                            String endPointURI)
//return a response soap message from the AP with
//the endpoint endpoint URI by sending a request
//soap message attrQuery to that AP

ArrayList parseAttributeQueryResponse(
                             SOAPMessage msg,
                             ArrayList tmp_attrs)
//return an array list of the values of the
//attributes tmp_attrs from the response soap
//message

String getAlternativeAttributeName(
                                String attribute)
//return the attribute in the implementation.
//For example, user.name.given in base data schema
//becomes first_name in a database or a HTML
//field name
```

PREFERENCEUTILS CLASS

This class is a collection of methods to handle user's preference. The attribute provider has more chance to use this class since the methods provided are mainly the preparation of the response SOAP message to and the handle of request SOAP message from the service provider.

Constuctor

No constructor in this class.

Methods

```
String getUserPreferences(String pseudonym)
//return the user's preferences for a specific
//user with the given pseudonym

String getStorageAttributeName(String attribute)
//return the name of the attribute in a storage
//e.g. database, file, LDAP, etc

String getAttributeValue(String pseudonym,
                     String storageAttributeName)
//return the value of an attribute for a specific
//user with the given pseudonym in the storage

SOAPMessage CreateSoapMessage(
                     ArrayList attributes,
                     ArrayList attributeResults,
                     String CorrelationMessageID)
//return a soap response message which is going
//to send to the SP. The inputs are: an array
//list of requested attributes, the values of
//those requested attributes, and the
//correlated message ID

SOAPMessage sendSoapMessageAndgetResponse(
                            SOAPMessage response,
                            String endPointURI)
//return a response soap message from the SP
//with the endpoint endpoint URI by sending
//a soap message "response" to that SP

String getStatusCode(SOAPMessage SPResponse)
//return the status code form the response soap
//message form SP
```

POLICYENGINEUTILS CLASS

This class is a collection of methods to handle intersection of the privacy policy and user's preferences. The attribute provider uses this class for determining whether the information is released or not.

Constuctor

No constructor in this class.

Methods

```
boolean isPrivacyPolicyExpired(
                   String PrivacyPolicyURI)
//return true if the Privacy Policy is already
//expired.
//Return False, otherwise.

boolean isUserPreferencesExpired(
                     String PreferencesFile)
//return true if the user's preferences is already
//expired.
//Return False, otherwise.

boolean comparePrivacyLabels(
                      String userPrivacyLabel,
                      String spPrivacyLabel,
                      String PrivacyLabelURI)
//return true if spPrivacyLabel is equal or
//stricter than userPrivacyLabel.
//Return False, otherwise.

boolean comparePrivacyHierachyURIs(
                      String spPrivacyHierachyURI,
                      String userPrivacyHierachyURI)
//return true if spPrivacyHierachyURI and
//userPrivacyHierachyURI are the same to see
//whether they are from the same Privacy
//Labeling System.
//Return False, otherwise.

boolean compareBaseURIs (String spBaseURI,
                         String userBaseURI)
//return true if spBase and userBase are the same
//to see whether they are from the same Privacy
//Labelling System.
//Return False, otherwise.

int getHierachyPostion (String PrivacyLabel,
                        String PrivacyHierachyURI)
//return an integer to represent the position of
//a PrivacyLabel in a hierarchy defined in the
//given URI (PrivacyHierachyURI)
```