On Supporting Active User Feedback in P3P

V.N. Venkatakrishnan
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: venkat@cs.uic.edu

Wei Xu
Department of Computer Science
Stony Brook University
Stony Brook, NY 11790-4400
Email: weixu@cs.sunysb.edu

Rishi Kant Sharda
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
Email: rsharda@cs.uic.edu

***Abstract*— We propose an extension to the P3P framework that enables a consumer and a web service to engage in active policy negotiation. In addition, we discuss enforcement strategies for negotiated consumer preferences.**

Keywords: *P3P, Web Service, Information Privacy, Policy Negotiation, Policy Enforcement, User Feedback.*

I. Introduction

The development of the world wide web as a platform for electronic commerce, auctioning and social networking has presented several challenges to end user privacy. Users are presented with situations that require them to disclose (either implicitly or explicitly) personal information to these web based services. To increase consumer confidence, web sites have made efforts to clearly display their policies regarding use of private information. Unfortunately, such policy descriptions are verbose and ridden with legal jargon, rendering them ineffective in increasing consumer confidence in the privacy of their personal information.

To address privacy concerns in web services, the Platform for Privacy Preferences framework (P3P [1]) has been proposed. P3P enables websites to express their terms of use regarding privacy in a machine-readable format using a standardized vocabulary. Ever since the standard was officially published by the W3C (the WWW consortium) in 2002, several leading websites have become P3P compliant (a list can be found in [2]), and P3P is expected to be more widely deployed in future. In addition, P3P has been adopted as the de-facto standard for web-services privacy (WS-privacy) [3].

The original P3P approach (as defined by the P3P 1.1 draft) follows a *client-centric* mechanism for checking a web site's privacy policy against a user's privacy preferences. A P3P user agent that runs on the client (usually, a browser extension) retrieves the P3P privacy policy from a web service, and checks it against the consumer's privacy preferences. Agrawal et al. [4] designed a *server-centric* P3P architecture mainly for better performance and scenarios involving thin clients. In this architecture, the consumer's entire privacy preferences are sent to the server and are matched with the web site's P3P policy at the server side.

Both the client and server based architectures can be examined from three key problem perspectives:

- *Policy compatibility checking*. This is the problem of checking whether the client's preferences match the web service's policy. For policy compatibility checking, a client-centric architecture is far superior compared to a server-centric one as the compatibility process runs entirely in the client end under the purview of the consumer. By contrast, in a server-centric approach, the consumer needs to transmit her entire preferences to the server and place much trust on the server to correctly match her preferences. Such asymmetric placement of trust violates the preferred security design principle of psychological acceptability [5].
- *Policy negotiation*. Policy negotiation concerns the issue of negotiating a new policy for the consumer in case of a policy mismatch. Using a purely client-centric architecture (such as the P3P original framework) it is impossible to provide any support for policy negotiation. This is because policy negotiation needs to engage the server for supporting any possible changes to the policy after a policy mismatch. A purely server-centric architecture can deal with policy negotiation by trying to enforce the consumer's policy, which is discussed below.
- *Policy enforcement*. Policy enforcement concerns the problem of enforcing any policy that is agreed between the consumer and the web service. Policy enforcement is inherently a server-based operation and hence requires the participation of the server.

From the above discussion, it is clear that for reasons of policy enforcement, participation from the server is needed, and a client-side policy compatibility checking is more suitable for psychological acceptability. Policy negotiation requires participation from both client and the server. This naturally leads us to an alternative architecture where the client and the server share the following responsibilities: The client performs policy compatibility checking, and the server performs policy enforcement. Both the client and server are engaged in policy negotiation. In this paper, we discuss the architecture of this framework and examine the key design and implementation issues in realizing this architecture.

Paper organization. The rest of the paper is structured as follows: We first review the related work in Section II. In Section III we use an example service to further motivate the need for policy negotiation in P3P. Then we describe our extensions to P3P for policy negotiation and enforcement in Section IV. We conclude the paper in Section V.

II. Related Work

A complete description about P3P and APPEL can be found at [1] and [6]. Yu et al provided a formal semantics for P3P in [7]. Several P3P and APPEL tools have been implemented. The most notable P3P client implementations include AT&T Privacy Bird [8] and the implementation of compact P3P policies in the Microsoft Internet Explorer for cookie handling.

Many researchers have pointed out the limitations of P3P and APPEL. Hogben ([9] [10]) noted the limitations of P3P in areas such as vocabularies as well as the ambiguity of APPEL. Agrawal et al [11] showed the limitations of APPEL in terms of clarity and expressiveness with a set of examples, and then proposed XPref, an XPath-based P3P privacy preference language, to address these problems. Kolari et al [12] proposed a different enhanced P3P privacy preference language called Rei. They also noted the limitations of the trust model of P3P and proposed an extensible trust model based on social recommendations. Although the P3P extension presented in this paper is based on APPEL, the similar technique can be applied to other privacy preference languages as well. Meanwhile, the proposed P3P extension can be complimentary to other suggested P3P enhancements.

Automated trust negotiation[13] and privacy negotiation [14] are well known concepts that have been studied in broader settings. In this paper, we have provided a framework for realizing such negotiation within the context of P3P, and provide details and transition steps to incorporate the privacy policy enforcement within the context of the P3P negotiation framework.

We proposed a framework for building privacy-conscious web services in our previous work [15]. The framework allows negotiation and enforcement of privacy policies, but uses its own protocol and is not based on P3P. In this paper, we have borrowed some of the ideas from that framework, and have applied them to P3P.

III. A Motivating Example

As a running example to facilitate our discussion, consider a web service `www.abcshop.com` that provides shopping services over the web. To improve user experience, the service uses an external "clickstream" service `www.hintsforclicks.com` that aims to dynamically provide shopping suggestions. It does so by collecting the user's browsing pattern and displaying suitable suggestions in the same browser window, but in a separate browser frame. To accomplish this, whenever the user clicks on a link, information about this link is sent to the clickstream service, which compares the current browsing pattern with its (internal) database. This comparison generates suggestions on third-party websites that are closely related to the current browsing context such as sponsored links with similar products. These external links are then inserted into a dynamically generated web page, which is rendered in the current browser window. Similar suggestions routinely appear during shopping visits on e-commerce sites such as `buy.com` and `amazon.com`.

Since consumer information is now provided to the clickstream service, the site privacy policy describes this practice. Using P3P, the policy can be described using the data schemas provided in the P3P vocabulary.

```
<STATEMENT>
  <PURPOSE required="opt-out">
    <individual-analysis/>
  </PURPOSE>
  <RECIPIENT><other-recipient/></RECIPIENT>
  <RETENTION><stated-purpose/></RETENTION>
 <DATA-GROUP>
    <DATA ref="#dynamic.clickstream"/>
    <DATA ref="#dynamic.http"/>
  </DATA-GROUP>
</STATEMENT>
```

The (relevant portions of) P3P policy for `abcshop.com` is shown above. It states that the web service collects `clickstream` data and other http data when a user browses their site. This is done for individual analysis (stated through the `PURPOSE` tag) by some other third party clickstream service, (through the `RECIPIENT` tag) which is has a different policy than `abcshop.com`, suggesting that clickstream information is disseminated to the third party. The visitor has the option to `opt-out` of this data collection.

We now describe the P3P compliance checking process. When the consumer receives the web site P3P policy, she checks if these match her privacy preferences, which is done using a P3P user agent (usually a browser extension). In our example, the consumer's privacy preferences on the use of clickstream information is matched with `abcshop.com`'s P3P policy. Consumer preferences are expressed in APPEL [6]. In APPEL, the preferences are specified in a set of preference-rules (called a *rule-set*) which can help a user agent (UA) to make automated or semi-automated decisions regarding the acceptability of a P3P policy.

In our example, the users considers her clickstream data as private. as given by the APPEL ruleset given below.

```
<appel:RULE behavior="block" prompt="yes"
    promptmsg="obligation:other-clickstream">
  <p3p:POLICY>
   <p3p:STATEMENT>
    <p3p:PURPOSE>
      <p3p:individual-analysis/>
    </p3p:PURPOSE>
    <p3p:DATA-GROUP>
      <p3p:DATA ref="#dynamic.clickstream"/>
    </p3p:DATA-GROUP>
    <p3p:RECIPIENT>
      <p3p:other-recipient/>
    </p3p:RECIPIENT>
   </p3p:STATEMENT>
  </p3p:POLICY>
</appel:RULE>
```

The rule set specifies the behavior of the user agent with respect to P3P policy. When a rule is matched (evaluated TRUE), there are three possible outcomes, specified in the same rule:

- `request` : the provided policy is acceptable.
- `limited` : the provided policy is somewhat acceptable.
- `block` : the provided policy is not acceptable.

A server-centric architecture of P3P requires the rule matching to be performed at the server. Such a preference matching process at the server end may not be acceptable to a conscious user. The current client-centric architecture of P3P too does not allow for negotiating a new policy when the original policy conflicts with the user preferences. We note that such

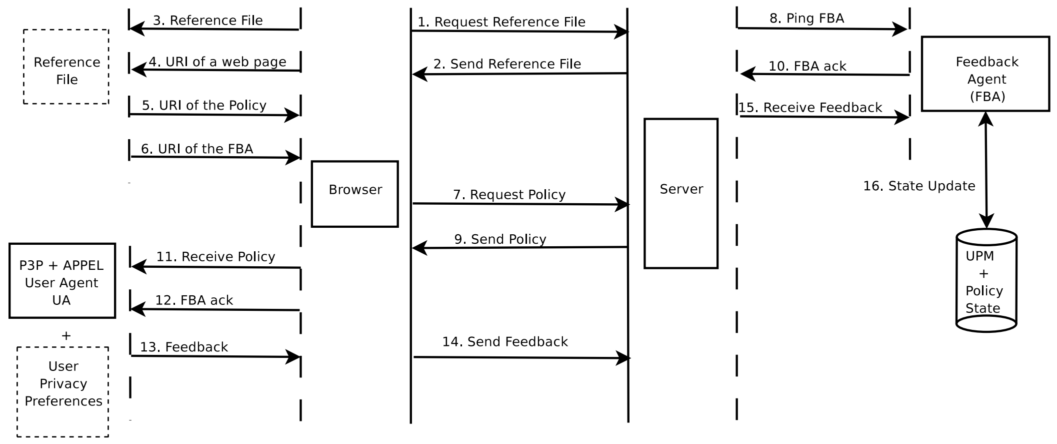

Fig. 1. Privacy negotiation framework

conflicts are bound to occur in practice. For instance, web sites provide a rich set of features to achieve greater levels of quality of service. This is typically achieved by interacting with several other entities to which consumer information dissemination may be required. When consumers use such web sites, conflicts between the user's privacy preferences and the web site P3P policy are likely to arise (in terms of disclosure of information to other parties). When the likelihood of such conflicts is frequent, web services will need to provide the user with the functionalities that respect the user's privacy preferences, possibly at a marginal cost of usability or functionality for the user. In our example, the web service can render its functionality without revealing the clickstreams of the user, at the marginal cost of functionality (i.e., absence of useful external link suggestions).

To support such privacy preferences, feedback from the conflict resolution process needs to be provided to the web service. This will allow the client to *negotiate* a new policy with the server, if it is possible. In the previous example, for instance, the user can let the web service know that her privacy preference is to have "no clickstreams collected and sent to an unrelated third-party." The web service can then use this feedback to provide a variation of its service that does not render the browser frame that sends clickstreams to the external third-party service.

In the case of the above example, P3P does not provide support for a privacy conscious user who does not want to reveal her browsing history to an external clickstream service.

Such feedback is currently accomplished outside of P3P, by accommodating it as part of the website functionality. This is done through opt-out text boxes and radio buttons in which the user fills with her privacy preferences. When such functionality is presented outside the P3P framework, we lose all the advantages of having P3P as a standardized and automated policy compliance mechanism for checking privacy preferences. Without the automated P3P process, the user needs to expend additional manual effort to ensure that her privacy preferences are met, by analyzing the data provided to the web site, searching for these options in the service web pages that vary from one service to another, and opting out of them. This manual process is also likely to be error-prone, and therefore weakens P3P as an effective standardization effort.

We present a set of extensions to the P3P framework to support *active user feedback* mechanisms that facilitate policy negotiation and enforcement of the negotiated user preferences. In the following section, we describe the extensions to components of the P3P framework to achieve active user feedback mechanisms.

IV. Description of the Extensions

In this section, we outline a scheme for policy negotiation in the P3P framework. As suggested in the introduction, this will require changes to both the client and the server.

Figure 1 shows the interaction diagram that allows for privacy negotiation between the client and the web service. It shows the sequence of numbered messages exchanged and corresponding actions during the negotiation phase. The client (browser) is equipped with a User Agent (UA) which is capable of matching P3P policies with user preferences specified in APPEL. The input for this client is the User Preference File which contains user preferences in APPEL. We also introduce a new component on the server side, called the Feedback Agent (FBA). The feedback agent maintains the (per-user) policy on the server side. This policy (state) is arrived by a sequence of negotiation messages exchanged between the client and the web service. This policy state is further used in during the feedback enforcement step (not shown in the interaction diagram).

We now briefly explain the steps in the negotiation process shown in the above figure. Steps 1-5 follow the usual P3P 1.1 specification, and are used for obtaining a P3P policy file from a web service. In addition a ping message exchange (step 6,8 and 10) also occurs which tests the existence of the FBA. In the absence of such a FBA, the interaction sequence resorts to the original P3P specification. Once the presence of the FBA is confirmed, the actual steps of negotiation begin, starting with the client testing the policy against user preference and generating feedback. In steps 9-15 the feedback is processed by the FBA and changes to the policy state are made for enforcement.

Although the figure shows only one iteration in such a negotiation, one can envision several iterations in the procedure, as steps 9-15 can be repeated till the final negotiations, until a common ground is reached. This is especially feasible if the server can trade-off functionalities (that possibly adds a cost of usability or functionality) for better privacy. However, we do not discuss negotiations involving multiple iterations further in this paper due to lack of space.

The new framework requires a few changes to the existing P3P standard. Specifically, changes are required in the P3P policy specification language, the user agent functionality, creation and deployment of a feedback agent and finally, modifications to the web service code. We discuss them in the following section.

A. Extension to P3P policy specification

The location of the server-side feedback agent is not known to a user agent. Although the agent can be made available at a fixed relative URL from the policy, a better alternative is an extension to the P3P policy to specify the location of the server-side feedback agent. We first note that every P3P policy has an associated `POLICY_REF` element. Therefore, `POLICY-REF` tag can have an attribute that may be used to specify the feedback URI (channel). This is done by specifying an additional element `FEEDBACK`, by specifying the URI in it's “about” attribute. For example, a feedback channel specification may appear as `<FEEDBACK about="/P3P/Feedback/......">` If such an URI specification is absent, then the user agent believes that the service does not support the negotiation mechanism.

B. Client-side Changes

We point out that while certain information may strictly be required in a web service's policy, other pieces of information may be purely optional. In the shopping scenario described earlier, while it is required to request a user's credit card information and disclose this information to the credit card company for billing, it is purely an optional functionality to send the user's clickstream to the third-party web service for collecting browsing behavior. To specify such required and optional behavior, we point out that the current P3P specification language already has constructs available to specify such behavior. Specifically, the `required` attribute (which allows three values: `always`, `opt-in` and `opt-out`) for the `PURPOSE` and `RECIPIENT` tags. The use of such optional information can clearly be negotiable with the server.

When preference matching with APPEL specifications is done, certain APPEL rules are matched. As noted above, only the `request` rules are allowed. In case a `limited` or `block` rule is matched, the same rule is encoded as a feedback message for the web service. Such active user feedback messages are encoded in the same preference language. The procedure that performs the matching in the user agent needs to be augmented to generate a collection of rules that compose an active user feedback message. In an active user feedback message, the desired privacy constraints are given as a collection of rules that deny data collection and/or dissemination. For instance, in our running example, the APPEL specification shown earlier results in an active feedback message, as it blocks collection of any clickstream information, and therefore conveys that no `DATA` items clickstreams should be disclosed to any `RECIPIENT` that are do not share the same P3P policies.

To transmit the user feedback back to the server, the P3P user agent invokes the service-side feedback agent defined in the P3P policy. This is accessed using the URL that was described earlier in the policy extension. The user agent posts the user feedback as input to the server-side agent.

C. Server-side modifications

A service-side feedback agent accepts user feedback, participates in the negotiation and delivers the final feedback to be enforced to the associated web service instance. The existence of this agent is crucial for processing the feedback delivered by the user agent. The feedback is an APPEL ”block” or ”limited” rule which was matched by the user agent and is interpreted by the feedback agent.

We now discuss the construction of the feedback agent. First, the feedback agent needs to have access to the same session tracking mechanism of the server. Using this, it relates the user feedback to the correct web service instance. It then incorporates the results of the feedback into the policy to produce a `User Preference Matrix` (UPM) that is kept for each user by the server. The UPM is an access control matrix that consists of (`recipient, data-item`) pairs. Each entry in the UPM is a boolean value that suggests whether the specified recipient can receive the corresponding data item. An example UPM appears in Figure 2.

To create such a UPM, first the P3P policy is partitioned into *essential* and *optional* fields. This can be done easily by scanning the policy for the optional fields that have an `opt-out` or `opt-in` value. Once such a partitioning is done, the UPM is created by the FBA initially based on the P3P policy and populated with flags which are necessary for essential services. The construction of the UPM follows the following steps:

1) *Creation.* For every piece of information to be collected from the user for any purpose a column is added to the UPM. For every different recipient (including `ours`) add a row to the UPM. The matrix is initialized to all zeros.
2) *Initialization.* Initially, for all the essential and optional data-services combinations the entry value of 1 is made.
3) *Update.* Upon receiving a feedback message from the UA, updates are made to the entries that correspond only to the *optional* items from the feedback message. These are flipped from 1 or 0.

So far, we have discussed the procedure for negotiating a policy between a user agent and a feedback agent. The entire negotiation procedure can be summarized as follows:

- The server sends (on clients request) a P3P policy, now specifying a feedback channel, for the User Agent(UA) to

	clickstream	http	contact-info
ours	0	1	1
other-recipient	0	0	0

Fig. 2. An example of User Preference Matrix, which states that the user does not want her privacy information, such as clickstream, http browsing data, and contact information, to be collected and sent to external services.

communicate with the Feedback Agent(FBA). The FBA has the capability to distinguish between various sessions between different user agents.

- The user agent evaluates the P3P policy against the user preferences. If a rule is matched, and is a `limited` or `block` rule, it becomes a feedback message for the web service and is sent to the FBA.
- After evaluating the feedback message from the UA, the FBA makes updates to a User Preference Matrix (UPM) and the P3P policy.
- During this update, the FBA ensures the minimum criteria for essential services is not violated.

D. P3P-Aware Service Code

Once the P3P policy is negotiated between the User Agent and the Feedback Agent, we need to make sure that the web service does exactly as stated in the just negotiated P3P policy. We note that enforcing an entire P3P policy in a web service is a difficult task in general as P3P policy specification is not suitable in its original form for direct enforcement. Here, we propose a *much restricted notion* of enforcement, that of only enforcing the user feedback preferences as part of a negotiated P3P policy. We claim that this form of enforcement is feasible, as web service providers already provide technical means for consumers to specify their privacy requirements. Our proposal only integrates such means into the P3P framework.

Similar to the P3P policy, each service code can also be partitioned into two parts: essential and optional services. The feedback preferences in a negotiated policy describe the user's choices on the optional services. The notion of enforcing such user preferences corresponds to respecting the user's choices only on these optional services. This is done by blocking the (optional) operations prohibited by the APPEL messages that were provided as feedback, while providing the essential services and any desired optional services. To achieve this, provisions need to be built inside the service code. These "hooks" place each optional operation inside a boolean condition that is required to be true for executing that operation. In fact, such checking code is common in existing services that support opt-in or opt-out services, although it is implemented outside of P3P. (The values used in the checking condition usually come from user inputs through the service user interface such as opt-out text boxes and radio buttons.)

In this section, we illustrate the creation of P3P-aware service code in which the checking code for enforcing a user's preferences can take the negotiated P3P policy as its input. An important idea in enforcement is our use of the User preference Matrix, or UPM, that was introduced in the earlier section. The UPM is used to pass the user's preferences into the service code and is thus used to control the execution of optional services. Next, we describe the necessary manual steps to modify existing service code to enable the use of UPM for enforcement of the user's privacy preferences.

Let us consider the shopping example service again. Below is the code snippet of a possible implementation of the service, in which a function named `getLogInfo` is defined to collect the user clickstream information and send it to a clickstream service.

```
// A function to collect user browsing data
// and notify a clickstream service
void getLogInfo() {
  log += getUserIP();
  log += getRequestTime();
  log += getHTTPStatus();
  clickstreamService.notify(log);
}
```

As the first step to enable a web service code with the P3P-aware features, information needs to be maintained about operations (third party API's or some internal functions) that perform external communication, and their associated variables that represent data items that are disclosed to service partners. An analysis of the web service structure is done and a mapping is created which defines how the data items referenced in the P3P policy and UPM can be mapped onto functions and variables in the service code. This analysis is one-time and can be performed manually or semi-automatically with the help of code scanning tools. An example mapping for `clickstream` is shown below:

```
<map>
  <recipient name="other-recipient">
    <data ref="clickstream">
      <function name="getLogInfo">
        <variable>log</variable>
      </function>
    </data>
  </recipient>
  ...
</map>
```

The mapping file is maintained in an XML format, and specifies functions and variables corresponding to each P3P recipient (e.g. `ours` or `other-recipient`) and each referenced P3P data item (e.g. `clickstream`). For instance, the example mapping states that the function `getLogInfo` and the variable `log` are used for the recipient `other-recipient` and data item `clickstream` in the P3P policy.

To actually enforce the negotiated policy, the web service code has to be modified so that every input/output (as well as internal read/write) operation of the service is encapsulated in a check that checks for secure information disclosure.

We can use the above mapping to help us to build the new P3P-aware service code from the original code. The mapping dictates how the relevant portions of the code need to be modified and how the UPM should be consulted for each modification. Specifically, we process each entry in the mapping file as follows. We first locate the corresponding lines in code, and then put in a condition check such that the following items hold:

- In case the flags in the UPM suggest that this operation (e.g. some third party operation) cannot be performed (as per user preferences) then code is not executed.

- In case the flags suggest that the operation can be performed but no user specific information can be sent then the variables pertaining to them are assigned junk/blank/null (whichever applies) values.

Only the code for optional services will be modified. The code for essential services is not modified.

As an example, the modified `getLogInfo` in the P3P-aware service code of the shopping service example looks something like:

```
// A function to collect user browsing data
// and notify a clickstream service
void getLogInfo() {
  if (UPM("other_recipient.clickstream") == 1) {
    log += getUserIP();
    log += getRequestTime();
    log += getHTTPStatus();
    clickstreamService.notify(log);
  }
  else {
    log += "clickstream data collecting "
           "not authorized by user";
  }
}
```

We have kept the enforcement technique purposefully simple for easier transition. The design of the UPM is simple enough to make this transition possible. More complex enforcement strategies such as the use of security automata [16] are needed for enforcing more expressive feedback messages. For example, such monitors are needed to enforce policies such as "information is disclosed to either one of two recipients but not both". Having such more complex enforcement schemes represent trade-offs in the expressiveness of the enforcement mechanism versus its complexity. We will study such monitors in our future work.

V. Conclusion

In this paper, we have proposed a P3P extension to support active user feedback and thereby performs policy negotiation. Our main contributions include:

- the introduction of a policy negotiation mechanism by which a user can negotiate a P3P policy with a web site based on her privacy preferences;
- the introduction of user preference matrix (UPM) which bridges gap between the user preferred P3P policy and the service code and thereby serves as a starting point for enforcing P3P policy in the service code.

To adopt the proposed P3P extension, the following is a summary of changes needed to be done on both the P3P clients and servers: a) The P3P policy needs to be updated to specify the user-configurable part of the policy; b) The user agent needs to be updated to support the generation and communication of active user feedback; c) A feedback agent on the server needs to be introduced to process the user feedback and populate the UPM; d) The service code needs to be updated to enforce the user's privacy preference as stated in the UPM.

The proposed P3P extension can be deployed in an environment that contains both the enhanced and original P3P-enabled clients and servers for reasons of scalability and also to enable a smoother transition. This P3P extension has been designed so that the original P3P user agents can interact with a web server enabled with the enhanced P3P for processing P3P policies and policy reference files in a usual way. The P3P extension mainly reuses the syntax available in the original P3P specification for this purpose. Some additional syntax introduced in the extension is presented as an optional features in the original P3P. To provide the backwards compatibility, each service code also runs in a "compatible" mode, in which if the additional UPM inputs are not available, the service code can switch back to non-P3P means such as HTML forms to read a user's choices on the optional functionalities provided by the service.

Acknowledgments. We thank R. Sekar for his insightful suggestions on an earlier draft. This research was supported in part by NSF grant NSF CNS-0551660, a grant from Computer Associates and an ONR grant N000140110967.

References

[1] L. Cranor et al, *The Platform for Privacy Preferences 1.1 (P3P1.1) Specification*, W3C working draft, February 2006.

[2] "P3P compliant web sites," Internet web site, Updated June 2006, http://www.w3.org/P3P/compliant_sites.php3.

[3] Http://www.serviceoriented.org/ws-privacy.html.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Implementing P3P using database technology," in *International Conference on Data Engineering (ICDE)*, March 2003.

[5] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *proceedings of the IEEE*, September 1975.

[6] L. Cranor et al, *A P3P Preference Exchange Language 1.0 (APPEL1.0)*, W3C working draft, April 2002.

[7] T. Yu, N. Li, and A. I. Antón, "A formal semantics for P3P," in *ACM Workshop on Secure Web Services*, October 2004.

[8] "Privacy bird," Internet web site, http://www.privacybird.com.

[9] G. Hogben, "A technical analysis of problems with p3p v1.0 and possible solutions," position paper for W3C Workshop on the Future of P3P. Available at http://www.w3.org/2002/p3p-ws/pp/jrc.html.

[10] G. Hogben, "Suggestions for long term changes to p3p," position paper for W3C Workshop on the Long Term Future of P3P. Available at http://www.w3.org/2003/p3p-ws/pp/jrc.pdf.

[11] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "An XPath-based preference language for P3P," in *International World Wide Web Conference (WWW)*, May 2003.

[12] P. Kolari, L. Ding, S. Ganjugunte, L. Kagal, A. Joshi, and T. Finin, "Enhancing web privacy protection through declarative policies," in *IEEE Workshop on Policy for Distributed Systems and Networks (POLICY 2005)*, June 2005.

[13] K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis, "Protecting privacy during on-line trust negotiation," in *2nd Workshop on Privacy Enhancing Technologies*, April 2002.

[14] Carnegie Mellon University, "Privacy server protocol project," Internet System Laboratory, Robotics Institute and eCommerce Institute, School of Computer Science. http://yuan.ecom.cmu.edu/psp/.

[15] W. Xu, V.N. Venkatakrishnan, R. Sekar, and I.V. Ramakrishnan, "A framework for building privacy-conscious composite web services," in *4th IEEE International Conference on Web Services (Application Services and Industry Track) (ICWS)*, September 2006.

[16] F. B. Schneider, "Enforceable security policies," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 1, 2001.