2011 Sixth International Conference on Availability, Reliability and Security

# A server-side approach to privacy policy matching

Åsmund Ahlmann Nyre, Karin Bernsmed
*SINTEF ICT*
*Trondheim, Norway*
{*asmund.a.nyre, karin.bernsmed*}@*sintef.no*

Solvår Bø, Stian Pedersen
*Norwegian University of Science and Technology (NTNU)*
*Trondheim, Norway*
{*solvarb,stianren*}@*stud.ntnu.no*

***Abstract*—With the increasing use of online services that require sharing of information there is a need for Privacy Enhancing Technology tailored for personal information control. Commonly, web privacy is handled through matching of privacy policies and user preferences using software agents on the client side. In this paper, we propose a new approach to privacy policy matching we denote *server-side* matching. By moving the matching logic from the client to the server, the client is alleviated from the resource consuming process of obtaining and matching policies and the service provider is able to adapt services to users' privacy preferences. We describe the architecture of a general solution and a prototype implementation of selected parts. The solution has only been subject to rudimentary testing, but our initial evaluation is promising.**

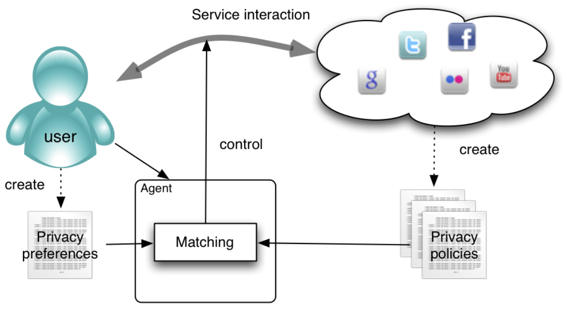


Figure 1: The basic architecture for matching privacy policies and preferences through the use of a user agent.

## I. Introduction

The future Internet is expected to become more service-oriented, have tighter social networks and an increased usage of context-aware applications, where the utility value is dependent on sharing of personal information. Privacy will therefore continue to be one of the main concerns for users on the Internet [1], [2]. Privacy Enhancing Technologies (PETs) for the web have thus far been focused on protecting regular computer mostly focused on hiding user identities by providing anonymity whereas little has been done to assist the users to understand the risks of releasing information about themselves. To protect their privacy it is of major importance that the users understand the implications of data sharing.

Research shows that although users are in general concerned about their privacy, they currently do not act according to their privacy principles. This is often referred to as the privacy paradox and is often attributed to their lack of knowledge and understanding of privacy issues [3]. We argue that privacy in the context aware, mobile world is a task far too difficult for users to handle by themselves. Instead we propose a shift from the client- and device-centred approach of agent-based privacy protection, to a *server-side privacy* solution where privacy protection is offered by the service provider. Hence, users do not have to decide whether to accept the privacy policy of a service, the service has to figure out how to fulfil the preferences of the user. This shift in setup will not only alleviate users from the burden of conducting continuously privacy decisions, but also provide a much needed incentive for service providers to implement privacy protection. By adapting and differentiating their services to the privacy preferences of their users, services providers are able to reach a broader audience than before. We provide an initial design of such an architecture and also an implementation of selected parts of our design, as a proof-of-concept.

This paper is organised as follows. Section 2 reviews related work and explains how our approach differ. Section 3 explains the concept of "server-side privacy" and our proposed design. In Section 4 we describe our prototype implementation of selected parts of the design, and then the evaluation is given in Section 5. Next, we discuss the advantages and disadvantages of our solution in Section 6 and outline the main directions for future work in Section 7, before we provide our concluding remarks in Section 8.

## II. Related work

The Platform for Privacy Preference Project (P3P)[4] was launched to assist users in evaluating privacy policies on the web. At a later stage A P3P Preference Exchange Language (APPEL)[5] was developed to automate the process of privacy policy evaluation by allowing users to specify their privacy preferences. Hence, whether a web site was to be accepted or not, was merely a question of whether the user preferences matched the web site's policy. The P3P/APPEL specifications suggest an architecture where the matching is done by an autonomous user agent (see Figure 1), but such an agent is not part of the specifications themselves.

The Privacy Bird [6][1] developed by AT&T was one of the first user agent implementations based on P3P. It was realised as a web browser plugin (Internet Explorer)

[1]Available from http://www.privacybird.org

and allowed users to specify preferences on a subset of the APPEL specification and perform automated matching with web sites' policies. Evaluation results was presented to users through a a bird icon that changed shape and colour depending on the presence of a P3P policy and the degree to which it matched the preferences. A similar approach was taken by Kolter et al. [7] but extended with more fine-grained preference specification and community support to further assist users in policy evaluations.

The approach taken by the PRIME project[2] is to create a privacy middleware for service providers to run their services on and a corresponding user client for users to interact with such services. The Integrated Privacy View (IPV) [8] takes a somewhat different viewpoint, where the service provider is required to map every input field on the website to the corresponding privacy statements in the policy. This is done by adding *privacy anchors* (reference to a policy statement) within the HTML encoding. Hence, users can, per input field, determine whether the information requested is in accordance with their preferences. Although these approaches implies that greater responsibility is placed on the service provider, the matching and privacy decisions are still performed by the client.

Common for all these solutions is the bureden placed on the users necessary to state, maintain and verify that their privacy preferences are being met.

## III. A SERVER-SIDE PRIVACY ARCHITECTURE FOR ADAPTIVE SERVICES

In this section we describe our proposed architecture for privacy policy matching. The main idea is to transfer the responsibility of obtaining and matching privacy policies with user preferences to the services providers. The purpose is to release the user from privacy preferences management and to minimize the client-side software that has to be installed, while at the same time giving service providers the ability to adapt their services to users.

The purpose of this section is to outline a generic design. Specific implementation details are therefore left out from this part. An example of a prototype implementation of the proposed architecture will be provided in the next section.

### A. Overview

Figure 2 shows an outline of our solution and how the different roles interact. The core concept involves three different roles; *users*, *privacy preference providers*, and *service providers*.

*1) User:* The user selects a set of privacy preferences that matches his/her intention to share personal data. The privacy preferences is used to put restrictions of what type of data the user find acceptable to share, how the data can be used, how long it should be kept by the service providers and with whom the service provider is allowed to share the data. The user will then refer to a particular set of these privacy preferences when interacting with service providers.

[2]Privacy and Identity Management for Europe – http://www.prime-project.eu

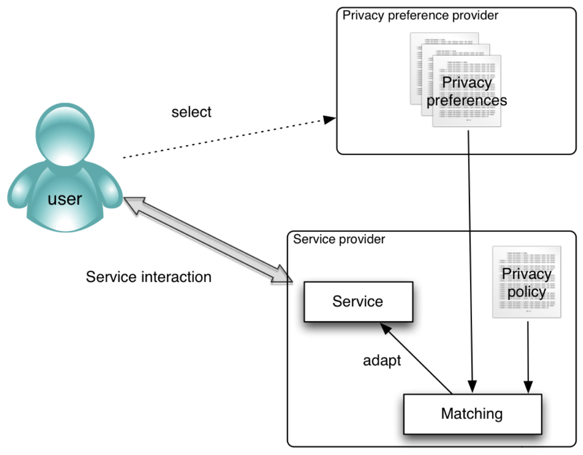


Figure 2: Design overview of server-side privacy

*2) Privacy preference provider:* To aid the user in the process of specifying privacy preferences, our model utilises a privacy preference provider. This entity holds a database of pre-defined privacy preference sets that the user can choose from. The database contains sets that are designed to match the privacy preferences of most stereotype users. For example, to comply with users with strong reluctance to reveal information about themselves the preference provider offers a set of preferences that put very strong restrictions on the collection and usage of personal data. Other data sets contain less strict requirements. The privacy preferences sets are stored and made available online, without being associated with any particular user.

*3) Service provider :* In our model, privacy-aware service providers will be notified, at session initiation, when users have privacy preferences stored at a preference provider. The main task of the service providers is then to collect the privacy preferences set from the privacy preferences provider, compare the preferences with their privacy policy and to adapt their services accordingly.

The general process flow is given below. Step 1 is considered an initialisation step and is only performed once, whereas steps 2–6 are repeated whenever the users issues a request for a service.

1) The user specifies his privacy preferences through a preference provider by selecting a predefined preference file. The corresponding preference identifier is stored at the client.
2) When requesting a service, the client transfers the stored preference identifier to the service provider.
3) The service provider makes a request to the privacy preference provider to obtain the preference file corresponding to the identifier received from the client.
4) The preference file is then matched against the privacy policy of the service.
5) Whenever the privacy policy does not match the preferences of the user, the service provider attempts to adapt the service such that a match is obtained.

6) The outcome of the process is either the original service, a limited service or no service at all. In any case, the service offered to the user will adhere to his privacy preferences.

The architecture proposed in Figure 2 is generic and not tied to any specific implementation. The roles describe logical roles and there may be situations in which one entity operates in two different roles. For example, a service provider may also operate as a privacy preference provider. Or a user may chose to create and store his own privacy preferences. The seperation of roles in the figure does therefore not imply different entities.

### B. Preference specification

Preference generation could be handled in many different ways that would still adhere to our proposed architecture. However, there are some fundamental features that must, or must not, be in place for it to be an acceptable privacy solution. An obvious approach is to simply transfer whatever preference solution currently available (e.g., [7], [6]) and simply transfer the preference file to provider's fileserver. However, this would either mean that the user would have to specify such a preference file on his own, or that preference generation software would need to be installed. Neither solutions is ideal in our opinion. Additionally, the content of a highly personalised preference file would of course be considered personal information and should therefore not be distributed to anyone without any protection. Hence, the preference solution itself could become a threat to the privacy of its users. Through an evaluation meeting with the Norwegian Data Inspectorate it was pointed out that to avoid being a potential privacy liability, a number of people should have identical preferences. That is why we propose to have preference providers offering a fixed set of privacy preferences for users to choose from. This will not only solve the problem of personal preferences, but at the same time help users to specify preferences without requiring additional client-side software.

Still, one of the main problems of privacy preference specification is that the decisions humans make are based on the context in which it is done. Thus, users may have entirely different attitudes towards privacy for example in their professional and private life. To cater for this, users are allowed to select multiple preference files to be used in different contexts.

### C. Privacy policy language

Both privacy policies and preferences must be provided in a format that is machine understandable, in order to facilitate matching. The only assumption we make regarding such languages is that the service provider is able to perform matching. That being said, the preference language need not be particularly expressive since preferences must be kept general in order to limit the number of different preference files available to the users. As a consequence, any expressiveness beyond what is offered as preferences, is impossible to take advantage of when specifying privacy policies.

### D. Client software

While we strive to prevent users from having to download and install client software for our solution to work, it apparently is unavoidable. The reason is that the service provider must be notified of which preferences the user has selected. Hence, the basic responsibility of the client is to record the preference identifier selected by the user and then forward this to the service provider whenever the user issues a request for a service. Additionally, if the user can select multiple preference identifiers for different contexts, the client must also do this.

In order to maintain a light-weight client, context switches are handled by direct input from the user. For example, the user can set the client to be in *private*-mode or *work*-mode to turn on the privacy preferences selected for the private and professional life, respectively. This greatly reduces the number of preferences a user can select and handle efficiently, but at the same time ensures that the complexity of the client is kept at a minimum.

To prevent any unnecessary processing delay, all requests made by a user should contain the preference identifier, regardless of whether the service provider in question supports the technology or not.

### E. Policy matching and service adaptation

Service tailoring and context-aware services are becoming increasingly popular due to their improved usefulness. But at the same time, such services are often considered problematic with respect to users privacy. In our solution, we utilise users' privacy preferences as the basis for service adaptation and thereby neutralising the conflict between usefulness and privacy protection.

We assume a tight mapping between service features and privacy policy statements. Hence, whenever a policy statement is in conflict with a preference rule, the service provider is capable of removing the features corresponding to the statement. For example if the privacy policy states that click-stream information is used for tailoring the service and the user's preference states that it will not allow click-stream information to be collected, then the service can remove the tailoring feature and thereby deliver a service in accordance with the preferences. Obviously, if the statement in question corresponds to all features of service, then the service should not be offered to the user.

Whenever features are removed from a service due to conflicting privacy preferences, the user is made aware of this through the user interface of the service. Unlike many agent-based technologies (e.g. [6]) our solution does not rely on any specific icons to convey matching results to users. This is because the service offered to the user by definition is in accordance with his preferences, and that the user is more likely to demand an explanation for missing features than is the case for traditional policy matching.

## IV. Prototype implementation

In this section we elaborate on our implementation of the concept described in the previous section. As a proof-of-concept application, we have only implemented parts of the proposed deign.

### A. Setup and simplifying assumptions

We chose to use P3P [4] as the privacy policy language and APPEL [5] for preferences. This choice was made primarily based on the ease of use, simplicity and tool support available for P3P/APPEL. The server-side functionality was implemented using the Python[3] programming language on a traditional Apache web server[4].

Since a full-scale end-user test was not within reach of this project, we settled for a rudimentary implementation of the preference provider service. One of the main parts of such a service would be the user interface and the actual preference selection. But, since such an interface could not be thoroughly tested, the service was implemented as a standard web server solely for the purpose of retrieving preference files at the request of the service provider. Hence, our implementation assumes that the user already has selected a preference file, and therefore only consider the event where the user requests a service from a service provider.

### B. Client software

The sole purpose of the client software is to record the selected preference files and transfer the preference identifier to the service provider upon making a request. Hence, the software itself should be as small as possible to prevent it from becoming an obstacle in user adoption. We therefore decided to use web cookies [9] and a small web browser plug-in for transferring the preference identifier to the service provider. The idea is that the preference provider sets a cookie containing the preference identifier (a URL) when a preference file is selected by the user. Afterwards, whenever the user requests a web page, the client software creates a new cookie for the requested domain and inserts the preference URL. The web browser will ensure that the newly created cookie is transferred to the host along with the initial request. Hosts that do not implement server-side matching will simply ignore the added information.

A perhaps better option for transferring the preference URL would have been to extend the HTTP [10] request header to contain a new field. The problem would be that an extended HTTP request header would have had to be implemented by the web browser itself and not by a plug-in software. Thus, for our purposes, the cookie-based approach best serves our needs.

### C. Privacy policy matching

Once the service provider receives a request containing a privacy preference cookie, it will automatically download the preference file corresponding to the URL contained in the cookie. Hence, the service provider need not be aware, nor have any agreements with the preference provider other than the ability to download the file. The preference file along with the service providers policy file is then fed to the matching engine.

[3]http://www.python.org
[4]http://www.apache.org

Privacy policy matching is done by comparing the preference rules to the statements of the policy. In the examples depicted in Figure 3, the preference rule states that any service collecting *physical* information to distribute publicly, to unrelated services, delivery services or others, should be blocked. Similarly, the policy statement indicates that the service will collect both physical, online, demographic and purchase data to be given to the service itself or a delivery service. Thus, there is a match between the categories *physical* and the recipient *delivery*, which triggers the preference rule to *block*.

### D. Service adaptation

In the event that the matching algorithm detects a conflict, service adaptation is attempted to resolve the conflict. Continuing the example from Figure 3, the service could be adapted by either removing the data category *physical* or the recipient *delivery* and all corresponding features. Our prototype implementation of a web service contain a fictitious online shop where we have chosen to remove the delivery recipient and hence all corresponding features. Thus, the service is obliged not to deliver physical information to any delivery services, which is essential in order for the shop to send bought items to the customers. Therefore, the service cannot offer the customer to buy anything, but he may still view the items on sale. Figure 4 and 5 depict the resulting web page from a matching process without conflicts and the one identical to the above-mentioned example. The latter shows that the buying feature has been removed and is accompanied by a message stating the cause of the removal and how the buying ability may be reinstated.

## V. Discussion

Service providers currently have no incentive to implement privacy enhancing technology which serves as a partial explanation for the relatively modest adoption of P3P [11]. At a first glance, placing an additional burden on the service provider in terms of both implementation effort and resource consumption at run-time does not seem to contribute to the service provider buy-in. However, the ability to adapt services to the preference of users and also more concretely show, through feature removal, what features rely on the different parts of the privacy policy. The current implementation is not sufficient to perform a proper trade-off analysis of the costs and benefits of our solution.

HTTP Cookies have often been linked to privacy issues on the web. However, commonly they constitute a part of the problem, not part of the solution. But, as long as the information contained in the cookie is not personal or does not identify any persons, the use of cookies does not threat privacy. Another problem might be that our use of cookies is not entirely according to the specification [9]. Particularly, the server is supposed to set a cookie using a set-cookie field in the response header, whereas

```
<appel:RULE behavior="block">
  <p3p:POLICY>
    <p3p:STATEMENT>
      <p3p:DATA GROUP>
        <p3p:DATA>
          <p3p:CATEGORIES
             appel:connective="or">
           <p3p:physical/>
          </p3p:CATEGORIES>
        </p3p:DATA>
      </p3p:DATA GROUP>
      <p3p:RECIPIENT
          appel:connective="or">
          <p3p:other-recipient/>
          <p3p:public/>
          <p3p:unrelated/>
          <p3p:delivery/>
        </p3p:RECIPIENT>
      </p3p:STATEMENT>
    </p3p:POLICY>
</appel:RULE>
```

(a)

```
<STATEMENT>
  <PURPOSE>
    <current/>
    <develop />
  </PURPOSE>
  <RECIPIENT>
    <ours/>
   <delivery/>
  </RECIPIENT>
  <DATA-GROUP>
    <DATA ref="#dynamic.miscdata">
      <CATEGORIES>
        <physical/>
        <online/>
        <demographic/>
        <purchase/>
      </CATEGORIES>
    </DATA>
  </DATA-GROUP>
</STATEMENT>
```

(b)

Figure 3: Simple examples of (a) an APPEL-based preference file and (b) a P3P-based policy file .

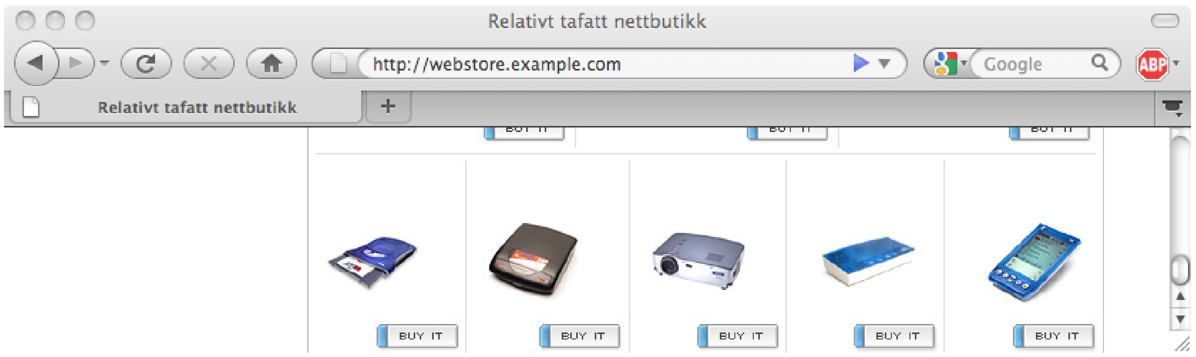

Relativt tafatt nettbutikk
http://webstore.example.com
Google
Relativt tafatt nettbutikk
BUY IT
BUY IT
BUY IT
BUY IT
BUY IT

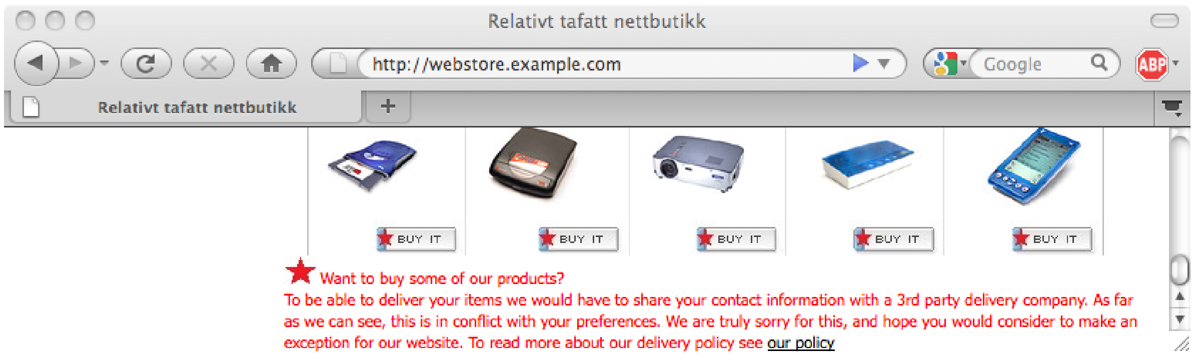
Figure 4: Service adaptation based on matching result: privacy preferences and policy match

in our solution the cookie is set by the client software upon requesting a service from the provider. Additionally, cookies was invented to provide state management to the stateless HTTP [9] and not as a general parameter passing between client and host. Although we seemingly breach both privacy and the intended use of cookies, we argue that when used properly, cookies do not pose a significant threat to privacy, and that our use of cookies is a way to handle the state in which a user has selected preferences.

In the U.S., a central idea in privacy protection is the right to opt-out of information collection that does not directly correspond to the service. Previous evaluations of agent based privacy enhancing technologies have stressed the difficulties users have with opt-ing out [12]. Our approach removes the need for opt-outs since this is done by default if preferences state that such information collection is not allowed. Hence, our solution does provide automatic opt-out, a feature that was deemed impossible by the Privacy Bird development team due to the limitations of P3P [6].

It is essential for our system to prevent the preferences sharing from becoming a privacy liability. In general, the requirement must be that the selected preferences must be considered open to any recipient such that sharing them with a service provider in itself does not constitute a privacy breach. In case anonymity is required this requirement would imply that a minimum of users share the same preference ID. By attempting to cater for more individual preferences there is a risk that the number of preference identifiers increase so much that the likelihood of sharing preferences with other users becomes to low.

By transferring the logics from the user to the service provider, one may argue that the user looses control as the provider may simply adapt the policy to match the user preferences. However, in the traditional case of privacy policy matching, the provider may not be truthful about its privacy policy. Hence, the user may think he is in control of the matching, but he may be matching based on a false policy. We therefore argue that our solution does not in effect constitute a major transfer of control from the user to the provider. All policy based approaches to privacy effectively rely on the service providers to be truthful about their policies for the system to work. We further believe that the ability to adapt a service to users' preferences provides an incentive for the service provider to behave more honestly, since a wider range of users may use a limited service as opposed to no service at all.


Relativt tafatt nettbutikk
http://webstore.example.com
Google
Relativt tafatt nettbutikk
BUY IT
BUY IT
BUY IT
BUY IT
BUY IT

Want to buy some of our products?
To be able to deliver your items we would have to share your contact information with a 3rd party delivery company. As far as we can see, this is in conflict with your preferences. We are truly sorry for this, and hope you would consider to make an exception for our website. To read more about our delivery policy see our policy

Figure 5: Service adaptation based on matching result: privacy preferences and policy do not match

## VI. Future Work

There are several aspects of our solution that need more work. The prototype implementation is rudimentary in some aspects, particularly the preference specification and client software constitute an important area of future work in order to conduct full end-user testing and verification of the approach. The preference provider could be worth an investigation of its own, especially the user interface for selecting and specifying privacy preferences. The client should not only be implemented as a prototype, but also tested with different solutions for transferring the preference identifiers.

## VII. Conclusion

We have described a new approach to privacy protection on the web. Our solution places more responsibility on the service provider for ensuring that user's privacy is adequately protected, but at the same time also enables service providers to benefit from this protection by adapting their services to the users' preferences. The prototype implementation of parts of this approach serves as a proof-of-concept, but requires additional development in order to provide sufficient evidence of suitability. However, we still believe the server-side approach to privacy policy matching is a viable solution and none of our findings have directly contradicted this.

## References

[1] J. Bryce and M. Klang, "Young people, disclosure of personal information and online privacy: Control, choice and consequences," *Information Security Technical Report*, vol. 14, no. 3, pp. 160 – 166, 2009, the Changing Shape of Privacy and Consent. [Online]. Available: http://www.sciencedirect.com/science/article/B6VJC-4XMK24J-1/2/b3d8b9a37b4a4154822fafdf6ff29d47

[2] S. Chai, S. Bagchi-Sen, C. Morrell, H. Rao, and S. Upadhyaya, "Internet and online information privacy: An exploratory study of preteens and early teens," *Professional Communication, IEEE Transactions on*, vol. 52, no. 2, pp. 167–182, June 2009. [Online]. Available: http://dx.doi.org/10.1109/TPC.2009.2017985

[3] B. Berendt, O. Günther, and S. Spiekermann, "Privacy in e-commerce: stated preferences vs. actual behavior," *Commun. ACM*, vol. 48, no. 4, pp. 101–106, 2005.

[4] "W3C. Platform for Privacy Preferences. http://www.w3.org/P3P/."

[5] L. Cranor, M. Langheinrich, and M. Marchiori, *A P3P Preference Exchange Language 1.0 (APPEL1.0)*. World Wide Web Consortium, 2002. [Online]. Available: http://www.w3.org/TR/P3P-preferences/

[6] L. F. Cranor, M. Arjula, and P. Guduru, "Use of a p3p user agent by early adopters," in *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*. New York, NY, USA: ACM, 2002, pp. 1–10.

[7] J. Kolter, T. Kernchen, and G. Pernul, "Collaborative privacy management," *Computers & Security*, vol. 29, no. 5, pp. 580–591, 2010.

[8] S. E. Levy and C. Gutwin, "Improving understanding of website privacy policies with fine-grained policy anchors," in *Proceedings of the 14th international conference on World Wide Web (WWW '05)*, 2005.

[9] D. Kristol and L. Montulli, *RFC2965: HTTP State Management Mechanism*. IETF Standards track, 2000. [Online]. Available: http://tools.ietf.org/html/rfc2965

[10] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *RFC2616: Hypertext Transfer Protocol – HTTP/1.1*. IETF Standards track, 1999. [Online]. Available: http://tools.ietf.org/html/rfc2965

[11] S. Egelman, L. F. Cranor, and A. Chowdhury, "An Analysis of P3P-Enabled Web Sites among Top-20 Search Results," in *In ICEC '06: Proceedings of the 8th international conference on Electronic commerce (New*. ACM Press, 2006, pp. 197–207.

[12] L. F. Cranor, P. Guduru, and M. Arjula, "User interfaces for privacy agents," *ACM Trans. Comput.-Hum. Interact.*, vol. 13, no. 2, pp. 135–178, 2006.