Privacy Interfaces for Information Management

Tessa Lau, Oren Etzioni, Daniel S. Weld
University of Washington
Department of Computer Science & Engineering
Box 352350
Seattle, WA 98195 USA
{tlau,etzioni,weld}@cs.washington.edu

ABSTRACT
To facilitate the sharing of information using modern communication networks, users must be able to decide on a privacy policy—what information to conceal, what to reveal, and to whom. We describe the evolution of privacy interfaces—the user interfaces for specifying privacy policies—in COLLABCLIO, a system for sharing web browsing histories. Our experience has shown us that privacy policies ought to be treated as first-class objects: policy objects should have an intensional representation, and privacy interfaces should support direct manipulation of these objects. We also show how these conclusions apply to a variety of domains such as file systems, email, and telephony.

Keywords
Privacy, user interfaces, direct manipulation, WWW, information retrieval, intensional/extensional set representations.

INTRODUCTION
It is commonplace that modern communication networks should support the sharing of information while protecting people's privacy. To this end networks provide mechanisms that enable each user to specify a *privacy policy*—what information to conceal, what to reveal, and to whom. For example, file systems support protection modes for directories and documents, and the telephone network provides Caller ID but enables callers to make themselves anonymous.

However, we observe that today's *privacy interfaces*—the user interfaces to these privacy mechanisms—are woefully inadequate. A user with a particular privacy policy in mind often lacks a convenient means for enforcing it. For example, there is no way to instruct one's phone to "ring if the call is from a friend or family member, but forward everyone else to the answering machine"; a user must either screen each call individually or forward all calls to the answering machine. Similarly, to share files in Windows NT or UNIX, one must individually manipulate each file and folder.

Today's privacy interfaces are based on properties of individual objects—to enforce a general privacy policy, each affected item must have its privacy property set individually. This is inconvenient for large numbers of items. For example, users refuse to set a "protection" on each e-mail message they receive. Moreover, users do not have the ability to proactively specify complex policies that will cover messages not yet received. For the sake of convenience, people default to coarse-grained privacy policies where all potentially sensitive information such as e-mail is kept under "lock and key."

In this paper, we recount the evolution of privacy interfaces in COLLABCLIO—a system that supports automated sharing of Web browsing histories. COLLABCLIO stores a person's browsing history indexed by keyword (and other attributes). A typical COLLABCLIO query might be "Show me all the pages visited in the `.edu` domain containing the phrase `direct manipulation`." Significantly, COLLABCLIO users can make queries against the browsing history of other users, raising obvious privacy concerns.

We have redesigned COLLABCLIO's privacy interface several times in response to user feedback. Our experiences led us to conclude that privacy policies should be first-class objects that are easy to create, inspect, modify, and monitor. They should have a compact representation and apply proactively to items not yet created.

These conclusions are sufficiently broad as to apply to such diverse domains as the Windows NT file system, e-mail, and telephony. We have analyzed the privacy interfaces in each of these domains, and show how they fail to treat privacy policies as first-class objects.

RELATED WORK
Several systems have already addressed the problem of sharing URLs. Warmlist [8] is similar to COLLABCLIO in that it automatically indexes the content of web pages stored in a user's bookmark list. However, the only facility for sharing URLs with other users is by indirectly importing other users' Warmlists. Several other systems allow users to share URLs with one another directly.

WebTagger [7], Grassroots [6], and Jasper [2] provide facilities for sharing bookmarks between colleagues via a

centralized repository. By requiring users to explicitly choose which bookmarks are shared with which people, however, they expect each user to anticipate the interests of her colleagues, and manually enter the web page into the system each time she comes across something which ought to be shared.

Another area of related work concerns the issue of privacy in conjunction with collaborative systems involving live video feeds. Bellotti [1] proposes a framework to guide the incorporation of privacy into the design of collaborative systems, as well as a set of criteria for evaluating such systems; the evaluation section discusses our experience with COLLABCLIO in light of several of her criteria. In addition, Hudson and Smith [4] study privacy in conjunction with a video awareness system. The shadow-view technique (representing areas of recent motion in a live video feed with darkened squares) implements a compact and proactive policy for video privacy.

Yenta [3] is a distributed agent system that uses email and other sources to build up a profile of a user; this profile is used to automatically match users with similar interests. While Yenta was designed to provide information security using encryption, it does not directly address the issue of information privacy which we are concerned with here.

COLLABCLIO

We now turn to the evolution of COLLABCLIO—our testbed for investigating privacy interfaces. We begin by describing COLLABCLIO's domain.

Web History Domain

As we access more and more information via the web, standard navigation solutions such as bookmarks and favorites become less adequate for enabling us to find our way back to pages of interest. There is a clear need for a technology to mitigate the "lost in hyperspace" phenomenon, and facilitate the retrieval of useful links.

In response to this need, we developed CLIO—a program that automatically indexes the content of web pages that its user visits. CLIO runs on a person's workstation and captures her browsing history automatically. A user can search her CLIO for pages which she has previously visited by describing the desired web page in terms of attributes such as the page's keywords, parts of its title and URL, etc. For example, one could search for "All pages visited in the last two weeks that contained the keywords `privacy` and `security`."

In addition to retrieving URLs for personal use, we often need to share URLs with colleagues. Most approaches to this problem have centered around shared bookmark lists (for example [2, 6, 7]). However, such approaches require a user to anticipate which URLs may be of interest to others, and to manually enter such URLs into the system.

To support URL sharing, each user may elect to register her CLIO with a centralized server; any CLIO can be contacted at any time to service a "remote" query on another user's behalf. This network of CLIOs makes up the COLLABCLIO system. Thus, a user can ask his CLIO to query his colleagues' CLIOs in order to discover who has visited web pages with certain attributes. For example, one could search for "All pages which Joe has visited that contain the phrase `collaborative filtering`." To discover pet owners, one might search everyone's CLIOs for "All pages containing the word `cats`."

Of course, remote queries against users' web browsing histories might reveal information that they would prefer to hide from other users, such as stock quotes, class grades, fetishes, or health concerns. Logically, pages in one's web browsing history can be partitioned into equivalence classes based on the groups of people who are authorized to see those pages. In the simplest case, there are two classes: private web pages (those which should be visible only to the owner) and public pages (those which can be shared with other people). Clearly, COLLABCLIO should only return public pages in response to remote queries. To allow users to classify pages as public and private, we developed a privacy interface for COLLABCLIO.

Example-based privacy interface

Our first privacy interface design consisted of two mechanisms: a record light and a search-and-mark tool. The record light widget (Figure 1) is designed to be kept onscreen near a user's web browser window. It is based on the idea of the red light on a video camera. When the light is on, actions (web page visits, in this case) are recorded as public; when the light is turned off, web pages are recorded as private. Users can toggle the status of the record light at any time; this action changes the classification of the page currently displayed in the browser. The record light is sticky: once it has been toggled, it remains in that state until the user explicitly toggles it back. The use of this record light interface lets users classify every web page immediately as either public or private.

Figure 1: Screenshots of the record light window. The top window is displayed when the record light is toggled to PRIVATE, the bottom when it is set for PUBLIC.

The second mechanism, the search-and-mark tool, was meant to be used in conjunction with the record light, as a method of reviewing and amending previous decisions. Once web pages have been indexed into the COLLABCLIO system, a user can use the search-and-mark mechanism to change web page classifications. She can do this by using CLIO to search her history, selecting one or more of the returned pages, and marking them as either public or private.

An informal user survey, however, revealed that this interface was inconvenient to use for certain types of privacy policies. For example, to implement the policy "Hide all visits to web pages in the `.com` domain," a user would have to remember to either toggle the record light to private each time a `.com` site was visited, or to periodically search for all `.com` pages and mark them private at a later date. In the first case, the number of actions on a user's part increases with the number of sites visited. The second case is equivalent to choosing a coarse-grained default protection and periodically refining it to better reflect the desired privacy policy. Furthermore, in this example there is a window of time during which private information could be revealed: a remote query occurring after a page was visited, yet before it was marked as private, might reveal sensitive information.

In addition, some users found it hard to go back and visualize their privacy policy; there was no way to list or summarize all the private web pages in one's history.[1] Another criticism noted that the record light wasn't proactive: if a site was marked private in the past, subsequent visits to the site weren't automatically marked private, but were classified according to the current setting of the record light.

Lessons Learned

In considering these concerns, we realized that the root of the problem was the lack of an explicit privacy policy object: privacy was treated as a *property* of individual documents, not as an *object* in its own right. We hypothesized that a person's privacy policy ought to a first-class object with a compact representation; it should apply proactively to items not yet created. The privacy interface should make it easy to create, inspect, modify, and monitor such objects.

To clarify this objective, we introduce the distinction between intensional and extensional representations of privacy policies [5]. An *extensional* representation describes a set by enumerating the items in it (such as a list of all private web pages). The record light interface creates an extensional privacy policy. In contrast, an *intensional* representation describes a set by characterizing the objects in the set. Consider, for example, the policy "Hide all web pages that contain the word `sex`." Using an extensional privacy interface such as the record light, one must notice when a web page contains the word `sex`, and toggle the record light accordingly. On the other hand, one would be able to state this policy explicitly in an intensional representation, and rely on the system to enforce it. Furthermore, the interface could "remember" the policy and apply it to future web pages as they are being visited and indexed by COLLABCLIO.

Although the search-and-mark mechanism gives the impression of an intensional representation, in fact it preserved the extensional representation used by the record light mechanism. Privacy was still implemented as a property of each document. One user was surprised to hear that although he had used the search-and-mark mechanism once to classify `.com` sites as private, future visits to `.com` sites were not automatically classified as private.

These considerations led us to develop a privacy interface for COLLABCLIO that supported an intensional privacy policy representation. This interface is described in the next subsection.

Rule-based privacy interface

COLLABCLIO's second privacy interface centers around the privacy policy editor window (Figure 2). This window supports the creation, inspection, and modification of privacy policies. A privacy policy consists of a default protection (either public or private), and a list of rules which describe a set of exceptions to that default.

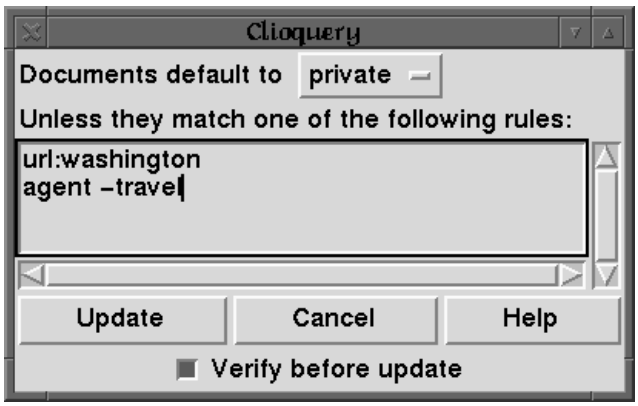

Figure 2: Screenshot of the privacy policy editor interface.

Each line in the policy represents one rule. Each rule is a list of words, using a syntax similar to that used in popular Web search engines such as Alta Vista and Lycos: a minus sign in front of a word means negation, and a `url:` prefix specifies a match against the document's URL instead of its textual content. There is an implicit conjunction over the words in each line. The union of the sets described by the rules makes up the set of exceptions to the default privacy.

For example, the privacy policy

```
url:washington
agent -travel
```

[1] An intermittent COLLABCLIO user was particularly vociferous about the difficulty of recalling his privacy policy after some time away from COLLABCLIO.

describes a set of documents consisting of all web pages whose URL contains the string `washington`, as well as all web pages that contain the word `agent` but not the word `travel`. If the default policy were private, then the web pages contained in this set would be the only public documents in this user's CLIO.

Since users were not always sure of the exact coverage of the rules they created, we added two monitoring facilities to COLLABCLIO. Figure 3 shows the monitor window that displays the title of each web page and its classification as it is visited in a web browser. In addition, we provide a query-log window that displays which URLs (if any) were returned in response to remote queries. The monitor and query-log windows enable a user to verify that the policy created in the rule-editor window is having the desired effect.

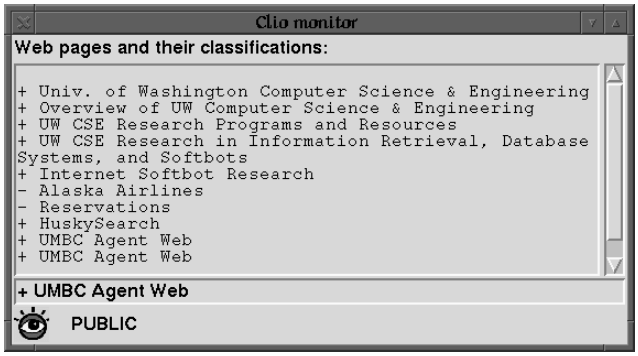

Figure 3: Screenshot of the monitor window. Titles of recently browsed web pages are displayed; they are prefixed by their classification into public (+) or private (-). The icon at the bottom shows the classification of the current web page, displayed just above it.

Direct manipulation of privacy policies

Our claim that privacy policies should be treated as first-class objects leads naturally to the conjecture that a privacy interface should support direct manipulation of privacy policies. This term, first coined by Shneiderman [9], refers to user interfaces with the following three properties:

1. Continuous representation of the object of interest.
2. Physical actions or labeled button presses instead of complex syntax.
3. Rapid incremental reversible operations whose impact on the object of interest is immediately visible.

In this section we show that all of our interfaces embodied some of these properties, but further improvement may be possible.

At first, it may appear that the record light interface satisfies all three properties. However, while the light is indisputably a continuous representation, it represents the privacy policy *applied to a single point* (the web page currently being viewed) instead of the complete policy object itself. Pressing the button in the record light widget causes the function's value at this point to change; this action can be immediately reversed. In a sense this mechanism does support direct manipulation, but one can only view and change the policy piece by piece.

While it does not provide a continuous representation, the search-and-mark interface can be used to directly view and modify a larger fraction of the privacy policy, by selecting a set of web pages. The results of the search are then displayed, along with their privacy classification. A physical action (selecting the desired web pages and activating a menu item) will subsequently modify the privacy policy; this action is reversible by reselecting those objects and choosing a different menu item.

Since it does not have a compact representation, the privacy policy object is not continually represented in either of the extensional interfaces. It would be cumbersome to traverse a long list of web pages and their classifications.

In contrast, the rule-editor interface (by virtue of its intensional representation) provides a continuous, compact textual representation of the privacy policy. The policy can be modified directly in the rule-editor window by editing the text using standard text-editing commands.

Since privacy policies can grow complex, the monitor and query-log windows provide two alternate views of the privacy policy function: monitoring and verification. The monitor window shows how the function classifies web pages as they are browsed. The query-log window verifies that the function is correct by showing which web pages were returned in response to a remote query.

Table 1 summarizes how well each of the privacy mechanisms meets the criteria for direct manipulation.

	Type	Cont. Repr.?	Physical actions?	Reversible w/ feedback?
Record light	Ext	Low	High	High
Search-and-mark	Ext	Low	High	High
Rule-editor	Int	High	Med	Med

Table 1: Privacy interfaces in COLLABCLIO, and how well they satisfy properties of direct manipulation interfaces.

EVALUATION

Of the people using our system, four have been using the record-light and search-and-mark privacy interfaces for several months. In an informal study, we introduced these users to the rule-editor interface, had them use it for an hour, and asked their opinions of the new interface. Based

on our preliminary results, all users preferred the rule-based interface to the example-based interface.

We now analyze COLLABCLIO in terms of Bellotti's [1] criteria for evaluating ubiquitous computing services.

***Flexibility**: what counts as private varies according to context and interpersonal relationships. Thus mechanisms of control over user and system behaviors may need to be tailorable to some extent by the individuals concerned.* We designed flexibility into the system from the start. Privacy policies are configurable, not fixed; the system does not impose any particular privacy policy on anyone. One measure of flexibility is the expressiveness of the language available for specifying privacy policies.

COLLABCLIO's example-based privacy interface is fully expressive since any policy can be specified by toggling the record light between public and private at the appropriate times, or by later changing the classification of web pages using the search-and-mark mechanism.

The rule-based interface is fully expressive with respect to the keyword and URL attributes it supports; any privacy policy which can be expressed in terms of the keywords used in web pages, or their URLs, can be entered into the system.

Although the interface requires that a privacy policy be encoded as a list of conjunctive rules, this restriction does not reduce the expressive power of the interface. The list of conjunctive rules is treated by COLLABCLIO as a *disjunctive normal form* (DNF) formula, i.e. a disjunction of conjunctions. Any boolean logic function can be converted into DNF, thus any policy over these attributes can be expressed in the rule language. In practice, users who were able to express their policies using compact rule sets were satisfied with the ease of creating, inspecting, and modifying their privacy policies using this interface.

Although the rule language is expressive, one user noted that this presented a problem when trying to classify web pages containing little text (for example, pages using HTML frames, or heavily graphics-based pages); this user had to resort to using an inconvenient list of URLs to classify those pages. Otherwise, users found that a compact list of keywords and URL fragments were sufficient to classify most web pages.

***Trustworthiness/Learnability**: systems must be technically reliable and instill confidence in users. In order to do this they must be understandable by their users. Proposed design solutions should not require a complex mental model of how the system works.* While the extensional privacy interface appears to be conceptually simple, several users misunderstood how it worked. The interface implements a model in which each visit to a web page is either public or private; as web pages are visited, they are added to the system with a classification determined by the current setting of the record light. Subsequent use of the search-and-mark mechanism changes the classification of documents already in the system, but this action does not apply proactively to future visits to web pages. Two users mistakenly believed that by using the search-and-mark mechanism to specify a privacy policy, this policy would be applied to web pages seen in the future.

We designed COLLABCLIO's rule language and monitoring facilities so that this interface would be easily understood by users. The interface relies on a simple rule language with a familiar syntax; neither disjunction nor parentheses are allowed in rule antecedents to maintain rule simplicity. In addition, the interface avoids rule conflicts by stipulating that all rules (except the default) classify pages in the same way. If the default is public, all pages matching any rule are private and vice versa. Thus, rule conflicts are impossible and rule ordering has no impact on which pages are private. Finally, the query-log and monitor window help the user to verify that her policy is having the intended consequences.

Although not our main concern, bugs and security issues continue to be an impediment to user acceptance of our system.

***Low effort**: Design solutions must be lightweight to use, requiring as few actions and as little effort on the part of the user as possible.* The record light requires a small, constant amount of effort to classify each incoming web page. As mentioned in the previous section, some users found that this interface required too much effort. In essence, the policy object created using the extensional privacy interface is neither compact nor proactive, thus making the effort required to create, inspect, and modify one's privacy policy too high. For one user, difficulties in visualizing his policy also served to erode his trust in the interface: "if I can't remember or visualize my policy as encoded in COLLABCLIO, how can I be sure it's doing the right thing?"

By virtue of being intensional, the rule-based interface has two key properties that reduce the effort required to use it. First, the interface supports a *compact* privacy policy object—instead of a list of thousands of web pages and their classifications, the policy representation consists of a set of keywords and URL fragments. As the number of web pages in a user's CLIO grows, a simple change to her privacy policy may change the classification of a large number of web pages. Second, the interface is *proactive*—the privacy policy is easily applied to documents as they are added to the user's CLIO. Users agreed that while the rule-based interface required more effort up front to create a policy, over time it required less effort and was more convenient than the original example-based approach.

Ideally we might allow a user to enter single-page exceptions to rules, rather than having to change rules in the privacy policy until every web page is classified correctly. However, this would make rule ordering significant, requiring a more complex mental model to use the rule language.

Appropriate timing*: feedback and control should be provided at a time when they are most likely to be required and effective.* Both interfaces display immediate feedback about the operation of one's privacy policy. If the page is not classified correctly, one may immediately change the privacy policy in order to rectify the problem.

Perceptibility*: feedback and the means to exercise control should be noticeable.* Feedback and control are provided by the record light, monitor, and query-log windows.

Unobtrusiveness*: feedback should not distract or annoy. It should be selective and relevant and should not overload the recipient with information.* See Figures 1, 2, and 3.

Minimal intrusiveness*: feedback should not involve information which compromises the privacy of others.* We adopt the principle that one's privacy policy ought to remain private, despite multiple queries against that person. (We considered numerous "attacks" against a user's privacy policy and designed the query facility to prevent them.) Also, when a person issues an anonymous query against other people, her name is not logged along with the names of people making normal queries in the query-log.

Fail-safety*: in cases where users omit to take explicit action to protect their privacy, the system should minimize information capture, construction, and access.* In both interfaces, the default privacy policy is to classify web pages as private, thus minimizing the amount of information shared with others.

Meaningfulness*: Feedback and control must incorporate meaningful representations of information captured and meaningful actions to control it, not just raw data and unfamiliar actions.* The record light is analogous to the familiar red light on a video camera. The rule language is similar to the language used in major web search engines for finding web pages.

Low cost*: Naturally, we wish to keep costs of design solutions down.* Our implementation is a prototype; the emphasis on low-cost design and implementation reflects a tradeoff in the trustworthiness and reliability of the resulting system.

PRIVACY INTERFACES IN OTHER SYSTEMS

Our experience with COLLABCLIO led us to the conclusion that privacy interfaces ought to support privacy policies as first-class objects, and that compact, proactive privacy policies can best be represented intensionally. To show that our insights can be applied to domains besides web browsing, we use them to critique the privacy interfaces in three different systems: Windows NT 4.0, email, and telephony.

Windows NT 4.0 enables users to share files with colleagues across the network. Each file has associated with it a set of permissions; these permissions grant varying levels of access to listed users. A dialog box displays the list of users/groups allowed to access each file or directory, along with the permissions granted to each one. Checkboxes control whether a setting is applied to the current file, or to all files recursively in this subtree. In this dialog box, users and groups may be added, deleted, or have their access rights modified.[2]

This system treats privacy as a property of files and folders instead of as an object in its own right. Due to this extensional representation, privacy does not scale to large numbers of items. It is difficult to visualize. For example, there is no way to list which files are shared with a particular user in Windows NT 4.0. In addition, the privacy of a particular file depends on its location (whether or not its enclosing folder is shared). Excepting the case where a file inherits permissions from its enclosing folder, privacy is not proactive—it does not apply to items that have not yet been created. There is no way to express a proactive policy such as "Share all Microsoft Word documents, regardless of their location."

Email clients such as Pine, MH, and Eudora provide no specific mechanisms for expressing a privacy policy over email messages. For example, a user might send and receive email messages related to camera equipment, and wish to share this archive with other users. Since his email program provides no mechanisms for automatically sharing information, he must use the file system to accomplish sharing, subject to the underlying mechanism's limitations. In this case, the user must manually save all camera-related messages to a special folder, and use whatever file system mechanisms are available to make the contents of this folder public.[3]

In the telephony domain, both the caller and the recipient of the call have information they may choose to keep private. Caller ID is a technology that reveals the name and number of the phone used to make the call. The caller may wish to keep her phone number private and the recipient may wish to avoid revealing whether she is at home to take the call. To protect her privacy, the caller has the option of remaining anonymous on a particular call or on all calls. In response, the recipient has the option of refusing to receive any anonymous calls. Answering machines are a technology that enable a person to "screen" individual calls, see who is calling, and decide whether to answer the call or let

---

[2] The UNIX file system provides a similar command-line based interface. The `umask` command implements a dynamically-scoped default protection, in contrast with the lexically-scoped protection of NT's folder permission inheritance.

[3] This single mechanism conflates multiple objectives. A user has no way of dissociating the privacy of email messages with his organization for locating them himself. A user who ordered a used camera via email might want to keep this with his other camera messages, yet be unable to do this safely if the message contained his credit card number.

the caller leave a message. Both technologies allow people to make decisions regarding individual calls (or all calls) but fail to enable them to articulate more expressive policies based on groups of users.

The telephone system uses an extensional representation of privacy policies, and hence policies for phone calls are not compact or proactive. For example, the telephone system does not allow one to state the proactive policy "Answer all phone calls from coworkers, and direct all other calls to the answering machine." In order to enforce such a policy, a person would have to screen each call individually. Similarly, a person cannot instruct the Caller ID system to reveal his phone number under certain conditions (when calling family) but not others (when calling vendors).

FUTURE WORK

Our testbed implementation is only the beginning. We plan to gather more users so we can conduct larger studies of user acceptance. There are many directions to go from here, but we focus on two main areas: increasing the expressiveness of privacy policies, and making it more convenient to use the system.

User feedback has revealed several areas in which the privacy rule language in COLLABCLIO is not expressive enough. These areas include time-based policies and finer-grained classification.

Our simplification of policy assumes that policies are not time-dependent. However, this precludes the expression of policies that are a function of time, for example:

- Don't share any information about class grades until grades have been released at the end of the grading period.
- Hide all web pages visited during non-work hours.

In the future, we plan to investigate interfaces for allowing users to specify time-dependent policies.

We have assumed a binary classification scheme for web pages: public and private. However, there are many cases in which this is insufficient: for example, a user might want to define a group of `friends`, with whom he'll share information that is private to everyone else. This complicates visualization of the privacy policy.

Although not described in this paper, COLLABCLIO provides a small amount of support for tailoring privacy policies based on the identity of the person asking for information (the *asker*). The current system provides anonymity settings similar to those found in the Caller ID system; the asker can be anonymous when making a query, and the responder has the option of refusing anonymous queries or responding anonymously. A future extension will be to support a richer set of asker attributes, and allow a privacy policy to be conditional based on these attributes, such as the asker's privacy policy or his research group. This would allow policies such as:

- Symmetry: answer a query only if the asker would answer the same query for me.
- Share my current research results only with people in my research group.

In addition to expressiveness, it would be possible to increase the convenience of the current system. One way to accomplish this would be the use of machine learning to generalize privacy policy rules based on a set of labeled examples, or construct an intensional policy given an extensional representation. However, users might not understand how the machine learning algorithm worked, and would be less likely to trust it to learn the correct policy. A system which incorporated machine learning might allow a user to classify certain representative pages, and use the machine-generated rules as guidelines to help the user formulate his desired privacy policy.

CONCLUSION

We have introduced the notion of an explicit privacy policy object, and discussed two classes of interfaces for specifying privacy policies. We have conducted a case study of intensional and extensional interfaces in the web browsing domain, and discussed the tradeoffs between them. Our experience has led to the following conclusions:

- Privacy policies should be treated as first-class objects.
- Policy objects should be easy to create, inspect, modify and monitor; this is facilitated by a direct manipulation interface.
- Policy objects should have a compact representation and apply proactively to items not yet created; this is facilitated by an intensional representation.

ACKNOWLEDGMENTS

This research was funded in part by Office of Naval Research grants 92-J-1946 and N00014-94-1-0060, by ARPA/Rome Labs grant F30602-95-1-0024, by a gift from Rockwell International Palo Alto Research, by National Science Foundation grants IRI-9357772 and IRI-9303461, and by a National Science Foundation graduate fellowship.

REFERENCES

1. Victoria Bellotti. Design for Privacy in Multimedia Computing and Communications Environments. *Technology and Privacy: the New Landscape.* MIT Press, 1996, in publication.
2. N. J. Davies, R. Weeks, and M. C. Revett. *Information agents for the World Wide Web*, pages 81-99, Springer-Verlag, 1997.
3. Leonard N. Foner. A security architecture for multi-agent matchmaking. In *Proceedings of the Second International Conference on Multiagent Systems*, pages 80-86, Kyoto, Japan, December 1996.

4. Scott E. Hudson and Ian Smith. Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems. In *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work*, pages 248-257, Boston, MA, November 1996.
5. Peter Jackson, Han Reichgelt, and Frank van Harmelen. *Logic-Based Knowledge Representation*, page 41. MIT Press, 1989.
6. K. Kamiya, M. Roscheisen, and T. Winograd. Grassroots: a system providing a uniform framework for communicating, structuring, sharing information, and organizing people. *Computer Networks and ISDN Systems*, 28(7-11):1157-1174, May 1996.
7. Richard M. Keller, Shawn R. Wolfe, James R. Chen, Joshua L. Rabinowitz, and Nathalie Mathe. A Bookmarking Service for Organizing and Sharing URLs. In *Sixth International World Wide Web Conference*, Santa Clara, CA, April 1997.
8. P. Klark and U. Manber. Developing a personal Internet assistant. In *Proceedings of ED-MEDIA 95 - World Conference on Educational Multimedia and Hypermedia*, Graz, Australia, pages 372-377, June 1995.
9. Ben Shneiderman. Direct manipulation: a step beyond programming languages. *Computer*, 16(8):57-69, August 1983.