Tessa Lau, Oren Etzioni,
and Daniel S. Weld

Privacy Interfaces for Information Management

A system for examining Web browsing histories helps create a set of guidelines for designing privacy interfaces.

The designers of information management software must strike a delicate balance between protecting user privacy and facilitating the sharing of information. Since there is no universal policy appropriate for all users, designers must provide users with a means of specifying their own individual privacy policies. Each user then determines what information to conceal, what to reveal, and to whom. While information protection mechanisms abound, the user interface to such mechanisms has received scant attention.

In fact, current 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, in order to share files in Windows NT or Unix, one must manipulate each file and folder individually.

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 email message they receive. Moreover, users do not have the ability to proactively specify complex policies that will automatically cover messages not yet received. As a result, people default to defensive privacy policies where all potentially sensitive information such as email is kept under lock and key.

In this article, we propose a set of guidelines for designing privacy interfaces that facilitate the creation, inspection, modification, and monitoring of privacy policies. These guidelines are based on our experience with CollabClio—a system that supports automated sharing of Web browsing histories. CollabClio stores a person's browsing history and makes it searchable by content, keyword, and other attributes. A typical CollabClio query might be: "Show me all the pages Tessa has visited in the .edu domain that contain the phrase 'direct manipulation.' " Since a CollabClio user can make queries regarding the browsing history of other users, there are obvious privacy concerns.

QUENTIN WEBB

Our proposed guidelines are sufficiently broad to apply to privacy interfaces in such diverse domains as the Windows NT file system, email, and telephony. We have analyzed the privacy interfaces in each of these domains, and discuss how they could be improved in light of our design guidelines.

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 Web pages.

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 workstation and captures the user's browsing history automatically. A user can search CLIO for previosuly visited pages by describing the desired Web page in terms of attributes such as keywords in the page's content, parts of its title and URL, when it was visited, and so on. 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. Previous approaches to this problem tended to center around shared bookmark lists (for example [2, 6, 7]). 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 overcome these limitations, COLLABCLIO takes a novel approach.

To support URL sharing, each user may elect to register his or 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 CLIO to query 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 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 have 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 red light on a video camera. When the light is on, Web page visits are recorded as public; when the light is turned off, Web pages are logged 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. 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 (Figure 2), 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. A user can do this by using CLIO to search his or her history for all pages with certain attributes (for example, all pages in the .com domain). The user can then click on one or more of the retrieved pages, and mark them as either public or private by selecting an item from a menu.

An informal user survey, however, revealed that this interface was inconvenient to use for certain types of privacy policies. For example, in order to implement

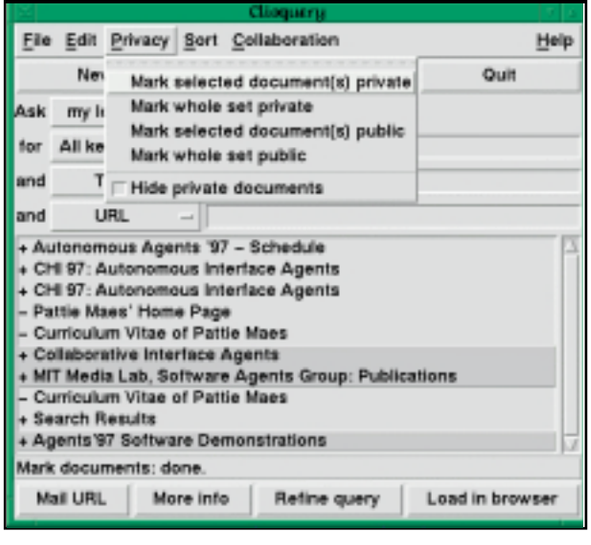
the policy "Hide all visits to Web pages in the .com domain," a user would have to remember to either toggle the record light private each time a .com site was visited, or to periodically search for all .com pages and mark them private at a later date. In the first case, the number of actions on a user's part increases with the number of sites visited. In the second case, 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. Another criticism noted 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.

Design Guidelines for Privacy Interfaces

In considering our experience, we realized the ideal privacy interface should make it easy to create, inspect, modify, and monitor privacy policies. In addition, privacy policies should be proactive—that is, apply to objects as they are encountered. Our initial design failed to support these goals to a sufficient degree.

Figure 2. The search-and-mark tool, part of the CLIO user interface. The lower half of the window displays the results of a CLIO search; titles of recently browsed Web pages are displayed; they are prefixed by their classification into public (+) or private (-); several documents have been selected and highlighted. The user is about to select a menu option that will mark these selected documents as private.



The record light interface allowed a user to create a privacy policy one document at a time. The search-and-mark tool allowed users to inspect and modify their policies (again, on a document-by-document basis). No support was provided for monitoring of policies (that is, verifying one's policy worked as it was intended under a workload of remote queries). Again, privacy policies were not proactive.

Although our first privacy interface achieved some of these goals in part, we realized the root of the problem was the fact that one's privacy policy was represented as a property of individual documents (e.g., public or private), and not as an entity in its own right.

To address this shortcoming, we introduce the distinction between *intensional* and *extensional* representations [5] of privacy policies. An extensional representation enumerates all the items in a set (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 declaratively in an intensional representation. Furthermore, such a declarative policy could be applied automatically to future Web pages as they are being visited and indexed by COLLABCLIO.

Although the search-and-mark mechanism gave the appearance of an intensional representation by retrieving sets of pages, in fact, it preserved the extensional representation used by the record light mechanism; privacy was still implemented as a property of each document. In addition, the privacy policy created using this interface was not proactive: 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.

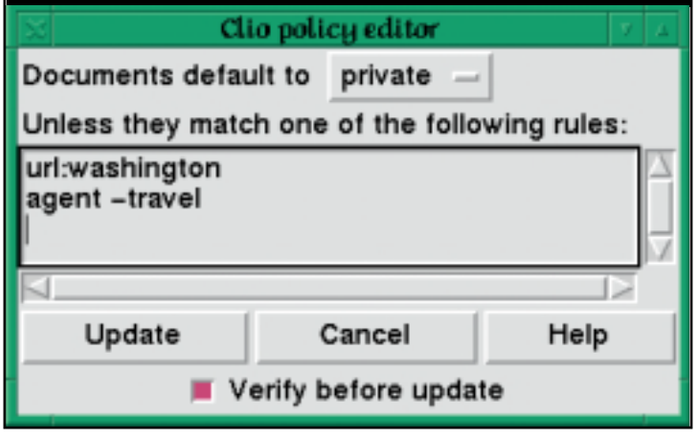
These considerations led us to develop an intensional privacy interface for COLLABCLIO.

Rule-based Privacy Interface

COLLABCLIO's second privacy interface centers around the privacy policy editor window (Figure 3). 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.

Each line in the policy represents one rule. Each rule is a list of words, using a syntax similar to that

Figure 3. The privacy policy editor interface. Clicking on the Update button puts the policy displayed in the editor window into effect. However, if "verify before update" is enabled, Clio will check for conflicts between the updated policy and existing page privacy settings; if such conflicts are detected, Clio will ask the user to verify that the changes are indeed intended.



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.

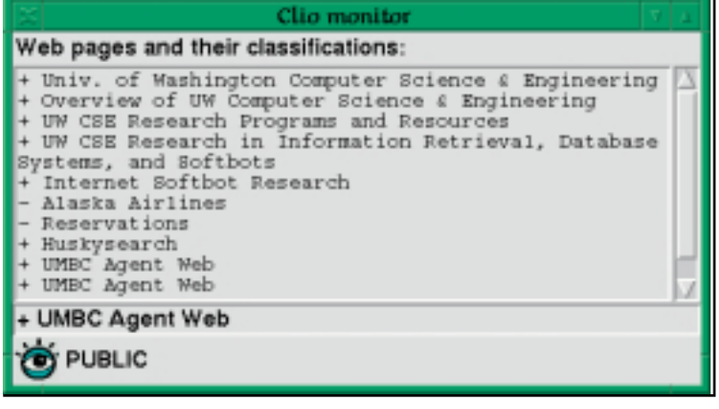
For example, the privacy policy

url:washington
agent -travel

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 4 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 the policy created in the rule-editor window is having the desired effect.

Figure 4. 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.



Privacy Interfaces in Other Systems

Our experience with COLLABCLIO led us to the general conclusion that privacy interfaces should make it easy to create, inspect, modify, and monitor privacy policies, and that proactive privacy policies can best be represented intensionally. To demonstrate that our guidelines are broadly applicable, we use them to critique the privacy interfaces in three radically 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 a set of permissions associated with it. 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.[1]

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, a privacy policy does not scale to large numbers of items. It is difficult to inspect. For example, there is no way to list which files are shared with a particular user

[1]The Unix file system provides an analogous command-line based interface for setting default protection of files (the `umask` command).

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 automatically apply to items as they are created. There is no way to express a 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 the email program provides no mechanisms for automatically sharing information, the user 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.[2]

Privacy interfaces in the telephony domain also have an all-or-none flavor that forces users to choose between making a global decision versus analyzing each item individually, but does not allow for a middle ground. Answering machines, for example, do not allow one to state the policy "pass through all phone calls from family members, and take a message from all others." Similarly, a caller cannot instruct the Caller ID system to reveal his or her phone number under certain conditions (when calling family) but not others (when calling vendors). The telephone system relies on an extensional representation of privacy policies. As a result, both caller ID and answering machines allow people to make decisions regarding individual calls (or all calls), but fail to enable them to articulate proactive and more expressive policies for groups of users.

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 colleagues, and manually enter the URL into the system each time he or she comes across something which ought to be shared.

Another area of related work concerns the issue of privacy in 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. In addition, Hudson and Smith [4] study privacy in a video-awareness system. The shadow-view technique (representing areas of recent motion in a live video feed with darkened squares) implements a proactive policy for video privacy. However, neither project suggests, as we do, the idea of an explicit privacy interface, or the encoding of privacy policies in an intensional representation.

Yenta [3] is a distributed agent system that uses email and other sources to build a profile of a user that 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.

In the commercial arena, Firefly Network, Inc. has introduced the Firefly Passport—a compact object representing an individual's privacy policy that specifies who is online, as well as other various preferences. This passport, if widely adopted, would represent an important step forward in enabling Internet users to conveniently control the information disclosed to others when online. However, the passport relies on an extensional representation of a user's privacy policy and, thus, suffers from scaling problems.

Future Work

There are many directions to go from here, but we focus on increasing the expressiveness of privacy policies, and making it more convenient for the user to formulate privacy policies.

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 policy simplification 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," or "Hide all Web pages visited during non-work hours."

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

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

We have assumed a binary classification scheme for public and private Web pages. 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 or she will share information that is private to everyone else. This complicates visualization of the privacy policy.

Although not described in this article, 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. Sample policies include:

- *Symmetry.* Answer a query only if the asker would answer the same query for me.
- *Positive information balance.* Answer a query only if the asker has shared information at least as often as she has requested it.
- *Group-based.* Share my current research results only with people in my research group.

In addition to increased expressiveness, we can help the user in formulating a privacy policy by using machine learning techniques to automatically create candidate privacy policy rules based on a set of labeled example pages provided by the user. A system that 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 the desired intensional privacy policy. Of course, extensive user studies are necessary to evaluate this idea, and COLLABCLIO as a whole.

Conclusion

We have introduced COLLABCLIO—a testbed for investigating privacy interfaces—and described design guidelines for privacy interfaces resulting from our experience with the system. We have discussed the privacy interfaces in other domains such as file systems, email, and telephony, and noted where they fail to meet our guidelines for effective privacy interfaces.

In summary, our experience has led to the following general conclusions:

- Privacy interfaces should facilitate the creation, inspection, modification, and monitoring of privacy policies.
- Privacy policies should apply automatically to objects as they are encountered.
- One way of achieving these goals is the use of an intensional representation of privacy policies.

The goal of enabling users to achieve fine-grained control of the information they reveal has received scant attention in previous work both in the user interface literature and in practice. Yet, we pay a price when information we could readily use, and that others would readily share, is hidden because it is too tedious to make decisions regarding tomorrow's personal information using yesterday's privacy interfaces. C

REFERENCES

1. Bellotti, V. Design for privacy in multimedia computing and communications environments. *Technology and Privacy: The New Landscape*. MIT Press, Cambridge, Mass., 1996.
2. Davies, N.J., Weeks, R., and Revett, M.C. Information agents for the World Wide Web. Springer-Verlag, Berlin, 1997, 81–99.
3. Foner, L.N. A security architecture for multi-agent matchmaking. In *Proceedings of the Second International Conference on Multiagent Systems.* (Kyoto, Japan, Dec. 1996).
4. Hudson, S.E. and Smith, I. Techniques for addressing fundamental privacy and disruption tradeoffs in awareness support systems. In *Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work.* (Boston, Nov. 1996).
5. Jackson, P., Reichgelt, H., and van Harmelen, F. *Logic-Based Knowledge Representation.* MIT Press, Cambridge, Mass., 1989.
6. Kamiya, K., Roscheisen, M., and Winograd, T. Grassroots: A system providing a uniform framework for communicating, structuring, sharing information, and organizing people. *Computer Networks and ISDN Systems 28,* 7 (May 1996), 1157–1174.
7. Keller, R.M., Wolfe, S.R., Chen, J.R., Rabinowitz, J.L., and Mathe, N. A bookmarking service for organizing and sharing URLs. In *Proceedings for the Sixth International World Wide Web Conference.* (Santa Clara, Calif., Apr. 1997).
8. Klark, P., and Manber, U. Developing a personal Internet assistant. In *Proceedings of ED-MEDIA 95—World Conference on Educational Multimedia and Hypermedia*. (Graz, Australia, June 1995).
9. Shneiderman, B. Direct manipulation: A step beyond programming languages. *Computer 16,* 8 (Aug. 1983).

TESSA LAU (tlau@cs.washington.edu) is a Ph.D. student in the Department of Computer Science and Engineering at the University of Washington in Seattle.

OREN ETZIONI (etzioni@cs.washington.edu) is an associate professor in the Department of Computer Science and Engineering at the University of Washington in Seattle.

DANIEL S. WELD (weld@cs.washington.edu) is a professor in the Department of Computer Science and Engineering at the University of Washington in Seattle.

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.