Private Web Search

Felipe Saint-Jean[*]
Yale University

Aaron Johnson[†]
Yale University

Dan Boneh[‡]
Stanford University

Joan Feigenbaum[§]
Yale University

ABSTRACT

Web search is currently a source of growing concern about personal privacy. It is an essential and central part of most users' activity online and therefore one through which a significant amount of personal information may be revealed. To help users protect their privacy, we have designed and implemented Private Web Search (PWS), a usable client-side tool that minimizes the information that users reveal to a search engine. Our tool protects users against attacks that involve active components and timing information, to which more general Web-browsing privacy tools (including the combination of FoxTor and Privoxy) are vulnerable. PWS is a Firefox plugin that functions as an HTTP proxy and as a client for the Tor anonymity network. It configures Firefox so that search queries executed from the PWS search box are routed through the HTTP proxy and Tor client, filtering potentially sensitive or identifying components of the request and response.

Categories and Subject Descriptors

K.4.1 [**Public Policy Issues**]: Privacy

General Terms

Security

Keywords

Web Search, Privacy, Anonymity, PWS, Tor, Firefox

[*]Supported by a Kern Family Scholarship and ARO grant W911NF-06-1-0316. Email: felipe.saint-jean@yale.edu.

[†]Supported by NSF grant 0428422 and ARO grant W911NF-05-1-0417. Email: aaron.johnson@yale.edu.

[‡]Supported by NSF and the Packard foundation. Email: dabo@cs.stanford.edu.

[§]Supported in part by NSF grants 0331548 and 0534052, ARO grant W911NF-06-1-0316, and US-Israeli BSF grant 2002065. Email: joan.feigenbaum@yale.edu.

WPES'07, October 29, 2007, Alexandria, Virginia, USA.

1. INTRODUCTION

The August 2006 release by AOL of the search queries of approximately 650,000 people [4] served as an alert to the privacy threat of online searching. Although the users' IP addresses were replaced with meaningless numbers, it was easy in many cases for a member of the general public to identify a user from his queries. The search-engine company itself has even greater power to identify users. This is worrisome, because queries can be very revealing, and the number of queries that search engines receive is growing as they improve and expand their web databases. Search-engine companies are strongly motivated to collect and analyze these data, because their business model is based on extracting user information in order to better target online advertising.

The Web-search scenario is also a good one in which to have a focused discussion about privacy. It has a few properties that are extremely important from the user's point of view, *i.e.*, with respect to actions the user can take to control what he reveals about himself. First, search services are widely used, and thus there is hope of hiding in the crowd. Second, because of the large number of users, a concrete, widely available tool for enhancing privacy might produce useful feedback. Third, it is a point of connection among most web activities; so the privacy concerns are larger than in more specific web services.

Private Web Search (PWS) is a Firefox plugin that protects the user's privacy by minimizing the information sent to the search engine, while still getting the query result. This is done by filtering the request and response and by routing them through an anonymity network. The user is thus protected from information leaks that might lead to his identification.

2. PROBLEM STATEMENT

Search-engine queries can reveal a great deal about a user. The query terms themselves can include clearly identifying information such as a user's name, Social Security Number, and location. They may also indicate things about a user's work, family, interests, and future plans. Other aspects of the search request, such as the IP address, the HTTP headers, and the time of day, also let search engines learn things about the user, such as his ISP, browser, and activity at a specific time. Clearly, many users would like to keep this kind of thing private. Privacy must, however, be achieved while still providing users with the search functionality. A trivial way to protect privacy would be to send no queries at all, but this is unacceptable.

Before we make our notion of privacy more precise, consider several scenarios in which a user's search queries are used to try to learn things about him:

1. The search-engine company runs a large-scale profiling operation in which it tries to learn as much as possible about its users. The engine could link queries and build user profiles under the assumption that queries done on the same day from the same IP address come from the same user. These profiles could be combined with information found online, such as personal web pages and government records, to learn things such as the users' names and addresses.

2. The search-engine company is more focused and monitors queries for terms of interest. These could include things like subjects of national-security interest to the government or products of partner companies. Once a term of interest is encountered, it could, as before, be linked to other queries issued around the same time and from the same IP address, as well as with online sources of information. If the terms of interest were selective enough and the interested party motivated enough, the profiles could also be compared to and combined with less available sources of information such as logs from the user's ISP or public records that must be retrieved physically.

3. The adversary wants to learn the queries of a specific user. Perhaps an employee with access to the data is curious about a celebrity, or law enforcement is gathering data in a criminal investigation. In this case, the adversary has significant background knowledge about the user to start with. The adversary might, for example, know where the user was at a certain time or what his ISP is; perhaps the adversary can guess the query terms that the user is likely to use, such as his name or something related to his work. It is easy to see how this background information can help the adversary determine the queries that were issued by the user.

In all of these situations, the privacy concern arises as the search engine becomes able to make good guesses about the source of some search queries. The engine is aided in this task by knowledge of user behavior - some that it starts with and some that it develops as it examines the queries. We don't have much hope of preventing an adversary from guessing the source of queries that are likely to come from only a few users: full names, addresses, credit-card numbers, etc. Therefore, our privacy goal will be to prevent an adversary from improving its guess too much after observing incoming queries, while still providing users with the search results that they want.

To state the privacy issue more concretely, assume that there is some set of users U and that the adversary knows the size of U. We can model web search as a probabilistic process. Let there be some probability distribution on the search queries that the users will make in a given time period. Our adversary has prior knowledge about search queries made in some period of time in the form of an estimate of the probability distribution over sets of queries. He gets some information about the queries that were performed in the form of the value of a random variable that represents his observations. From this he can infer a conditional distribution on which queries were performed and by whom. We want to minimize the difference between the prior distribution and this posterior distribution. In particular, we don't want to increase by too much the probability that a particular user issued a particular query.

We won't develop this model of the problem any further in this paper; nor will we attempt to precisely express and analyze PWS or other solutions in it. However, we will use it to understand how different approaches protect privacy. Moreover, this view of privacy illustrates how the problem of private web search relates to other privacy problems that have been studied.

There are several practical tools [7, 15, 22] that offer ways to hide major clues (*e.g.* IP address) to the user's identity. However, for the most part they do not address more subtle attacks such as Flash cookies and cache timing [6, 10]. Also, none of these tools is convenient and comprehensive. We want to provide a tool that is easy to use and is effective at protecting users' privacy during web search.

3. RELATED WORK

3.1 Current approaches

One straightforward way to protect privacy in web searching is to use an anonymizing proxy. Lists of freely accessible proxies are available online. Using these hides the true source IP address. However, because all queries are sent through the same proxy they can easily be linked together. Also, the adversary need only obtain logs from the proxy to determine the true source.

These concerns can be addressed by using the anonymity network Tor [20], which is essentially a network of anonymizing proxies. The source of the connection to the search engine is rotated periodically among the routers in Tor. Also, connections are routed through several Tor routers and encrypted in such a way that logs from all are necessary to determine the true source.

Still, the HTTP request itself might release information about the source, *e.g.*, through cookies or the User-Agent header. Also, the HTTP response might include ways to get the client to reveal itself, such as JavaScript that communicates with the engine. A filtering HTTP proxy, such as Privoxy [15], can eliminate some of these possibilities, but it is a general tool for all web browsing that does not include sufficient filtering for web-search results. In particular, the search engine can employ techniques such as redirects in the search results and cache-timing attacks [6, 10].

This solution may also be somewhat difficult for users to install and configure. Browser plugins, such as the FireFox plugins FoxTor [7] and TorButton [21], can make this easier. Even with these tools, if the user doesn't want to run all HTTP requests over the slower Tor network, he must go through the effort of manually enabling and disabling the use of Tor.

The TrackMeNot[22] tool uses a different approach. It attempts to protect the user's privacy by issuing computer-generated random queries. The objective is to make it hard for the search engine to distinguish real user queries from the computer-generated "cover traffic." Users can be identified by IP address, but what they search for is obscured to some extent by noise. TrackMeNot has not been formally analyzed, however, and it is not clear how indistinguishable

one can actually make the false queries from real ones. Real queries are often semantically related in subtle ways and may include very specific and identifying terms (*e.g.*, addresses and names). This scheme also adds undesirable extra load on the search engine.

3.2 Related privacy research

Two well known problems in the privacy literature have significant similarities to private web search: privacy-preserving data publishing [23] and private information retrieval [8].

Privacy-preserving data publishing is the problem of making a database of personal information available to the public for data mining without revealing private information about individuals. Census officials, for example, may want to provide census data so that researchers can learn general things about the population. However, they don't want to expose any individual's private information, such as his or her income. Some approaches to this problem include generalizing identifying fields [18, 13], adding random noise to the entries in the database [1, 5], and randomly adding and deleting entries [16].

Private web searching can be viewed as an instance of this problem by taking the database to be the set of search queries, the publisher to be the users, and the public to be the search engine. Solutions to privacy-preserving data publishing therefore suggest solutions to private web search. Hiding the source IP address and normalizing the HTTP headers of a web request, for example, can be viewed as an application of the generalization technique. PWS adds random noise to the response time of a query by sending it over the Tor network. Using Tor perturbs the network latency, making it harder to identify users based on their network round-trip time. This is because the network latency will depend on the randomly chosen Tor path. We are prevented by our functionality requirement from deleting queries, but adding random queries is exactly the approach taken by the TrackMeNot utility [22].

Web search differs from data publishing in several ways that affect the ability to transfer solutions between the two. First, the data in web search are being "published" by many users who are unknown to one another. We want to avoid any solution that requires coordination among the users, such as k-anonymous generalization [18]. Also, web search has a limited functionality requirement - we must obtain search results. Therefore we can freely modify any part of the request other than the search terms without being concerned that it might affect the utility of the data for data mining. Finally, because we must obtain accurate search results, we cannot in general add noise to the query terms.

In the Private Information Retrieval problem (PIR) [8], a user wants to query a database without revealing the query. It isn't hard to see that solving this problem would solve the web search problem. One simple PIR solution is for the database owner to send a copy of the database to the user. The size of the database may well be very large, however. Solutions that are information-theoretically secure and have lower communication requirements [2] involve querying copies of the database. Single-copy solutions with asymptotically low communication based on computational-hardness assumptions also exist [12].

The problem with applying PIR schemes to private web search is that search databases are huge. Replication for privacy purposes would be very costly. The single-database PIR schemes just aren't fast enough. Their response algorithms must touch every piece of the data when computing a response, or the adversary can determine that some entries were not queried.

4. ON ANONYMITY NETWORKS

When a user establishes a connection with a server and is concerned about the misuse of the personal information that the server will gather during the connection, there are a couple of approaches he can take. He can understand the server's privacy policy and trust the server to enforce it, or he can remain anonymous. Of course not all services can be accessed anonymously, but this approach should work for Web searching.

We built a client-side tool because we do not trust servers to enforce reasonable privacy policies. Indeed, a complaint in the AOL case has been lodged with the FTC arguing that AOL violated its own privacy policy [4]. That being the case, we want users to remain anonymous. A key step in realizing this is the use of anonymity networks [20, 11], and in particular the Tor onion-routing network, to obscure the source of the connection. Onion routing [17] uses a network of routers to forward messages in a way that breaks the link between incoming and outgoing traffic. This is done by layering encryption so that each router knows the previous and the next hops but nothing more. This kind of network provides practical and robust anonymity for Internet traffic and thus is very useful for our project. Tor [3] is a widely used implementation of onion routing. As of January 2007, it consists of over 800 routers and serves an estimated 200,000 users.

We are using Tor for the specific purpose of private web search, and so there may be ways to customize its operation. One that we have implemented is building 2-hop paths instead of 3-hop paths. The argument for three hops in Tor is that an adversary that controls a router should not be able to know all of the routers on the circuits it observes. Our adversary is a search-engine company, however, and we assume that it does not try to break the anonymity of the Tor network. Therefore, we can improve speed by removing one hop. We also suggest that Tor-router operators may be more willing to act as exit nodes for the popular search engines. Providing them with exit policies that allow such access could help the performance of our tool.

Tor provides an important part of our solution by hiding the source IP address. However, the search engine can get information about the source of a query in other ways.

5. SENSITIVE INFORMATION IN WEB SEARCHING AND USER TRACKING

There are many sources of potentially identifying information in web search. Table 1 show a summery of this information. First, there are IP addresses. The IP address by itself provides a large amount of information about the user. It may provide geographical location, ISP, or institution. Although associating exactly one user with an IP address is not simple, the address certainly narrows the possibilities. Because a user's IP address will, in all likelihood, be the same for a while, it provides strong linkability among queries issued during that time period. There is also timing information at the IP level. The server side of the connection will be able to estimate the round-trip time (RTT) of packets.

This will allow the adversary to distinguish between users with RTTs that are far apart.

As seen in [14], at the TCP level, inspection of packets can reveal information about the machines involved in the connection. Uptime (time since system boot), operating system, and other properties of the connection can be estimated with high accuracy by passive inspection of the network traffic.

Then, there is the HTTP header, in which there is a lot of information that allows tracking. Cookies can usually uniquely identify the user, and they provide query linkability for even longer periods of time than the IP address. Also, there is a large set of flags and markers that give the search engine specifics about the user's system. Although these are required for the processing of many types of web requests, they are not required for search.

Once the search engine gives a response, the Web browser will interpret and execute all elements in the Web page. Many of these elements are designed to provide the user with a rich experience and will thus initiate further network connections. Each additional connection may implicate privacy. For example, image, frame, and style tags will generate the download of additional files. Each of this connections must be dealt with in detail.

Beyond HTML tags, there are a variety of active components that can be embedded in the web page. These active components are used in general to improve the user's experience and enhance user interfaces, but they can also reveal private information about the user. For example, it is common practice for search engines to use JavaScript in order to get feedback about the URL selected by the user. Most of the active components execute within a constrained environment, but they are generally able to transmit to the server information specific to the user. In addition to JavaScript, we must deal with Flash, ActiveX, Java, and a variety of plugins. These active components could, in principle, be used to fingerprint the user's machine, potentially identifying the user.

The search engine can also use several web-timing attacks [6, 10] to distinguish among users. The contents of a user's web cache can be queried by sandwiching a request for some page between queries to the search engine's own content in the HTML response. By measuring the time between the requests for its own content, it can determine whether the page is in the user's cache. The contents of the user's DNS cache can be similarly queried. The search engine can potentially use this technique to read and write unique signatures in user caches.

The last piece of information transmitted when searching the Web is the search query itself. Each query gives some information about the user. In other words, a given query could be issued by a subset of the user base but probably not by all users. The problem grows as queries are linkable. Each query reduces the set of possible users that might have issued the queries, and, once the set of queries gets large enough, they can be related to a user or a small set of users. This means that, in order to achieve our main objective, we need to work towards reducing the linkability of queries to each other as well as the linkability of queries to users. It should be hard for the search engine to determine that a large set of queries was issued by the same user.

Table 1 summarizes the information that is revealed by a general Web transaction. 1 and 2 are very standard and mostly independent of the application or website. That is why 1 and 2 can be addressed by general Web-privacy tools like Tor+Privoxy. We can't expect to hide 4 without the search engine's cooperation and maintain functionality and usability. But what about 3 and 5? The information that is revealed at those levels is application-specific; thus, it is not possible to delete this information without considering the application semantics and expect to preserve functionality. This issue requires a specific solution for each Web application. PWS is a specific solution for Web search.

6. IMPLEMENTATION

6.1 General architecture

PWS is a Firefox plugin within which run a Tor client and an HTTP proxy. When the user executes a query, it connects to the HTTP proxy. The proxy filters the HTTP request, then sends it to the search engine over the Tor network. Later the proxy receives the response from Tor, filters the HTML to remove all active components, and gets the answer back to Firefox for display. Table 1 shows which PWS modules take care of the various types of information leaks that may occur during search. Right now PWS can only be used with Google [9]; it would be straightforward to extend the plugin to let users select from multiple search engines, and we intend to do so. Also, Google guesses the user's desired language from the IP address of the source, and Tor may send the query from servers around the world. Therefore we currently send queries to the English language URL (`http://www.google.com/intl/en/`). Allowing users to select a search language slightly reduces anonymity, but it an essential usability feature that we also plan to add.

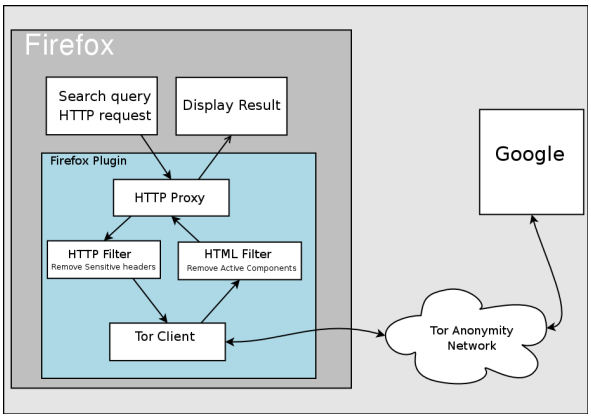

Figure 1: PWS general architecture

6.2 HTTP filter

The HTTP module's goal is to normalize the HTTP request so that it looks as similar as possible across all PWS users. Query terms will be different, of course, but all protocol specifics of the connection should be removed. A general HTTP header looks like figure 2.

Much of the information in this header is not needed to resolve the query but helps the search engine to identify users. The HTTP filter in PWS makes all headers look something like this:

```
GET /search?hl=en&q=tennis+tournament HTTP/1.1
```

	Level	Identifying information	Solution
1	TCP/IP		Tor
		IP address	
		Institution or ISP	
		Operating system	
		Uptime	
		Timing (RTT)	
2	HTTP Headers		HTTP filter
		Cookies	
		Operating system make and version	
		Browser make and version	
		Encoding and language	
3	HTML		HTML filter
		JavaScript	
		Timing (web timing attacks)	
4	Application		Open problem
		Query terms	
		Time of day of the query	
5	Active components		HTML filter
		...	
		...	
		...	

Table 1: Gray items (3 and 5) are new to PWS and provide protection not provided by Tor+Privoxy

```
GET /search?hl=en&lr=&q=tennis+tournament&btnG=Search HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.7) Gecko/20060921 \\
            Ubuntu/dapper-security Firefox/1.5.0.7
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.google.com/search?hl=en&q=tennis&btnG=Google+Search
Cookie: PREF=ID=71566fe64d9cded5:TM=1165265408:LM=1165265408: S=b7rWrEz_I8UIXl5U; \\
            SID=DQAAAGoAAABj6AMZNxlm1JiSeJcN0jZG [...]
```

Figure 2: Sample of HTTP-header section

```
Host: www.google.com
User-Agent: Mozilla/5.0
Accept-Encoding: gzip,deflate
```

The only things that change from user to user are the query terms. This is the only module that must be reimplemented to support different search engines.

6.3 Tor client

The Tor client serves two different purposes. First, it makes it hard for the search engine to link a user to a source IP address of a query. Second, it allows us to change that source IP address between queries in order to reduce query linkability. Every query is issued through a different channel and, as long as the query rate is below our channel-rebuild rate, channels are not reused. Therefore the source IP of every query is randomly and indepently selected from the routers in the Tor network (or, more accurately, from the exit nodes).

Our implementation of a Tor client makes heavy use of Jap [11]. The Jap proxy is not solely a Tor client, but it implements one as part of its feature set. The Jap Tor client is written in Java and embedded in the PWS plugin using the Java Firefox Extension code from the SIMILE [19] project. We use paths of length 2, instead of the default 3, in order to improve performance. Still, using PWS is much slower than a direct search, because queries are routed through the Tor network. In our tests, a query was, on average, 20 times slower than a direct connection: 10 seconds vs 0.5 seconds. The variance in response time was much higher, too.

6.4 HTML filter

The HTML filter's job is to remove any component that may provide feedback to the search engine. Recall that this is the additional privacy protection that PWS provides over and above the protection provided by Tor+Provoxy. This is done by parsing the response HTML and extracting only the information needed to present an answer to the user. The information extracted is the result description, text abstract, and result URL. These are extracted using regular expressions. Using the extracted text, a new HTML file is built, with all HTML generated by PWS. This has only the results and no embedded objects. This means that the user only performs one HTTP GET per query, preventing cache timing attacks [6]. All active components such as JavaScript and Flash are removed, so that no extra code is executed.

One alternative approach might be to disable all features in the browser that can initiate additional connections and leak information. We prefer our approach for two reasons. First, it gives better assurance that all active components and undesired labels are removed, even those we did not think about while writing PWS. Second, it gives us control over the functionality. It is possible that disabling some feature will break functionality. By generating new HTML, we can be sure that functionality is preserved.

Because we are protecting only connections to the search engine, we need to be much more careful about additional connections than other tools that filter all connections. Just loading an image is very dangerous if the resulting connection is not filtered. The HTML filter makes sure this does not happens.

Changes in the search engine's HTML code can make the HTML filter fail to produce an output. That means that we need to keep up with the changes. We don't expect the HTML in the response to change too frequently. Also, updating Firefox plugins is a painless process for the user.

7. HOW TO USE PWS

Usability is an important objective of ours, and so in this section we describe briefly how the plugin works from the user's point of view.

The PWS plugin is distributed as a single .xpi file that, to be installed, needs to be dragged into the plugin installation window. Once it's installed, Firefox will need to be restarted. After that, a new search option will show up among the search engines in the search box (see figure 3). That search option is labeled "PWS Google search." When the user searches using this option, the query will be routed through the HTTP proxy and the Tor client. The response will take some more time and will look different from the standard Google response, because the HTML was filtered and then displayed. Notice that the user can choose between regular Google queries and PWS just by selecting accordingly from the search box.

Figure 3: PWS user interface

8. CONCLUSIONS AND FUTURE WORK

Taken together, search queries can reveal a lot of information about a user. Experience has shown that we cannot rely on the search-engine companies not to release this information or otherwise use it in undesirable ways. Therefore, we suggest that users might want to minimize the information they give to the search engines by preventing them from identifying the source of the queries or even which queries originate from the same user.

Preventing search engines from linking queries requires eliminating all potentially unique information that gets sent to them. We have identified the IP layer, the HTTP request, and the browser's behavior on the HTML response as likely sources of this information and have designed a tool that prevents all of these sources from leaking. Previous tools deal only with part of the potentially identifying information; so PWS is a contribution over exiting software.

Our tool, Private Web Search (PWS), is a Firefox plugin that gives users a search box in the browser for private search. Queries are sent over the Tor anonymity network, after homogenizing the HTTP headers, and the results are extracted from the response to prevent active components or timing attacks from identifying the user to the engine. The plugin is easy to install and use and works transparently in the background after a request is made, with an increase in the response time as the only major change in the user experience.

There is a very interesting side impact of this project. Search-engine companies, as seen with AOL, want to release data to third parties in order to improve their understanding of web search. The problem is that there is no agreed-upon privacy standard for releasing these data. If we are able to reach agreement that our tool protects individual users' privacy, that by itself will advance the discussion about data-release policies. The search-engine company could simulate the case in which every user uses PWS and thus generate a dataset acceptable for distribution. This may, however, reduce the usefulness of the data for research purposes.

No usability tests have been made so far, but, because we regard usability as an important feature of PWS, we will conduct them as the application matures.

9. REFERENCES

[1] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In *Proceedings of the 2nd Theory of Cryptography Conference*, pages 363–385, February 2005.

[2] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 41–50, October 1995.

[3] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004. `http://tor.eff.org/tor-design.pdf`.

[4] Eff. `http://www.eff.org/Privacy/AOL/`.

[5] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM Symposium on Principles of Database Systems*, pages 211–222, June 2003.

[6] Edward Felten and Michael Schneider. Timing attacks on web privacy. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 25–32, November 2000.

[7] Foxtor. `http://cups.cs.cmu.edu/foxtor/`.

[8] William Gasarch. A survey on private information retrieval, 2004.

[9] Google. `http://www.google.com/`.

[10] Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell. Protecting browser state from web privacy attacks. In *Proceedings of the 15th International Conference on the World Wide Web*, pages 737–744, May 2006.

[11] Jap. `http://anon.inf.tu-dresden.de/index_en.html`.

[12] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 364–373, October 1997.

[13] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd IEEE International Conference on Data Engineering*, page 24, April 2006.

[14] p0f. `http://lcamtuf.coredump.cx/p0f/README`.

[15] Privoxy. `http://www.privoxy.org`.

[16] Vibhor Rastogi, Dan Suciu, and Sungho Hong. The boundary between privacy and utility in data anonymization, 2006. `http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0612103`.

[17] Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494, May 1998.

[18] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory, Palo Alto, CA, 1998.

[19] Simile. `http://simile.mit.edu/java-firefox-extension/`.

[20] Tor. `http://tor.eff.org`.

[21] Torbutton. `http://freehaven.net/~squires/torbutton/`.

[22] Trackmenot. `http://mrl.nyu.edu/~dhowe/TrackMeNot/`.

[23] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 3(1):50–57, March 2004.