Adnostic: Privacy Preserving Targeted Advertising*

Vincent Toubiana vincent.toubiana@nyu.edu
Arvind Narayanan relax@stanford.edu
Dan Boneh dabo@cs.stanford.edu
Helen Nissenbaum hfn1@nyu.edu
Solon Barocas solon@nyu.edu

Abstract

Online behavioral advertising (OBA) refers to the practice of tracking users across web sites in order to infer user interests and preferences. These interests and preferences are then used for selecting ads to present to the user. There is great concern that behavioral advertising in its present form infringes on user privacy. The resulting public debate — which includes consumer advocacy organizations, professional associations, and government agencies — is premised on the notion that OBA and privacy are inherently in conflict.

In this paper we propose a practical architecture that enables targeting without compromising user privacy. Behavioral profiling and targeting in our system takes place in the user's browser. We discuss the effectiveness of the system as well as potential social engineering and web-based attacks on the architecture. One complication is billing; ad-networks must bill the correct advertiser without knowing which ad was displayed to the user. We propose an efficient cryptographic billing system that directly solves the problem. We implemented the core targeting system as a Firefox extension and report on its effectiveness.

1 Introduction

Online behavioral advertising (OBA), also called online behavioral ad targeting, is not a new practice. In the late 1990s, DoubleClick used 3rd party cookies to track users across sites and presented ads based on the user's browsing patterns. Today OBA is a large and complex industry where entire exchanges trade user information specifically for the purpose of better ad placement. Even Google announced[1] in March 2009 that its AdSense service will begin targeting ads based on the user's browsing behavior [4]. In the next section we survey some of the key players in this space.

Although behavioral advertising promises an improvement over more scatter-shot approaches, it has evoked great concern and protest, particularly among privacy and consumer advocacy groups. Some are concerned that OBA is manipulative and discriminatory, but the dominant concern is its implications for privacy. As early as February 2000, the Electronic Privacy Information Center filed a complaint on this practice with the Federal Trade Commission (FTC) and in 2002 a settlement was reached in which DoubleClick agreed to expire its cookies after 5 years and to post a clear notice on sites where it collects user information [14].

More recently, in 2007 the FTC undertook an ongoing examination of OBA and in February 2009 issued a revised report and recommendations [11]. Viewing online advertising as a key reason for free content on the web, the FTC allows the continued practice of OBA, but with various safeguards built into it, such as greater transparency, provision for consumer opt-out, and special handling of so-called sensitive information such as health and financial information.

Last year, out of similar concerns, Congress began a study of deep packet inspection at Internet Service Providers (ISPs). Deep packet inspection is a related practice developed by companies such as Nebuad and Phorm, enabling ISPs to track user behavior and sell the information. In April 2009 a house subcommittee began looking into whether the practice needs to be outlawed [33].

Our work. While the policy debate on OBA is heating up, the discussion is framed around the premise that *behavioral targeting is inherently in conflict with privacy*. That is, if privacy is to be preserved then OBA is impossible. Similarly, if behavioral targeting takes place then user privacy is at risk. In this paper we present a way around the dilemma, arguing that other alternatives exist. We argue, in particular, that it is possible to support targeted advertising without compromising user privacy and propose an architecture to do so.

*This work was supported by the NSF PORTIA and MURI PRESIDIO projects.

[1]The following quote from [4] is illustrative: "if users visit a number of sports pages, we'll add them to the *sports enthusiast* interest category."

At a high level, our system called *Adnostic* is implemented as a browser extension that runs the behavioral targeting algorithm on the user's browser by processing the browser's history database. The results are kept in the browser and are used to select ads to display on the page. If the user does not click on the ad the selection is never communicated outside the browser and consequently no information about the user's behavior is leaked. Users see ads relevant to their interests, but user information is not leaked to the outside world (i.e. outside the browser). We use cryptographic techniques to ensure accurate accounting between advertisers, publishers, and ad-networks without compromising user privacy.

The motivation behind *Adnostic* is not to replace the existing behavioral advertising infrastructure, but rather to complement it. Ad-networks can continue to provide a user-tracking based advertising service alongside a privacy-preserving service based on *Adnostic*. Web sites who wish to display ads can choose which advertising service they wish to use. We discuss this in more detail in Section 2.3.

Our architecture is described in Sections 3, 4 and 5. In Section 6 we discuss a preliminary prototype implementing the architecture. While our proposed architecture is designed to minimize disruption to the Web ecosystem, many other designs are possible. We survey a few alternatives in Section 7. We see this as a fascinating research area with many opportunities for future work. We survey related work in Section 8.

2 A brief overview of online advertising

The online advertising world has become incredibly complex with many involved parties playing different roles [3]. We give a brief overview to help explain our proposed architecture. We start with the basic terminology:

- **Advertiser:** a party that has an online ad it wants to embed in web pages across the web. The advertiser is willing to pay for this service.
- **Publisher:** a party who owns a web page (or web site) and is willing to place ads from others on its pages. The publisher expects to be payed for this service. We will use the NY-Times as our canonical publisher example.
- **Ad-network:** a party who collects ads (and payment) from advertisers and places them on publisher pages (along with paying the publisher). Example ad-networks include Google, Yahoo!, MSN, AOL, and AdBrite (listed in order of number of ads served in 2008 [6]).

Publishers who wish to work with a particular ad-network embed on their pages a link to the ad-network (in the form of an iframe, an image, a "web bug," or a one pixel transparent image whose sole purpose is user tracking). When a user views the page, the user's browser contacts the ad-network's servers, which enables the ad-network to track the user across all publishers that partner with them. Google AdSense, for example, can track users across 91,000 unique domains [6]. The ad-network then uses its targeting algorithm to decide which ad to present on the publisher's page. Ad targeting takes many variables into account: the advertiser's bid and budget for presenting ads, the content of the page on which the ad is to be placed (called *content-based* or *contextual* targeting), and the user viewing the ad (called behavioral targeting).

Tracking. Tracking requires the ability to associate an identifier with a user across publishers and across different visits. There are many ways to achieve this: third-party HTTP cookies are the most common approach. Cookies can be cleared from browser memory, and indeed, according to one study, 52% of users deleted their cookies at least once in 2004 [26]. In response, a variety of harder-to-detect methods are being used [38], with Flash cookies being the most popular, since most users lack awareness and tools to manage them. IP address is another tracking technique, albeit somewhat coarser-grained than cookies. User-agent strings are also used, especially in conjunction with other identifiers such as IP address. Browsers generally include many types of information in the user-agent string, such as browser name and version, OS name and version, any browser extensions installed, etc. The UA tracker project lists over 24,000 unique user-agent strings [30]. For more information on tracking, we refer the reader to a case study of user tracking techniques used on levis.com [15].

2.1 Existing behavioral targeting systems

In recent years many players have entered the behavioral targeting space. As mentioned above, Google AdSense began using behavioral targeting in March 2009. Following the FTC recommendations for transparency, Google alerted all their publishers to update their privacy policy to reflect this fact. Google and Yahoo! provide a user-profile page where Internet users can set their interests or opt-out of behavioral targeting altogether [20, 45], consistent with the FTC recommendations.

Many other players help advertisers with behavioral advertising [7, 18, 9, 16, 32, 34] and all provide an opt-out mechanism by which Internet users can ask to be removed from their database. Users, however, have to actively opt-out from each provider separately. This greatly limits the effectiveness of the opt-out mechanism since concerned users cannot be expected to manually opt-out of every tracking service on the web. A centralized opt-out service (similar

to the FTC's do not call list) would be helpful, but does not currently exist.

Several OBA providers [9, 16, 32, 34] specialize in user tracking and simply sell the information to advertisers and ad-networks. Some [9, 16] do so by paying publishers to link to their site. Others [32, 34] track users by contracting with Internet Service Providers (ISPs).

Some OBA providers [7, 18] specialize in *retargeting*: when a user visits an online merchant A the information is recorded and ads for A are presented to the user when visiting other publishers.

Most behavioral targeting systems work by categorizing users into one or more *audience segments* usually based on the data collected about their browsing behavior but possibly on search terms [45] or even placement in a social network [35] . The advertiser specifies the segments(s) that they would like their ad to target, and the ad-network considers this factor in addition to the advertiser's bid and the content of the page. The AOL advertising platform uses 70 audience segments listed at [5]. AudienceScience presents their list of 98 audience segments at [8]. The Google Ad Network uses a hierarchical categorization with 27 top-level categories [20], listed in the appendix.

2.2 Privacy and threat model

Privacy in behavioral targeting is a complex issue because there are many parties involved and many types of user data to be protected. Let us now examine them in detail and motivate the choices made by *Adnostic*.

Recall that there are 3 main parties involved in online advertising: the advertiser, the publisher, and the ad-network. In addition, there are two other parties that are relevant from a privacy perspective.

- **Content Distribution Network** or CDN: an entity such as Limelight or Akamai which provides a hosting service that the larger ad-networks use to actually serve the ads. If we assume that the CDN may "collude" with the ad-network, we may as well consider the two to be the same party. This assumption is examined in more detail in Section 7.

- **Trusted third party**[2]: a provider of a (typically cryptographic) service trusted not to collude with the ad-network or any of the other parties. Note, however, that it is not acceptable to leak any of the user's data to this entity. Ideally, this service should only be contacted infrequently.

Next, let us consider the types of data that the user may wish to keep private.

[2]Although in this instance it may be more appropriately be termed a trusted sixth party, counting the user and the four parties already introduced.

- **Clickstream**: a list of all URLs visited by the user (across all publishers who partner with a given ad-network).

- **Behavioral profile**: a short description of the user's interests inferred from the clickstream. Behavioral profiling in *Adnostic* which happens in the browser, will be explained in detail in Section 6.1. In current practice, the ad-network computes the user's behavioral profile from the clickstream in order to target ads.

- **Ad impression history**: a list of all ads that have been displayed to the user.

- **Ad click history**: a list of all ads that the user has clicked on.

Our goal is to enable ad-networks to provide the most relevant ads without collecting information that the user may wish to keep private. In *Adnostic* the ad-network provides targeted ads with no knowledge of the user's clickstream, behavioral profile, or ad impression history.

Adnostic does not hide the ad click history from the ad-network for two reasons. First, ad-networks need the ad click history to defend against click fraud scams [13]. Second, advertisers see all clicks on their ads and they have a strong incentive to share that information with the ad-network to provide better placement for their ads. Hence, even if we hide ad clicks from the ad-network, it will likely obtain the information anyhow from the advertisers. For these reasons current systems redirect clicks on ads through the ad-network's systems. *Adnostic* does not change this practice.

Other private targeting architecture may choose to keep more or keep less information hidden from the ad-network. We discuss a number of alternatives and tradeoffs in Section 7.

2.3 Incentives for deploying Adnostic

Behavioral targeting with *Adnostic* is more limited than behavioral advertising when user privacy is not a concern. While *Adnostic* requires no extra effort from publishers and advertisers, ad-networks will need to modify the way they serve ads in order to support *Adnostic*. The success of *Adnostic* is therefore crucially dependent on the incentives for ad-networks to implement and support for it. We discuss numerous reasons why the interests of ad-networks and other parties are well served by deploying an *Adnostic*-like system alongside other services (possibly non-private) it may offer.

Privacy-conscious publishers. First, ad-networks will most likely offer *Adnostic* as a service that complements

rather than replaces the existing OBA mechanism. Publishers can then decide whether they want to link to the ad-network's user-tracking system or privacy-preserving system. Ad-networks could reward publishers for linking to tracking-based systems by paying more for these links. But the privacy-preserving option might bring to the table sites who, out of concern for user privacy might otherwise not choose to serve ads at all.

Providing a privacy-preserving option would enable sites who care about user privacy to serve ads. For example, sites like Alcoholics Anonymous (`aa.org`) currently serve no ads. If *Adnostic* is available, they could use it to serve ads without violating their privacy policy.

Lower barrier to entry. Second, with *Adnostic*, user behavior is analyzed at the browser which has complete visibility into the user's browsing pattern. A small ad-network, in contrast, only sees the user's visited pages when the user visits a site that partners with the ad-network. With *Adnostic* a small ad-network will likely do much better at placing relevant ads for the user.

Regulatory compliance. Third, the FTC recommends that all ad-networks provide the user with an opt-out mechanism that disables user tracking. For users who opt out, the ad-network could use *Adnostic*. This still allows the ad-network to target ads without tracking user actions.

Potentially improved user tracking. Cookie and IP-based systems are unable to track users across different computers. Many users frequently change computers or terminals, re-install their operating system, clear their cookies, etc. A browser-based system has the potential to enable near-perfect user tracking, since many browser vendors (including Mozilla Firefox and Opera) offer tools to keep profiles across multiple installations synchronized [29]. Third-party plugins offering such functionality are available for all major browsers [41]. The ad-network might even choose to bundle a browser synchronizer with an *Adnostic*-like system.

The converse problem is to disambiguate different individuals who might use the same browser session. In Section 3.1.3, we discuss how *Adnostic* might be able to solve this problem better than current OBA systems.

Targeting in private browsing mode. All modern browsers support a mode in which no information about browsing activity (including cookies) is stored beyond the session. This is called *Private Browsing* in Firefox, *In-Private Browsing* in Internet Explorer, and *Incognito mode* in Google Chrome. When this mode is enabled, the user session cannot be tied to their previous activity, and therefore behavioral targeting in its current form is difficult. On the other hand, *Adnostic* continues to operate even while browsing privately. This is acceptable because while *Adnostic* serves ads in this mode, it does not update the user's behavioral profile. Thus *Adnostic* can extend the reach of behavioral targeting to private browsing modes.

User control via centralized interface. There is one major benefit of *Adnostic* for *users* in addition to privacy: users have the ability to control the selection of ads that they are served, and furthermore, they have a centralized interface for doing so, by editing the behavioral profile computed by *Adnostic*. Currently, users need to modify their behavioral profile separately on each ad-network, of which there are over 90 [39].

Standardized audience segmentation. Different ad-networks use different sets of audience segments or categories. This creates confusion for advertisers, limiting the ability to migrate to a different ad-network, hurting the industry as a whole. A standardization initiative by Tacoda a few years ago to did not succeed [28]. With *Adnostic*, a single categorization system is built into the client, and any ad-network adopting it would be able to present a uniform interface to the advertisers.

3 Adnostic architecture: Targeting with privacy

As discussed above, OBA holds the promise of more effective advertising while presenting users with ads that are more relevant to their interests (rather than, say, an endless stream of ads for cheap mortgages.) Accepting these propositions, we suggest that OBA would be more broadly accepted if it could be conducted without compromising user privacy. Accordingly, we propose a privacy-preserving architecture for OBA. Currently implemented as a browser extension, the system consists of the following components.

3.1 Behavioral profiling

3.1.1 Interest categorization

Our proposed system relies on the browser to locally process the history database to determine user interests. As the user browses more web pages, the interest categorization is continually updated and refined. One method of accomplishing this is for the extension to classify the topic of each viewed page by using natural language processing heuristics running in the browser.

Another method is to rely on an external list of URLs, but without compromising user privacy. The browser downloads from the ad-network (or another third party) a fixed list of URLs and their classification. The list can be updated periodically, but it is the same list for all users. For every record in the history database the browser consults the list to determine the record's classification. Pages not on the list are processed locally using natural language heuristics. For

a quantitative analysis of the feasibility of the two methods, see Section 6.3.

In principle web sites can categorize their own content and provide the categorization to browsers in an HTTP header as part of the response to an HTTP request. This will help the browser build a more accurate user profile. However, since there is little incentive for web sites to provide this service we do not rely on this approach.

3.1.2 Other aspects of behavior

In current OBA systems, *behavior* denotes more than *interest.* Specifically, the "audience segments" used by current OBA systems measure attributes such as *intent* and *influence*.

Intent refers to *intent to purchase*, and distinguishes "auto enthusiasts" from "auto shoppers," which are separate audience segments targeted by AudienceScience [8]. One possible way to measure intent is to observe user actions such as requesting an auto quote or adding an item to a shopping cart. Influence refers to influence over the purchasing habits of others; one possible way to measure it is to detect users who spend a disproportionate amount of time consuming the latest news and trends in their category of interest.

These nuanced attributes can certainly be measured by observing the user's browsing activity locally. In addition, a browser-based system has the potential ability to tie into other streams of user activity such as social networking services. However, it is not clear if the level of profiling/targeting accuracy can be as high as the current server-side tracking approach.

3.1.3 User Sessions

Finally, there is the issue of *session disambiguation*: what happens when more than one person uses the computer (using a single account)? Current OBA systems have difficulties with this; consequently, children who use their parents' computer may be presented with mortgage refinancing ads based on their parents' profile(s).

Since our system runs in the browser, it can do a better job of identifying the current user interacting with the system. One option is to use keystroke dynamics, which vary from user to user [31]. Another option is to identify the current user based on the immediate last few pages viewed. That is, if the user visits a number of child-related web sites within a short period of time, it is likely that the current user is a child rather than an adult.

3.2 Ad insertion

The process consists of a number of steps. When the user visits a publisher page (e.g. the NY-Times), the publisher sends back a page that points to the ad-network, as usual.

The content served by the ad-network first uses Javascript to check if the user has *Adnostic* installed (details of our proposed implementation are in Section 6.4). If *Adnostic* is not detected, then the ad-network loads a specific ad to be placed on the page. On the other hand, if *Adnostic* is detected, the ad-network instead sends back a list of n ads (say $n = 20$) that its existing pricing and content-based algorithms decide are most appropriate for the page. Each ad is tagged with a classification indicating its topic.

Our browser extension now uses the information obtained from processing the browser's history database to choose one of the n ads for the page, namely the ad that is most relevant to the user's interest. That ad is displayed as part of the page. In Section 6.4 we discuss our proposed implementation of the ad specification API and ad selection algorithm.

The choice of n is configurable. A larger value allows more precise targeting, but increases network bandwidth. A smaller value implies more coarse targeting, but reduces network bandwidth. In the next section we discuss what a reasonable value for n might be.

Note that ads are often served from a content distribution network (CDN) which has servers close to the user. Consequently, ad loads are typically faster than loading dynamic content from the enclosing page. Moreover, loading the ads need not slow down the rendering of the enclosing page. By loading ads *after* the enclosing page is rendered we minimize the impact on the user experience. Ads will appear a few milliseconds after the enclosing page is displayed.

3.3 Accounting

The final component is accounting. Accounting in the "charge per click" model remains unchanged: it is not a privacy violation for the ad-network to infer that a user clicked on an ad, and therefore the standard accounting procedures can be applied.

In the "charge per impression" model, the correct advertiser must be billed without the ad-network learning which ad was displayed to the user. In Section 5 we propose a solution to this problem based on homomorphic encryption. The key idea is that the ad-network can maintain encrypted impression counters throughout a billing cycle, and rely on a trusted third party to decrypt these counters at the end of the cycle.

4 Possible objections

A number of questions come to mind when first considering this proposal.

Network latency and bandwidth. We already argued that loading multiple ads can happen after the enclosing page is rendered, thus minimizing the impact on the user experience. Another option is for the *Adnostic* extension to prefetch ads before the user visits the publisher's page. Ad prefetching, however, does not fit well with current real-time ad auction systems and we do not include it in our system. Nevertheless, an arbitrary number of ads could be prefetched in the browser and stored locally. When a user visits a web page, the browser receives the list of n ads and compares it to the list of prefetched ads. If the ad that it chooses to display is already prefetched, it displays it immediately, thus speeding up page display. But, whether the selected ad has been prefetched or not, the extension downloads the listed ads that are not already prefetched to avoid information leakage to the ad-network.

Another issue is network bandwidth. While the big ad-networks serve ads from their own content distribution networks (CDNs), the smaller ones buy services from existing CDNs. Consequently, the extra traffic generated by *Adnostic* could have a non-trivial financial impact on the smaller ad-networks. One way to remedy this is to assume that the CDN is not colluding with the ad-network. The browser could then receive the list of ads from the ad-network, but only request from the CDN the ad that will be displayed to the user. This will greatly reduce the network bandwidth and latency of running *Adnostic* while keeping the ad-network in the dark as to what ad was displayed. The CDN could be contractually obligated to not share the choice of ad with the ad-network. Note that the CDN itself can determine the ads served to the user, but it has no financial incentive for doing so, given that it is prohibited from selling this data back to the ad-network. This mechanism is optional, but it may serve as a reasonable compromise between privacy and operational constraints.

Effectiveness of this form of behavioral advertising. Since the ad-network only sends a small number of ads to the browser, there is a risk that none of the transmitted ads match the precise user interest. For a given interest-segmentation system, the ad-network needs to send only one ad per segment to cover the spectrum of interests. Our survey of existing OBA systems (Section 2.1) shows that current systems use between 25 and 100 segments, which gives an upper bound on the number of ads sent. If we further assume that most users have two or more interests, the number can be brought down further while still retaining a high probability of matching a user interest.

Another possible drawback in terms of effectiveness is the accuracy of classifying user behavior in a browser-based model. In Section 3.1.2 we discussed how browser-based behavioral profiling can capture most of the nuances that are employed today. However, our current implementation treats behavior as synonymous with interest, and further research is necessary to establish that browser-based behavioral profiling can be sufficiently accurate.

Enforcement of non-tracking. In principle, an ad-network could offer an *Adnostic*-like service, but continue to track users as it does today. There is no fool-proof technical way to prevent tracking, short of relying on inconvenient anonymity services like Tor. In reality, we expect that enforcement of non-tracking will be done contractually, since tracking is easy to detect by manual inspection. Potentially, the *Adnostic* browser extension can include a user setting that uses heuristics (such as those used by AdBlock Plus [1] and RequestPolicy [36]) to disable tracking to the extent possible.

Ad blocking. While add-ons that block online ads – primarily Adblock Plus – have vocal users and advocates, these users are a tiny minority: under 10 million as of this writing [2], out of around 300 million users of Firefox, according to Mozilla [43]. Perhaps one reason that ad blocking is not more widely used is that users find benefit in viewing ads. *Adnostic* can cater to this pragmatic majority without compromising privacy.

Click fraud. In our architecture the initial interaction with the ad-network is unchanged. Hence, online click-fraud detection mechanisms should continue to work unchanged. This is critical to our design. We explain in Section 5 that offline methods (i.e. methods that credit an advertiser in case a click is later deemed fraudulent) also work.

4.1 Malice

Can *Adnostic* be used by malicious parties to infer user interests? We consider two attacks and explain how we address them.

Fraudulent clicks with active content. A malicious ad-network who wishes to learn user interests could provide ads to the browser and then use Javascript in the enclosing page to click on whichever ad was presented to the user. The resulting HTTP request sent to the ad-network would reveal information about the user's interests. This is not limited to ad-networks; any site the user visits could try this.

This attack is mitigated by standard click-fraud defenses. The browser renders the selected ad in an iframe with a distinct origin. By the same origin policy, the enclosing page has no visibility into the inner structure of the iframe and is prevented from clicking on elements in the iframe. While this works well for text and image ads, ads that contain active content (such as Flash ads) are a problem since the ad

could click on itself. As a result, *Adnostic* may need to block ads containing active content. Similarly, clickjacking [22] is a concern; it will need to be addressed by whatever browser mechanism is eventually adopted to prevent clickjacking.

Despite the same origin policy, the enclosing page has access to the height and width of the iframe. Therefore, it is essential that the iframe dimensions are independent of the ad being presented in the iframe.

Social engineering attacks. A malicious advertiser who wishes to learn user interests could provide an ad whose topic tag is "wrist watches," but the ad itself says "click here to get five dollars." A user who is fooled into clicking on the ad reveals to the advertiser that he is interested in wrist watches. Of course the user does not get the five dollars.

There are a number of defenses against this. One option is to rely on ad filtering at the ad-network. Malicious ads are quickly becoming a significant concern for ad-networks [40]. Microsoft, for example, recently filed suite against five ad-networks whose ad inventory happened to include malicious ads. To combat malicious ads, many ad-networks apply a filtering process before accepting an ad. The social engineering ads discussed in the previous paragraph are likely to be identified by this process and removed. Similarly, these ads can be quickly removed in response to user complaints.

Another potential defense is to add random noise. With some low probability, *Adnostic* could present to the user ads that are unrelated to his interests. As a result, the malicious advertiser cannot be certain that the victim is interested in wrist watches.

5 Billing and accounting

Accounting in the "charge per click" model remains unchanged: once users click on an ad they are directed to the advertiser's site, possibly through the ad-network, where accounting takes place.

Accounting in the "charge per impression" model is more challenging. Although n ads are pushed to the browser, only one ad is presented to the user and therefore only one of the n advertisers should be charged for the ad. We need to address the question of how the ad-network can charge the correct advertiser without knowing which ad was displayed.

We describe a cryptographic solution based on an additively homomorphic encryption and efficient zero-knowledge proofs. Recall that in a homomorphic encryption system, anyone given the public key pk and ciphertexts $E(\mathsf{pk}, x_1)$ and $E(\mathsf{pk}, x_2)$, can create the ciphertext $E(\mathsf{pk}, x_1 + x_2)$ without knowledge of x_1 or x_2. Similarly, given pk and $E(\mathsf{pk}, x)$, anyone can create $E(\mathsf{pk}, c \cdot x)$ for any scalar c. Many additively homomorphic encryption systems exist; we will use a system based on ElGamal encryption on elliptic curves, as it is conceptually simplest. In this version of ElGamal, to encrypt a small integer $m \in \{0, \ldots, b\}$ we encode m as a group element by computing $\hat{m} \leftarrow mP$ for some generator P and then encrypt $\hat{m}$ with the ElGamal system. The resulting system is additively homomorphic. Since messages are small, decryption is done by first doing ElGamal decryption to obtain $\hat{m}$ and then computing discrete-log of $\hat{m}$ to obtain m.

We assume that each ad at the ad-network is identified by an ad ID. Along with each ad, the ad-network will store in its database an encrypted counter, denoted by C_{ID}. The encrypted counter is managed as follows.

Initialization. At the beginning of the billing period — say at the beginning of the month or when the ad is first uploaded to the ad-network — the encrypted counter is initialized to zero. That is, the ad-network sets $C_{\mathsf{ID}} \leftarrow E(\mathsf{pk}, 0)$ for some public key pk which will be discussed later.

Ad insertion. When the *Adnostic* protocol runs, the ad-network chooses n ads to send to the browser. It remembers the IDs of the ads chosen and sends these n ads to the browser as an ordered set. It also sends the public key pk to the browser. Hence, the data sent to the browser consists of

$$\big(\mathsf{pk},\ \mathrm{ad}_1,\ \mathrm{ad}_2,\ \ldots, \mathrm{ad}_n\big)$$

The browser chooses an ad to display to the user. It then creates a binary vector $\bar{v} = (v_1, \ldots, v_n) \in \{0,1\}^n$ that represents which ad was presented to the user. That is, the vector $\bar{v}$ has n components where all are 0 except for the one component corresponding to the displayed ad which is set to 1. For example, if ad number 2 was presented to the user then the vector would be $(0, 1, 0, 0, \ldots, 0)$.

Next, the browser encrypts each component of $\bar{v}$ using pk to obtain a vector of ciphertexts

$$\big(E(\mathsf{pk}, v_1),\ \ldots,\ E(\mathsf{pk}, v_n)\big).$$

It sends to the ad-network this vector along with zero-knowledge proofs of the following facts: first, for $i = 1, \ldots, n$ the quantity v_i is either 0 or 1, and second, the sum $\sum_{i=1}^{n} v_i$ is 1. More precisely, the zero-knowledge proofs show that for all $i = 1, \ldots, n$ the ciphertext $E(\mathsf{pk}, v_i)$ is an encryption of 0 or 1 and that the ciphertext $E(\mathsf{pk}, \sum_{i=1}^{n} v_i)$ is an encryption of 1. Note that the ad-network can use the homomorphic property to build $E(\mathsf{pk}, \sum_{i=1}^{n} v_i)$ itself from the given vector of ciphertexts.

The proofs convince the ad-network that the vector $\bar{v}$ is a zero vector with only one non-zero element equal to 1. Similar zero-knowledge proofs are needed for homomorphic voting schemes and an example zero-knowledge protocol for this task is provided in Sections 2.6,3,4 of [12].

For a standard security level (e.g. 256-bit elliptic curve) the total length of the proofs is $288 \cdot n + 128$ bytes. For $n = 30$, for example, the proofs are about 9KB in total length.

Once the ad-network receives the vector of n ciphertexts $\big(E(\mathsf{pk}, v_1),\ \ldots, E(\mathsf{pk}, v_n)\big)$ along with the zero-knowledge proofs, it checks the proofs and rejects the response if the proofs are invalid. In this case, the browser is not following the protocol and none of the advertisers are charged for the ad impression.

If the proofs check out, then the ad-network knows that the vector correctly represents a single ad. It multiplies the vector by the scalar c representing the cost of the impression. The result is a vector

$$\big(E(\mathsf{pk}, c \cdot v_1), \ldots, E(\mathsf{pk}, c \cdot v_n)\big).$$

All these quantities are an encryption of zero except for one component which is an encryption of c.

The ad-network next uses the additive homomorphic property to add the encrypted values to the encrypted counter for each ad. That is, for $i = 1, \ldots, n$ the ciphertext $E(\mathsf{pk}, c \cdot v_1)$ is "added" to $C_{\mathsf{ID}i}$ where ID_i is the ID of the i-th ad.

The end result of this process is that the quantity c is added to the counter of the ad displayed to the user. The value of all other counters is unchanged (although the ciphertext representing this value is changed for all counters). Security of the public-key system ensures that the ad-network does not know which counter was changed.

Settlement. Finally, at the end of the billing period the ad-network needs to charge the advertiser for all the impressions the ad generated. To do so it will need to decrypt the encrypted counter associated with all ads belonging to this advertiser. There are two options, of which the first is our preferred approach.

- Have a trusted third party (TTP) decrypt the counters for all ads. Here the TTP is mostly offline and is only contacted once per billing cycle. The secret key corresponding to the public key pk is held at the TTP. The ad-network will need to send the encrypted counters to the TTP at the end of each billing cycle. The TTP decrypts and sends the response to the ad-network, possibly along with a zero-knowledge proof showing that decryption was done correctly. Playing the role of the TTP would fit in with the business model of organizations such as Verisign. To avoid over-reliance on any one party, the role of the TTP can be split into a k-out-of-n system using threshold cryptography.

- Have the advertiser decrypt counters belonging to its ads. In this case we would need a separate public key pk for each advertiser and the secret key will be managed by the advertiser. As part of decrypting the counter, the advertiser will provide the ad-network with a zero-knowledge proof that decryption was done correctly.

  We believe this solution is unworkable in practice since there is a strong incentive for the advertiser to collude with the ad-network and frequently decrypt the counter to learn which ads were served where.

The difficulty with ultra low-rate campaigns. Consider an ad that is rarely sent out, say it is only sent out to one user during an entire billing cycle. Let's call that user Alice. If at the end of the billing cycle the encrypted counter for that ad is 1, the ad-network learns that the ad was presented to Alice and therefore Alice is interested in the topic of that ad.

This extreme example shows that ultra low-rate campaigns can reveal user interests to the ad-network. A simple solution is to ensure that the TTP does not send low counter values back to the ad-network. For example, the TTP can add low variance noise to the decrypted counter or it can round the decrypted value to an acceptable grid point.

Advertising budgets. When launching ad campaigns, advertisers typically specify a maximum budget for the campaign. When the budget is exceeded, no more ads from the campaign are displayed. Our encrypted counter approach appears to prevent the ad-network from telling when the campaign budget is exhausted.

This issue can be addressed by estimation. If the campaign budget is B the ad-network can estimate how long it will take until the budget runs out. At the end of that period it would ask the TTP to decrypt the counter to obtain the accurate billing statement (subject to the low noise discussed in the previous paragraph). The ad-network credits the remaining budget to the advertiser, or accepts the loss if it underestimated the rate at which the budget will be used up.

6 Implementation

The current implementation of *Adnostic* consists of two modules. The user profiling module monitors browsing activity to build a list of user interests. The ad rendering module selects ads based on the user profile and inserts them into *Adnostic* compliant web pages. It can be downloaded as a Firefox extension at http://crypto.stanford.edu/adnostic/

6.1 User Profiling

Each time the user visits a web page, *Adnostic* extracts keywords from the page meta-data (keywords, description,

title) and URL. This list of keywords is then used to retrieve interest categories related to the page content. Once the list of categories related to a page is established, the list is stored via the Firefox tagging system. Therefore, when a user re-visits a page, interest categories are directly retrieved from the Firefox Tag database.

The categories derived from all the pages visited are aggregated into a continually updated overall interest profile. Pages are weighted unequally; three parameters are used to derive a weight: number of visits, number of clicks and the page viewing duration.

In the current version of the extension, the category list we use is derived from *Google Ad Preferences*. Therefore, the profile established by *Adnostic* could be exported to serve ads on the Google Ad Network. We now describe our categorization algorithm.

6.2 Web page categorization: precomputation

Google Ad Preferences uses a 3-level hierarchical categorization scheme with around 600 categories overall. Appendix A lists the top-level categories as well as the subcategories of one of the top-level categories.

Recall that our goal is to map keywords extracted from webpages into one or more of these categories. To accomplish this, we use a "folksonomy," specifically, a corpus of tags derived from delicious.com, a social-bookmarking site. Delicious.com exports a feed of site-wide bookmarking activity and tags in the form of a real-time RSS feed. We retrieved this feed continuously for a period of around 10 days, yielding 1.25 million entries. Each entry consists of the identity of the user who created the bookmark, the creation timestamp, the URL and title of the bookmarked page, and the tags assigned by the user. Folksonomies are an active area of research in the semantic web, information retrieval and data-mining communities [37, 23], and therefore we are making our dataset available to other researchers at http://crypto.stanford.edu/adnostic/.

Folksonomies are well-suited as corpora in information retrieval: they allow harnessing manual effort without monetary expense, and they stay up-to-date, quickly evolving the ability to categorize new words introduced into the popular lexicon. The usefulness of folksonomies in online advertising has been explored by authors at Yahoo Research, who propose using co-occurrences of tags to map search keywords to advertisements [10]. Our solution is similar, and is described below.

From each entry (i.e., delicious.com bookmark), we extract the title of the bookmarked page and the set of tags assigned by the user on delicious.com. Next, we apply stemming to each tag and each word in the title. Stemming is a natural language processing technique that turns each word into its word stem [44]. For instance, the words "interest," "interests," "interested," and "interesting" all result in the stem "interest." Thus, we convert each entry into a "bag of words" derived from the title and the tags.

Similarly, we convert each Google Ads Preferences (sub)-category into a bag of words; we include all of the parent categories in deriving the bag of words for a category. For example, *Beauty & Personal Care* $\rightarrow$ *Spas & Beauty Services* $\rightarrow$ *Massage Therapy* is converted into the set $\{$*beauty, care, massage, personal, services, spas, therapy*$\}$. Stemming is applied here as well. In the following discussion, tags and category-words are implicitly assumed to have already been stemmed.

The next step is the key to the algorithm: we create a matrix of cosine-similarity values between each of the most common 20,000 tags on delicious.com and each word that appears in any of the Google Ads Preferences categories (there are around 700 distinct category-words).[3] Cosine similarity is a simple and robust measure of similarity between two multi-dimensional vectors [42]. We derive binary vectors for tags/category-words, with a 1 corresponding to each entry containing that tag or word and 0 for each entry that does not. We then use cosine similarity to quantify the co-occurence of two concepts (tags/words) on delicious.com.

Given this pre-computed matrix of cosines, computing the categories for a list of keywords (obtained from a web page) is straightforward. Let us define the similarity between the keyword-list K and a category C as the mean cosine similarity between each keyword in K and each word in C. The similarity scores between K and each of the roughly 600 categories are computed, and the top 5 categories along with their scores are returned. If only top-level categories are needed, then scores of the descendant categories of each top-level category are aggregated.

A manual examination of the results shows that the categorization is almost always accurate; we are investigating ways to assessing the accuracy more formally.

6.3 In-browser categorization

When *Adnostic* is downloaded, it comes built-in with the list of categories and the cosine-similarity matrix described above, but not the folksonomy database. As long as the cosine-similarity matrix is available, the original tag database is not required in order to carry out the computations. An update to the folksonomy results in an updated matrix, and is distributed to users as an update to the extension itself. The key bottleneck, then, in terms of the download/on-disk size is the space requirement of the matrix.

[3] Recall that there are about 600 categories. Most categories have multiple words, and each word occurs in multiple categories, so there is no relationship between the two numbers.

Recall that the matrix consists of 20,000 rows and 700 columns. However, the matrix is sparse; therefore it can be compressed to less than 1 megabyte. We consider this an acceptable overhead to the *Adnostic* download size; further, the small size allows the matrix to be stored entirely in memory when *Adnostic* is active, making accesses very fast.

Finally, there is another approach that we have not yet investigated that can be used to augment our categorization algorithm: download a list of pre-categorized URLs, and use it to simply look up the categories. The folksonomy-based categorization algorithm is applied only as a fall-back if the URL lookup fails.

At first glance, it might appear that such an approach would result in a huge download size, but we must keep in mind that a small fraction of all the URLs account for the majority of visits. This can be quantified using the delicious.com database: the top 30,000 most frequently bookmarked URLs account for 630,000 entries, or about half of the total of 1.25 million. Assuming that the URLs are bookmarked on delicious.com with the same relative frequency with which they are visited, roughly half of page views would result in a hit. A pre-computed category lookup is potentially more accurate and also saves computation on the client, and is therefore worth investigating.

6.4 Ad rendering

We now describe the protocol by which a list of ads is sent to the browser, and also the process by which the browser selects and displays an ad.

Adnostic exposes an object `adnostic` in the global Javascript namespace. To test if the user has *Adnostic* installed, the web page checks for the existence of the variable. This is similar to the current practice for checking for the presence of Javascript libraries such as jQuery.

The API that *Adnostic* exposes consists of one function, `adnostic.render`. Before we describe the arguments to this function, let us briefly discuss how we can map current targeting algorithms to a browser-based system.

The inputs to the ad targeting algorithm in current systems consists of inputs from the advertiser, the publisher, and the user. Thus, in an *Adnostic*-like system, we need to encapsulate the first two inputs in a form that can be sent to the browser. There must also be some way to specify how to combine these with the user's inputs.

Let us examine the user-side inputs in more detail. This of course consists of the behavioral profile, but in addition, it might also be useful to incorporate information about the history of ads shown to the user. The reason for this is to enable heuristics that are used in current systems such as *frequency capping*, which refers to limiting the number of impressions of the same ad in a given time-period in order to avoid "saturation."

The level of expressiveness to allow is a tricky trade-off. At one extreme, one might imagine supporting a *query language* that allows essentially arbitrary computations over the user's behavioral profile and history of ad impressions. This is risky for a number of reasons, including the fact that it makes privacy breaches resulting from the user clicking on ads potentially much more serious.

Another extreme is to allow each ad to belong to a category, and allow no other customization: the browser picks the ad most closely matching the user's profile (possibly with some randomization), and all other heuristics are implemented by the browser with no input from the ad-network. This approach obviously leads to a poor accuracy of targeting.

We propose a middle ground. The ad-network sends the following parameters along with each ad 1) a list of behavioral categories, 2) a score representing how relevant the ad is to the page, and 3) for each *extension* (see below), any numerical parameters that the extension accepts.

The browser derives a combined score for each ad based the score sent by the ad-network and on how well the list of categories match the user's profile. Since there are multiple categories per ad, the number of categories can be significantly greater than the number of ads, while still achieving a full coverage of the interest space. Extensions are modifications to the basic targeting algorithm, and need to be explicitly encoded into *Adnostic*. Currently the only extension we have identified is frequency capping, and the ad-network is allowed to specify the frequency.

We are now ready to describe the inputs and implementation of `adnostic.render()`. For each ad, the associated attributes are an id, the URL, and the targeting inputs described above. There are also the height and width parameters, which need to be the same for all ads, for reasons explained in Section 4. Finally, there is the cryptographic key material required if the billing protocol of Section 5 is used.

The browser creates n DOM elements, one for each ad, with the corresponding URL, height and width parameters. An ad is then selected as described above and all the ad DOM elements except the selected one are hidden. Thus, all the ads are downloaded, but only one is displayed to the user. As discussed in Section 4.1, a malicious ad-network could attach a script to report the visibility state of each enclosed element and find out which ad was displayed. Such script could be embedded either in one of the ads sent to *Adnostic* or in any other DOM element that the ad-network could place on the enclosing page. To prevent this, our extension places these DOM elements in their own origin and blocks execution of all active content in the n created DOM elements [19].

6.5 Evaluation

We first evaluate the cost of *Adnostic* in term of advertisement rendering delay and then observe its impact on the page loading time. Some websites publish many ads, make intensive use of scripts and include many external elements that take time to load. Although *Adnostic* increases loading delay, its relative cost might be negligible on heavy websites and more significant on lightweight websites. To reflect this heterogeneity, four websites with different type of ads are considered in this evaluation:

- Website 1 (SlashDot), is a lightweight website publishing on average three banners.
- Website 2 (ReadWriteWeb), is a heavy website publishing on average thirteen banners and including content from external websites.
- Website 3 (SecretSoftwareClub), is a very lightweight website publishing only three text ads.
- Website 4 (TheRegister), is a website publishing text ads and banners.

Three different configurations of *Adnostic* , varying privacy and ads, are compared to the current targeting model. To carry out this experimentation, we replace each link to an ad-network by n iframes, each iframe downloading an ad from an external server. When a page is loaded by *Adnostic* the content of the ad-network script is replaced by the *Adnostic* rendering script which injects n iframes in place of the ad-network ad. The number of iframes and their content vary in the evaluated configurations. In the first evaluated configuration, each ad is replaced by ten banners of 50 kB. The two other configurations download respectively 10 and 20 Text Ads. Each evaluation is the result of at least 100 experimentations; results are illustrated with a 95 percent confidence interval.

6.5.1 Ad rendering time

Figure 1 depicts the average ad rendering time on the four websites. As one might expect, *Website 3* achieves the fastest rendering time since it publishes only text ads. *Adnostic* impact on average ad rendering time is significant in every configuration and on every website. Unsurprisingely, ads are rendered faster if *Adnostic* just downloads 10 text ads and increasing n from 10 to 20 significantly increases the average delay. When banners are displayed instead of text ads, average rendering time also increases since the size of the content to be downloaded is multiplied by at least ten. In fact, ad rendering delays are similar when *Adnostic* downloads ten banners instead of twenty text ads. Although the former configuration downloads more bytes, the latter opens twice more connections and is thus more impacted by Round Trip Time (RTT) and browser connection limits.

6.5.2 Page loading time

Figure 2 provides a comparison of the page loading time on the four different websites. As we expected, *Adnostic* impact on page loading delay is relatively low compared to the impact on ad rendering delay. On most website, using *Adnostic* to display text ads will not notably degrade browsing experience if n is set to 10 or less. Furthermore, for Websites 1, 3 and 4, *Adnostic* configuration has also little impact on the page loading delay.

The noticeable exception is *Website 2* which includes a lot of external content and publishes many ads. In order to load the page, the browser opens many connections thus increasing the delay. In fact, to download the content of the thirteen ads, Firefox should open one hundred and thirty connections if n=10 and two hundred and sixty when n=20. Recall that the standard configuration of Firefox 3.5 limits the number of opened connections to 30 with a limit of 15 connection per server, many connections have to be delayed until a previous connection closes. These limits have a greater impact on *Adnostic* configurations with high values of n.

Increasing the limit on the number of simultaneous connections would improve *Adnostic* performance but also degrade the general browsing experience. To provide a significant performance improvement, the number of simultaneous connections would have to be very high. A better alternative would be to modify both *Adnostic* client and ad server to fetch the n ads via a single HTTP request. Such an approach would also reduce the impact of round-trip time on page loading and resource consumption on the client and server. The main drawback of this approach is the need to modify the ad server.

7 Alternative privacy architectures

Adnostic is designed to provide specific privacy guarantees. Here we briefly discuss a spectrum of other architectures for behavioral targeting depending on what categories of user data are revealed, and to which parties. These are arranged in the decreasing order of privacy violation, and in the increasing order of complexity of the system required to implement them.

- **Reveal the clickstream.** This is the current paradigm. The ad-network aggregates the user's clickstream and uses it to compute the behavioral profile.

  None of the following architectures *prevents* the ad-network from learning the clickstream; they merely *remove the need* to do so. As we discuss in Section 4, en-

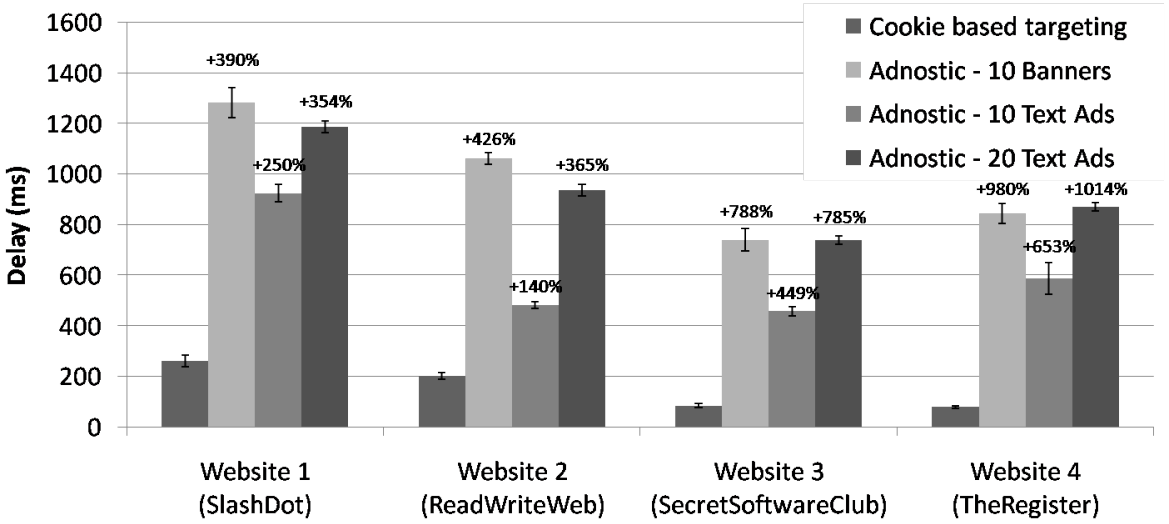

Figure 1. Average ad rendering time

"Cookie based targeting" refers to the current deployed tracking paradigm.
Percentages are shown in comparison to cookie based targeting.

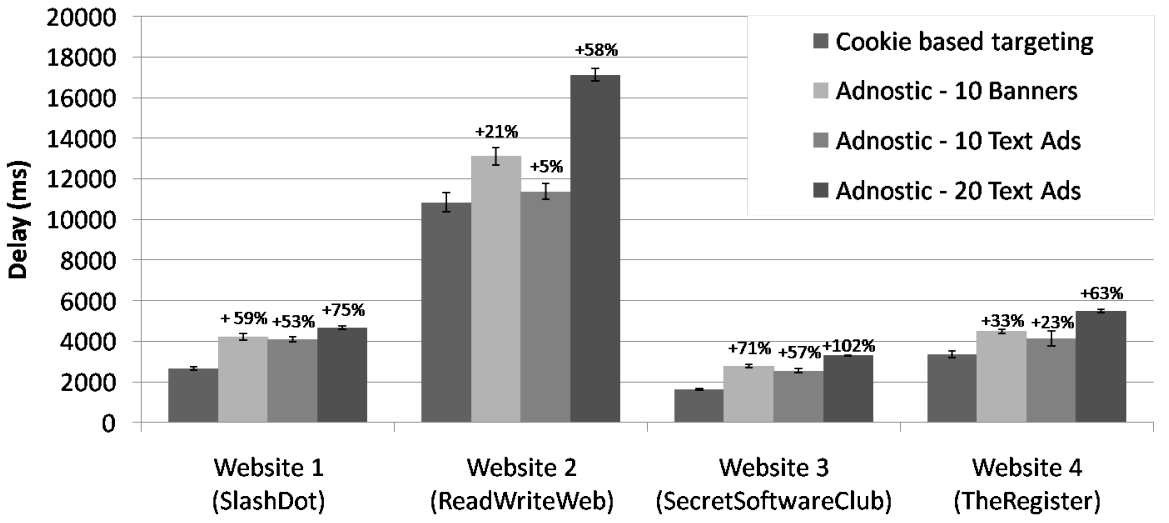

Figure 2. Average page loading time

forcement of non-tracking by the ad-network can only be done contractually.

- **Reveal the behavioral profile.** The browser computes the behavioral profile, as does *Adnostic* (Section 6.1). However, with every ad request the browser sends the profile to the ad-network in an HTTP header (similar to sending a cookie.) The bandwidth requirements are minimal: even with 1,000 categories, it is only about a kilobyte of data. Further optimization is possible: the profile need only be sent once to each ad-network per time period (say a week).

- **Reveal ad impressions to the CDN.** The ad-network offers a selection of ads to display, but is not told which ad was selected. The browser tells the CDN which ad was selected and downloads that ad from the CDN. If the CDN and the ad-network are non-colluding entities then the privacy guarantee is as strong as the following item (i.e., *Adnostic*), but is considerably more efficient. Care is needed in the implementation to ensure that nothing more than ad impressions are leaked to the CDN – specifically, the referer header must not be transmitted.

- **Reveal ad clicks to the ad-network (and advertiser)**. This is the *Adnostic* approach. Ad impressions are not revealed because the browser chooses to download all the ads that the ad-network offers, but only displays one of them.

- **Reveal nothing**. This is the approach taken by *Privad*, which we discuss in Section 8.

8 Related work

An independent project called *Privad* [21] has similar goals as *Adnostic* but is designed quite differently. *Privad* introduces four entities: *client*, *monitor*, *dealer* and *broker* (that we called the ad-network). The client builds a user profile and, according to this profile, requests relevant ads from the broker. The dealer anonymizes the client to prevent the broker from identifying the client. Communications between the client and the broker are also encrypted with the broker's public key and the dealer cannot see ads that are downloaded, viewed or clicked on by the user. The monitor is installed on user's computer and verifies that no information is sent by the client through a covert channel. Whereas *Adnostic* combines technological and policy based solutions, Privad is a purely technological approach privacy preserving targeted advertising. Consequently, *Privad* can not trust ad-networks and anonymizes every piece of information sent by the client. This anonymization impacts performance and makes click-fraud harder to detect.

Juels [25] proposed a personalized advertising delivery protocol based on Private Information Retrieval (PIR) and mix networks. With this scheme, advertisers furnish a negotiant function f that, for each type of profile, assigns the best corresponding ads in the advertiser database. In the most sophisticated version of the scheme, every customer has a public/ private key pair and publishes its ad request on a bulletin board. Advertiser servers apply a mix network based on m servers, that is used to retrieve each ad with $(m/2, m)$-group-privacy. Since the scheme makes extensive use of computationally intensive cryptographic operations, it cannot be used to retrieve ads on-the-fly. Instead, the author proposes to prefetch an advertisement and to wait until the user visits the page again to display it.

AdRosa [27] is an advertisement system that displays personalized ads on web portals. Personalization is based on user's behavior during an active session. Because it is based on active session, the AdRosa system does not require cookie based tracking. However, this session management is only designed for web portals and cannot be extended to support ad-networks. The occurrence of terms in titles, body and description field of a page are counted to infer the content of the page. Similarly, the content of an advertisement is retrieved by analyzing the content advertiser's website. Based on the content of the page visited by a user, AdRosa establishes the profile of the current active user. When a user visits a page, the portal provides a list of ads ranked according to user's profile, page content and ad content. Hence, the system implements behavior and content-based targeting, but does not protect user privacy.

TrackMeNot [24] is a tool used to hide user queries to search engines by issuing many fake queries. In principle, a similar approach can be used to defeat behavioral tracking: during a user session a browser extension can visit random pages unrelated to the user's interests. On the other side of the spectrum, IE8's InPrivate blocking can also defeat tracking, but potentially at the expense of breaking many web pages: publishers have no incentive to provide their content unless they can charge for an ad impression.

Faraht [17] describes a method for limiting the number of ads presented to the user in a user session without relying on cross-domain cookies. Instead, the approach uses page transition probabilities to limit the total number of viewed ads in expectation.

9 Conclusions

Our work exposes many of the issues that must be addressed in any attempt to reconcile the tension between behavioral targeting and user privacy. Our primary design goal is to support the wildly successful economic model currently used by ad-networks. In doing so we had to address issues of billing, advertising budgets, network latency,

targeting efficacy, malicious behavior, and many others. We proposed a cryptographic billing system that is likely to play a role in any proposal for privacy-preserving behavioral advertising.

The *Adnostic* system need not replace the existing ad infrastructure. Instead, we envision that *Adnostic* will complement it and offer more options for publishers and users in ad targeting. We hope to see further work on this topic exploring other designs with different trade-offs. Targeting with privacy is an important problem that needs more attention from the research community.

Acknowledgments. The authors would like to thank Adam Barth, Elie Bursztein, Collin Jackson, Foster Provost, Lynn Tao and anonymous reviewers for useful discussions and suggestions.

A Google Ads Preferences categories

Top-level categories

Animals
Arts and Humanities
Automotive
Beauty and Personal Care
Business
Computers and Electronics
Entertainment
Finance and Insurance
Food and Drink
Games
Home and Garden
Industries
Internet
Lifestyles
Local
News and Current Events
Photo and Video
Real Estate
Recreation
Reference
Science
Shopping
Social Networks and Online Communities
Society
Sports
Telecommunications
Travel

Subcategories of *Entertainment*

Entertainment → Celebrities
Entertainment → Clubs and Nightlife
Entertainment → Comics and Animation
Entertainment → Comics and Animation → Anime and Manga
Entertainment → Comics and Animation → Cartoons
Entertainment → Comics and Animation → Comics
Entertainment → Dancing
Entertainment → Entertainment Industry
Entertainment → Fashion and Modeling
Entertainment → Fun and Trivia
Entertainment → Humor and Bizarre
Entertainment → Humor and Bizarre → Bizarre
Entertainment → Humor and Bizarre → Humor
Entertainment → Humor and Bizarre → Paranormal
Entertainment → Movies
Entertainment → Movies → Bollywood and Lollywood
Entertainment → Movies → Horror Films
Entertainment → Movies → Movie Memorabilia
Entertainment → Movies → Movie Rentals and Sales
Entertainment → Movies → Science Fiction and Fantasy Films
Entertainment → Multimedia Content
Entertainment → Multimedia Content → Flash Content
Entertainment → Multimedia Content → Podcasting
Entertainment → Multimedia Content → Video Clips and Movie Downloads

References

[1] Adblock plus: Save your time and traffic. adblockplus.org/en/.

[2] Adblock plus statistics. https://addons.mozilla.org/en-US/statistics/addon/1865.

[3] Online advertising. en.wikipedia.org/wiki/Online_advertising.

[4] Google adsense. adsense.blogspot.com/2009/03/ driving-monetization-with-ads-that.html.

[5] AOL Advertising. Audience behavior. advertising.aol.com/advertiser-solutions/targeting/behavioral-targeting/audience-behaviors, 2009.

[6] Attributor. Get your fair share of the ad network pie, 2008. www.attributor.com/blog/get-your-fair-share-of-the-ad-network-pie.

[7] Audience science. www.audiencescience.com.

[8] Inc. AudienceScience. audiencetargeting.com, 2009.

[9] Bluekai. www.bluekai.com.

[10] Andrei Broder, Peter Ciccolo, Evgeniy Gabrilovich, and Bo Pang. Domain-specific query augmentation using folksonomy tags: the case of contextual advertising. In *Proceedings of the 1st Workshop on Information Retrieval for Advertising*, 2008.

[11] Federal Trade Commission. FTC staff report: Self-regulatory principles for online behavioral advertising, 2009. www.ftc.gov/opa/2009/02/behavad.shtm.

[12] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of Eurocrypt'97*, pages 103–118, 1997.

[13] Neil Daswani, Michael Stoppelman, and the Google Click Quality and Security Teams. The anatomy of clickbot.a. In *Proceedings of Usenix HotBots'07*, 2007.

[14] Doubleclick nearing privacy settlements, 2002. news.cnet.com/2102-1023-871654.html.

[15] Catherine Dwyer. Behavioral targeting: A case study of consumer tracking on levis.com. In *15th Americas Conference on Information Systems*, 2009.

[16] Exelate targeting exchange. exelate.com.

[17] Ayman Farahat. Privacy preserving frequency capping in internet banner advertising. In *Proceedings of WWW'09*, 2009.

[18] Fetchback. www.fetchback.com.

[19] Displaying web content in an extension without security issues. https://developer.mozilla.org/En/Displaying_web_content_in_an_extension_without_security_issues.

[20] Google. Ads preferences. www.google.com/ads/preferences, 2009.

[21] Saikat Guha, Alexey Reznichenko, Kevin Tang, Hamed Haddadi, and Paul Francis. Serving ads from localhost for performance, privacy, and profit. In *Proceedings of Hot Topics in Networking (HotNets)*, New York, NY, Oct 2009.

[22] Robert Hansen and Jeremiah Grossman. Clickjacking, 2008. www.sectheory.com/clickjacking.htm.

[23] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In *Proc. of The Semantic Web: Research and Applications*, volume 4011 of *LNCS*, pages 411–426. Springer, 2006.

[24] Daniel Howe and Helen Nissenbaum. TrackMeNot: resisting surveillance in web search, 2006. mrl.nyu.edu/~dhowe/trackmenot/.

[25] A. Juels. Targeted advertising ... and privacy too. In *Proceedings of CT-RSA 2001*, volume 2020 of *LNCS*, pages 408–424. Springer-Verlag, 2001.

[26] JupiterResearch. Measuring unique visitors: Addressing the dramatic decline in accuracy of cookie-based measurement. www.tvb.org/pdf/multiplatform/Jupiter-Research-Measuring-Unique/Visitors.pdf, 2004.

[27] P. Kazienko and M. Adamski. AdROSAadaptive personalization of web advertising. *Information Sciences*, 177(11):2269–2295, 2007.

[28] ClickZ Kevin Newcomb. Tacoda proposes behavioral targeting standards. www.clickz.com/3406921, 2004.

[29] Mozilla Labs. Weave. labs.mozilla.com/weave/.

[30] Scott J. LeCompte. Ua tracker: Tracking user-agents across the web. www.ua-tracker.com/.

[31] F. Monrose and A. Rubin. Authentication via keystroke dynamics. In *ACM Conference on Computer and Communications Security*, 1997.

[32] Nebuad. www.nebuad.com.

[33] Congress begins deep packet inspection of internet providers. bits.blogs.nytimes.com/2009/04/24/congress-begins-deep-packet-inspection-of-internet-providers, 2009.

[34] Phorm. www.phorm.com.

[35] Foster Provost, Brian Dalessandro, Rod Hook, Xiaohan Zhang, and Alan Murray. Audience selection for on-line brand advertising: Privacy-friendly social network targeting. In *Proceedings of the 15Th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2009.

[36] Justin Samuel. Requestpolicy. www.requestpolicy.com/. A Firefox extension that gives the user control over cross-site requests.

[37] Christoph Schmitz, Andreas Hotho, Robert Jäschke, and Gerd Stumme. Mining association rules in folksonomies. In *Data Science and Classification*, pages 261–270. Springer, 2006.

[38] Seth Schoen. New cookie technologies: Harder to see and remove, widely used to track you. www.eff.org/deeplinks/2009/09/new-cookie-technologies-harder-see-and-remove-wide, 2009.

[39] Christopher Soghoian, Sid Stamm, and Daniel Witte. Targeted advertising cookie opt-out (taco). taco.dubfire.net/.

[40] Ashlee Vance. Times web ads show security breach, 2009. www.nytimes.com/2009/09/15/technology/internet/15adco.html.

[41] Wikipedia. Comparison of browser synchronizers — wikipedia, the free encyclopedia. en.wikipedia.org/w/index.php?title=Comparison_of_browser_synchronizers&oldid=311634379, 2009. [Online; accessed 3-September-2009].

[42] Wikipedia. Cosine similarity — wikipedia, the free encyclopedia, 2009. [Online; accessed 18-December-2009].

[43] Wikipedia. Market adoption of mozilla firefox — wikipedia, the free encyclopedia, 2009. [Online; accessed 14-December-2009].

[44] Wikipedia. Stemming — wikipedia, the free encyclopedia. en.wikipedia.org/w/index.php?title=Stemming&oldid=314022163, 2009. [Online; accessed 15-September-2009].

[45] Yahoo! Ad interest manager. info.yahoo.com/privacy/us/yahoo/opt_out/targeting, 2009.