Privad: Practical Privacy in Online Advertising

Saikat Guha, Bin Cheng, Paul Francis
Microsoft Research India, and MPI-SWS
saikat@microsoft.com, {bcheng,francis}@mpi-sws.org

Abstract

Online advertising is a major economic force in the Internet today, funding a wide variety of websites and services. Today's deployments, however, erode privacy and degrade performance as browsers wait for ad networks to deliver ads. This paper presents Privad, an online advertising system designed to be faster and more private than existing systems while filling the practical market needs of targeted advertising: ads shown in web pages; targeting based on keywords, demographics, and interests; ranking based on auctions; view and click accounting; and defense against click-fraud. Privad occupies a point in the design space that strikes a balance between privacy and practical considerations. This paper presents the design of Privad, and analyzes the pros and cons of various design decisions. It provides an informal analysis of the privacy properties of Privad. Based on microbenchmarks and traces from a production advertising platform, it shows that Privad scales to present-day needs while simultaneously improving users' browsing experience and lowering infrastructure costs for the ad network. Finally, it reports on our implementation of Privad and deployment of over two thousand clients.

1 Introduction

Online advertising is a key economic driver in the Internet economy, funding a wide variety of websites and services. Internet advertisers increasingly work to provide more personalized advertising. Unfortunately, personalized online advertising comes at the price of individual privacy [22]. Privacy advocates would like to put an end to advertising models that violate privacy, and indeed have had some success with startups in the early stages of deployment [18]. On the other hand, they have had little success with the more entrenched ad brokers like Google and Yahoo! [11]. Arguably the reason why privacy advocates have failed here is that they offer no viable alternatives, and so the privacy solution they propose is effectively to end on-line advertising. This paper presents a practical and substantially more private online advertising system that attempts to offer that alternative.

To effect real change in the privacy of commercial advertising systems, we require that our design goals for Privad include commercial viability. This in turn requires that Privad:

1. is private enough that privacy advocacy groups[1] support it,
2. targets ads well enough to produce better click-through rates (or conversion rates, etc.) than current systems,
3. is as or less expensive to deploy than current systems, and
4. fits within the current business framework for online advertising, and therefore more likely has a viable business model. In particular, the interaction between Privad and end users, advertisers, and publishers, should not significantly change.

These goals are contradictory in nature, and much of the design challenge is finding the right balance of privacy and practicality. Although our arguments for scalability (goal 3) are strong and are buttressed by trace-based analysis, microbenchmarks, and deployment, we cannot definitively say that we have satisfied the other goals. While we hope to demonstrate better targeting through an experimental deployment (goal 2), this remains future work. The business model (goal 4) can ultimately only be demonstrated through a successful commercial deployment. While we have discussed our design with a number of privacy advocates, and have gotten favorable responses (goal 1), it is nevertheless hard to predict how they would react to a serious commercial deployment.

In practice we believe that a commercial deployment of Privad would be a constant balancing act between the goals listed above: the broker would gauge the reaction of privacy advocates, and strengthen or weaken privacy in response. In the absence of this commercial deployment and meaningful feedback from privacy advocates, our design assumes that privacy advocates will be hard to win over, and therefore favors privacy concerns over business concerns. In other words, our design attempts to produce the most private system possible within the constraint of achieving a merely *feasible* business model. In this paper, we nail down a design, present arguments as to why our practical goals are feasibly satisfied, and

[1] Private organizations like the Electronic Frontier Foundation (EFF) and the American Civil Liberties Union (ACLU), and government organizations like the Federal Trade Commission (FTC) and Europrise.

describe the security and scalability properties that our design ultimately achieves.

Privad preserves privacy by maintaining user profiles on the user's computer instead of in the cloud. A small amount of information necessarily leaves the user's computer: coarse-grained classes of ads a user is interested in, the ads the user has viewed or clicked on and the websites that carried the ads, and the ranking of ads for auctions. This information, however, is handled in such a way that no party can link it back to the individual user, or link together multiple pieces of information about the same user. An anonymizing proxy hides the user's network address, while encryption prevents the proxy from learning any user information. A trusted open-source reference monitor at the user's computer prevents any Personally Identifying Information (PII) other than network address from leaving the computer.

By contrast, current advertising systems, such as Google and Yahoo!, are in a deep architectural sense not private: they gather information about users and store it within their data centers. These systems do not lend themselves to being audited by privacy advocates or regulators. Users are essentially required to completely trust these systems to not do anything bad with the information. This trust can easily be violated, as for instance in a confirmed case where a Google employee spied on the accounts of four underage teens for months before the company was notified of the abuses [4].

Privad is considerably more private than current systems (though admittedly this is a low bar; we believe that privacy advocates will hold us to a much higher standard). Privad does not, for instance, require trust in any single organization. Additionally, Privad is designed to be auditable by third-parties. Most of this auditing is automatic, through the use of a simple reference monitor in the client. While Privad makes it much harder for an organization to gather private user information, Privad's privacy protocols are not bullet-proof (for instance with respect to collusion and covert channels), and so Privad allows the use of human-assisted or learning-based monitoring to detect misbehavior at the semantic level.

The anonymizing proxy (called *dealer*) is a significant change to the current business framework (goal 4). The dealer is run by an untrusted third-party organization, e.g. datacenter operators. We discuss in later sections the justification behind the dealer model, auditing mechanisms, and the feasibility of providing the service. We estimate the dealer's operating cost at around a cent per user per year (Section 4). This can easily be met with funding from privacy-advocates or levies on brokers.

The other significant change is client software on the users' computers. A key challenge, then, is incentivising deployment of this client software. Privad is not aimed for users that disable ads altogether. For users

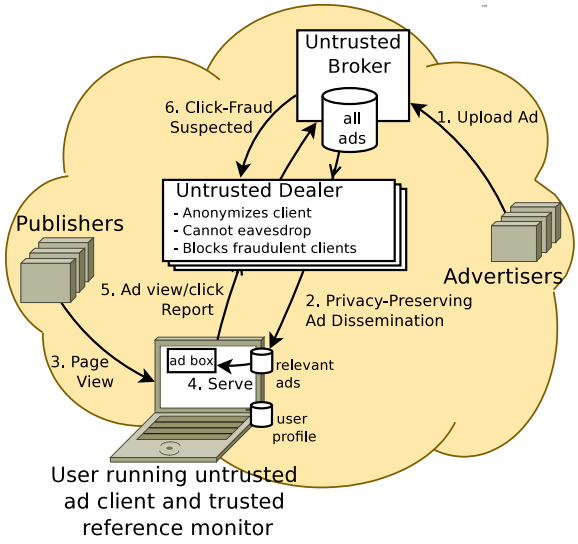

Figure 1: The Privad architecture

that do view and occasionally click ads, deploying requires first that Privad not degrade user experience in any way. We can ensure this by only showing ads in the same ad boxes that are common today (unlike previous adware, which employed disruptive advertising). Second, especially early on there must be some positive incentive for users to install it. This could be done through bundling other useful software, shopping discounts, or other incentives. Finally, it requires that privacy advocates endorse Privad. This at least prevents anti-virus software from actively removing the Privad client. Ideally, it even leads to privacy-conscious browser vendors (e.g. Firefox), anti-virus companies, or operating systems installing it by default.

The contributions of this paper are as follows: it presents a complete *practical* private advertising system. It describes the design of Privad, presents a feasibility study, and contributes a security analysis including both privacy and click-fraud aspects. It also gives a performance evaluation of our complete proof-of-concept implementation and pilot deployment of over two thousand users. Overall, Privad represents an argument that highly-targeted practical online advertising and good user-privacy are not mutually exclusive.

2 Privad Overview

There are six components in Privad: client software, client reference monitor, publisher, advertiser, broker, and dealer (see Figure 1). Publisher, advertiser, and broker all have analogs in today's advertising model, and play the same basic business roles. *Users* visit *publisher* webpages. *Advertisers* wish their ads to be shown to users on those webpages. The *broker* (e.g. Google) brings together advertisers, publishers, and users. For

each ad viewed or clicked, the advertiser pays the broker, and the broker pays the publisher.

There are three new key components for privacy in Privad. First, the task of profiling the user is done at the user's computer rather than at the broker. This is done by *client* software running on the user's computer. Second, all communication between the client and the broker is proxied anonymously by a kind of proxy called the *dealer*. The dealer also coordinates with the broker (using a protocol that protects user privacy) to identify and block clients participating in click-fraud. Finally, a thin trusted *reference monitor* between the client and the network ensures that the client conforms to the Privad protocol and provides a hook for auditing the client software. Encryption is used to prevent the dealer from seeing the contents of messages that pass between the client and the broker. The dealer prevents the broker from learning the client's identity or from linking separate messages from the same client.

At a high level, the operation of Privad goes as follows. The client software monitors user activity (for instance webpages seen by the user, personal information the user inputs into social networking sites, possibly even the contents of emails or chat sessions, and so on) and creates a user *profile* which contains a set of user *attributes*. These attributes consist of short-term and long-term *interests* and *demographics*. Interests include products or services like sports.tennis.racket or outdoor.lawncare. Demographics include things like gender, age, salary, and location.

Advertisers submit ads to the broker, including the amount bid and the set of interests and demographics targeted by each ad. The client requests ads from the broker by anonymously subscribing to a broad interest category combined with a few broad non-sensitive demographics (gender, language, region). The broker transmits a set of ads matching that interest and demographics. These ads cover all other demographics and fine-grained locations within the region, and so are a superset of the ads that will ultimately be shown to the user. The client locally filters and caches these ads. If the user has multiple interests, there is a separate subscription for each interest, and privacy mechanisms prevent the broker from linking the separate subscriptions to the same user.

Ad auctions determine which ads are shown to the user and in what order. The ranking function, identical to the one used in industry today, uses in addition to the bid information, both user and global modifiers. User modifiers are based on things like how well the targeting information matches the user, and the user's past interest in similar ads. Global modifiers are based on the aggregate click-through-rate (CTR) observed for the ad, the quality of the advertiser webpage, etc.

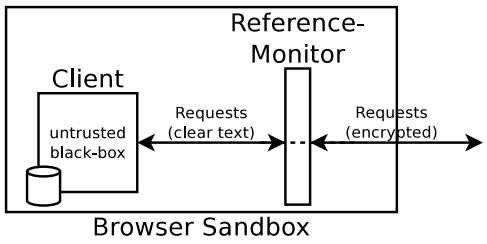

Figure 2: The Client framework

When the user browses a website that provides ad space, or runs an application like a game that includes ad space, the client selects an ad from the local cache and displays it in the ad space. A report of this *view* is anonymously transmitted to the broker via the dealer. If the user clicks on the ad, a report of this *click* is likewise anonymously transmitted to the broker. These reports identify the ad and the publisher on who's webpage or application the ad was shown. Privacy mechanisms prevent multiple reports from the same user from being linked together by the broker. The broker uses these reports to bill advertisers and pay publishers.

Unscrupulous users or compromised clients may launch click-fraud attacks on publishers, advertisers, or brokers. Both the broker and dealer are involved in detecting and mitigating these attacks (Section 3.4). When the broker detects an attack, it indicates to the dealer which reports relate to the attack. The dealer then traces these back to the clients responsible, and suppresses further reports from attacking clients, mitigating the attack.

Users, or privacy advocates operating on behalf of users, must be able to convince themselves that the client cannot *undetectably* leak private information. While having a trusted third-party write the client software might appear at first glance to be an option, it doesn't solve the problem — a trusted client simply moves the trust users place on brokers today to the third-party. At the same time, it requires brokers to make their trade-secret profiling algorithms known to the third party, and to parties auditing the client. Instead, Privad places a thin trusted reference monitor between the client and the network giving users and privacy advocates a hook to detect privacy violations (Section 3.5). It treats the client in a black-box manner (Figure 2), allowing the broker to use existing technological and legal frameworks for protecting trade-secret code. The reference monitor itself is simple, open source, and open to validation so its correctness can be verified, and can therefore be trusted by the user.

Note that Figure 1 does not portray the interaction that takes place between client and advertiser *after* an ad is clicked. For the purpose of this paper, we assume that a click brings the client directly to the advertiser as is the case today. We realize that this is a problem, because the finer-grained targeting of Privad gives unscrupulous ad-

vertises more information than they get today. The Privad architecture leaves open the possibility of privately proxying the post-click session between client and advertiser, and even protecting the client from inadvertently releasing sensitive information. Because of space limitations, we do not further discuss this option, and only consider protecting the user from the broker and dealer. Privad does not modify today's relationship between client and publisher.

3 Privad Details

This section provides details on ad dissemination, ad auctions, view/click reporting, click-fraud defense and the reference monitor. It also puts forth some of the rationale for our design decisions. These details represent a snapshot of our current thinking. While ad dissemination, reporting, and reference monitor are quite stable, the click-fraud defense, and auctions may easily evolve as we do more analysis and testing. We present them here so as to present a complete argument for Privad's viability.

3.1 Ad Dissemination

The most privacy-preserving way to disseminate ads would be for the broker to transmit all ads to all clients. In this way, the broker would learn nothing about the clients. In [13], we measured Google search ads and concluded that there are too many ads and too much *ad churn* for this kind of broadcast to be practical. We observed that the number of impressions for ads is highly skewed: a small fraction of ads (10%) garner a disproportionate fraction of impressions (80%). Furthermore, this 10% of ads tend to be more broadly targeted and therefore of interest to many users. It may therefore be cost effective to disseminate only this small fraction of ads to all users, for instance using a BitTorrent-like mechanism. For the remaining 90%, however, a different approach is needed. We therefore design a privacy-preserving pub-sub mechanism between the broker and client to disseminate ads.

The pub-sub protocol (Figure 3) consists of a client's request to join a *channel* (defined below), followed by the broker serving a stream of ads to the client.

Each channel is defined by a single interest attribute and limited non-sensitive broad demographic attributes, for instance wide geographic region, gender, and language. The purpose of the additional demographics is to help scale the pub-sub system: limiting an interest by region or language greatly reduces the number of ads that need to be sent over a given channel while still maintaining a large number of users in that channel (in the k-anonymity sense). Channels are defined by the broker. The complete set of channels is known to all clients, for instance by having dealers host a copy (signed by the broker). A client joins a channel when its profile attributes match those of the channel.

Figure 3: Message exchange for pub-sub ad dissemination. $E_x(M)$ represents the encryption of message M under key x. B is the public key of the broker. C is a symmetric key generated by the client for only this subscription.

The join request is encrypted with the broker's public key (B) and transmitted to the dealer. The request contains the pub-sub channel ($chan$), and a per-subscription symmetric key (C) generated by the client and used by the broker to encrypt the stream of ads sent to the client. The dealer generates for each subscription a unique (random) request ID (Rid). It stores a mapping between Rid and the client, and appends the Rid to the message forwarded to the broker. The broker attaches the Rid with ads published, which the dealer uses to lookup the intended client to forward the ads to.

The broker determines which ads should be sent and for how long they should be cached at the client. For instance, the broker stops sending ads for an advertiser when the advertiser nears his budget limit. Note that not all ads transmitted are appropriate for the user, and so may not be displayed to the user. For instance, an ad may be targeted towards a married person, while the user is single. Because the subscription does not specify marital status, the broker sends all ads independent of marital status or other targeting, and the client filters out those that do not match. Over time, the broker can estimate the number of ads that must be sent out for a particular advertiser to generate a target number of views and clicks.

3.2 Ad Auctions

Auctions determine which ads are shown to the user and in what order. For the advertiser, the auction provides a fair marketplace where the advertiser can influence the frequency and position of its ads through its bids. The broker additionally wants to maximize revenue, primarily by maximizing click-through rates (CTR). This is because most of today's advertising systems charge advertisers for clicks, not views. The broker also wants to minimize auction churn, generally by using a second-price auction [8]. A second-price auction is one whereby the bidder pays not the amount he bid, but the amount bid by the next lower bidder. This prevents the bidder from having to frequently change its bid in an attempt to probe for the bid value one unit higher than the next lower bidder.

Compared to today's brokers, which have full information about the system and can decide exactly which ads are shown where, in Privad both the client and the broker influence which ads are shown. This changes

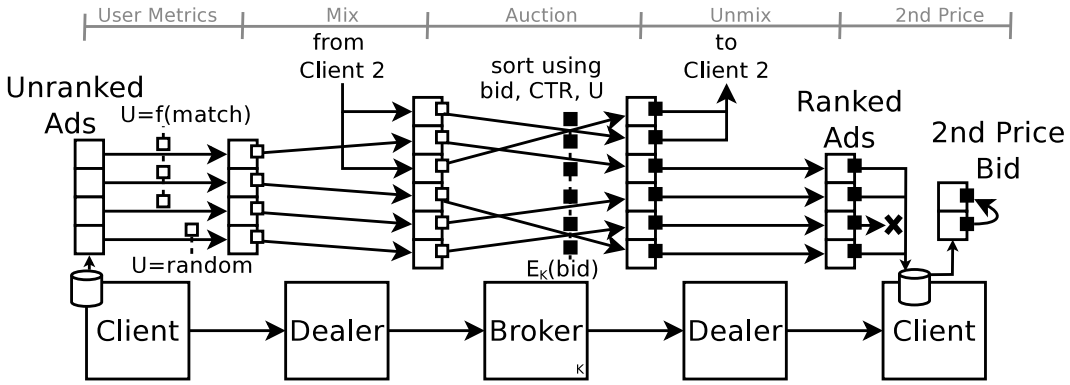

Figure 4: Industry-standard GSP Auction. Client annotates ads (across all channels) with quality of match, or random number if the ad doesn't match the user. Dealer mixes annotations from multiple clients. Broker ranks ads by bid, global click-through rate, advertiser quality, and match quality, and annotates the result with opaque bid information. Dealer slices auction result by client. Client filters out non-matching ads. Client reports encrypted second-price bid on click.

many aspects of the auction: for instance when the auction is run, over what set of ads, and the criteria by which second price is decided. The design space for Privad auctions is very large, and its complete exploration is a topic of further study. Nevertheless we describe two proof-of-concept auctions here.

A simple auction from this design space goes as follows. The broker periodically runs the auction over the set of ads targeted to a given pub-sub interest channel, producing a ranked set of ads. The ranking is preserved when ads are sent to clients. Clients filter out non-matching ads, slightly modify the ranking according to the quality of the demographic match for each ad, and show ads to users based on the modified ranking. When the broker receives a click report, it uses its original ranking to select the second price.

This auction is clearly different from Google's GSP auction [8]. For instance, with GSP, the auction is run when the browser requests a set of ads, and the second price is based on the ad below the clicked ad on the actual web page. We cannot necessarily say that our simple auction is worse than or better than GSP—this is a complex question and depends on, among other things, the evaluation criteria. As a demonstration of commercial viability, however, we now present a more complex auction that is identical to the industry-standard GSP auction mechanism.

In this second approach (Figure 4), the broker conducts the auction in a separate exchange. First, ads are sent to clients using pub-sub as originally described. The broker attaches a unique instance ID (Iid) to each copy of the ad published (not shown in figure). For each ad, the client computes a coarse score (U), typically between 1 and 5, as follows: for ads that match the user, the score reflects the quality of match with 5 signifying the best possible match. For ads that don't match the user, the score is a random number. To rank ads, the client sends (Iid, U) tuples for all ads in the client's database to the dealer. The dealer aggregates and mixes tuples for different clients before forwarding them to the broker. The broker ranks all the ads in the message. The ranking is based on both global and user modifiers (e.g. bids, CTR, advertiser quality, and client score). Note the ranked result contains all ads from the same client in the correct order, interspersed with ads for other clients (also in their correct order). The broker returns this ranked list to the dealer. The dealer uses the Iid to slice the list by client and forwards each slice to the appropriate client. The client discards the ads that do not match the user, and stores the rest in ranked order.

To obtain the GSP second price, the broker encrypts the bid information with a symmetric key (K) known only to the broker and sends it along with the ad. When a set of ads are chosen to be shown to the user, the client pairs up the encrypted bid information for ad $n+1$ with that of ad n. This encrypted bid pair is sent as part of the click report, which the broker decrypts to determine what the advertiser should be charged.

3.3 View/Click Reporting

Ad views and clicks, as well as other ad-initiated user activity (purchase, registration, etc.) needs to be reported to the broker. The protocol for reporting ad events (Figure 5) is straightforward. The report containing the ad ID (Aid), publisher ID (Pid), and type of event (view, click, etc.) is encrypted with the broker's public-key and sent through the dealer to the broker. The dealer attaches a unique (random) request ID (Rid) and stores a mapping between the request ID and the client, which it uses later to trace suspected click-fraud reports in a privacy-preserving manner.

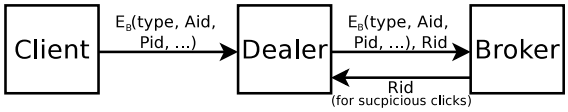

Figure 5: Message exchange for view/click reporting and blocking click-fraud. B is the public key of the broker. Aid identifies the ad. Pid identifies publisher website or application where the ad was shown. For second-price auctions, the opaque auction result is included. Rid uniquely identifies the report at the dealer.

3.4 Click-Fraud Defense

Click-fraud consists of users or bots clicking on ads for the purpose of attacking one or more parts of the system. It may be used to drive up a given advertiser's costs, or to drive up the revenue of a publisher. It can also be used to drive up the click-through-ratio of an advertiser so that that advertiser is more likely to win auctions.

Generally speaking, privacy makes click-fraud more challenging because clients are hidden from the broker. Privad addresses this challenge through an explicit privacy-preserving protocol between broker and dealer. Both the broker and dealer participate in detecting and blocking click-fraud; the dealer by measuring view and click volumes from clients, the broker by looking at overall click behavior for advertisers and publishers.

Blocking a fraudulent client once an attack is detected is straightforward. When a publisher or advertiser is under attack, the broker tells the dealer which report IDs are suspected as being involved in click-fraud. The dealer traces the report ID back to the client, and if the client is implicated more than some set threshold, subsequent reports from that client are blocked.

As with today's ad networks, there is no silver bullet for detecting click-fraud. And like ad networks today, the approach we take is *defense in depth* — a number of overlapping detection mechanisms (described below) operate in parallel; each detection mechanism can be fooled with some effort; but together, they raise the bar.

Per-User Thresholds. The dealer tracks the number of subscriptions, and the rates of view/click reports for each client (identified by their IP address). Clients that exceed thresholds set by the broker are flagged as suspicious. The broker may provide a list of NATed networks or public proxies so higher thresholds may apply to them.

Blacklist. Dealers flag clients on public blacklists, such as lists maintained by anti-virus vendors or network telescope operators that track IP addresses participating in a botnet. Dealers additionally share a blacklist of clients blocked at other dealers.

Honeyfarms. The broker operates honeyfarms that are vulnerable to botnet infection. Once infected, the broker can directly track which publishers or advertisers are under attack. When a report matching the attack signature is received, the broker asks the dealer to flag the originating client as suspicious.

Historical Statistics. The dealer and broker maintains respectively a number of per-client, and per-publisher and per-advertiser statistics including volume of view reports, and click-through rates. Any sudden increase in these statistics cause clients generating the reports to be flagged as suspicious.

Premium Clicks. Based on the insight behind [20], a user's purchase activity is used as an indication of honest behavior. Clicks from honest users command higher revenues. The broker informs the dealer which reports are purchases. The dealer flags the origin client as "premium" for some period of time, and attaches a single "premium bit" to subsequent reports from these clients.

Bait Ads. An approach we are actively investigating is something we term "bait ads" (similar to [14]), which can loosely be described as a cross between CAPTCHAs and the invisible-link approach to robot detection [26]. Basically, bait ads contain the targeting information of one ad, but the content (graphics, flash animation) of a completely different ad. For instance, a bait ad may advertise "dog collars" to "cat lovers". The broker expects a very small number of such ads to be clicked by humans. A bot clicking on ads, however, would unwittingly trigger the bait. It is hard for a bot to detect bait, which for image ads amounts to solving semantic CAPTCHAs (e.g. [9]). Bait ads are published by the broker just like normal ads. When a click for a bait ad is reported, the broker informs the dealer, which flags the client as potentially suspicious.

These mechanisms operate in concert as follows: per-user thresholds force the attacker to use a botnet. Honeyfarms help discover botnets, and blacklists limit the amount of time individual bots are of use to the attacker. Historical statistics block high-intensity attacks, instead forcing the attacker to gradually mount the attack, which buys additional time for honeyfarms and blacklists to kick in before significant financial damage is caused. At the same time, bait ads disseminated proactively can detect low volume attacks due to the strong signal generated by a relatively small number of clicks, while disseminated reactively, bait ads can reduce false positives. And finally, premium ads, by forcing the attacker to spend money to acquire and maintain "premium" status for each bot, apply significant economic pressure, which is magnified by bots being blacklisted.

Overall these mechanisms have the effect of more-or-less putting Privad back on an even footing with current ad networks as far as click-fraud is concerned.

3.5 Reference Monitor

The reference monitor has six functions geared towards making it difficult for the black-box client to leak pri-

vate information. We model the reference monitor on Google's Native Client (NaCl) sandbox [33] that allows running untrusted native code within a browser. As with NaCl, the sandbox presents a highly narrow and hardened API to untrusted code, and is itself open to validation by security experts and privacy advocates.

The reference monitor is hardened in at least the five follow ways. First, the reference monitor validates that all messages in and out of the client follow Privad protocols. For this, the client is operated in a sandbox such that all network communication must go through the reference monitor in the clear (Figure 2). Second, it is the monitor that encrypts outbound messages from the client (and decrypts inbound messages). Third, the monitor is the source of all randomness in messages (e.g. session keys, randomized padding for encryption etc.). Fourth, the monitor may additionally provide cover traffic or introduce noise to protect user privacy in certain Privad operations. Fifth, the monitor arbitrarily delays messages or adds jitter to disrupt certain timing attacks.

Technological means for disrupting covert channels is, of course, not enough since the client may attempt to leak information through semantic means. For instance, the client might send lima-beans when it really means no-health-insurance. The sixth and final function of the reference monitor is therefore to provide an auditing hook, which can be used for instance to interpose a human-in-the-loop. Interested users may occasionally inspect messages for accuracy, and/or privacy advocates may set up honeyfarm clients, train them with specific profiles, and monitor them for inconsistent behavior using automated techniques presented in [12].

3.6 User Profiling

Even though the client is ultimately in charge of profiling the user, it can nevertheless leverage existing cloud-based crawlers and profilers through a privacy-preserving query mechanism. At a high level the query protocol is similar to the pub-sub protocol (Figure 3) operating as a single request-response pair; the request contains the website URL and the response contains profile attributes. Beyond this, the client can locally scrape and classify pages, incorporate social feedback, or even allow publisher websites to explicitly influence the profile. Overall, the user profiling options in Privad *adds to* existing cloud-based algorithms while preserving privacy, and therefore has the potential to target ads better than existing systems.

4 Feasibility

To validate the basic feasibility of Privad, we estimate worst-case network and storage overhead based on a trace of ads delivered by Microsoft's advertising platform (processing overhead is measured in Section 6).

Network and storage overhead at the client is due primarily to pub-sub ad dissemination. We use a trace of Bing search ads to determine an expected number of channels per client and ads per channel. We make the pessimistic assumption that all ads associated with a channel are transmitted to all subscriptions for that channel. We expect to be far more efficient than this in practice, since we can design our pub-sub service so that clients receive only fractionally more ads than necessary to fill their ad boxes (subject to k-anonymity and advertiser budget constraints). Summarizing our results, assuming compression and a 1MB local cache, we estimate the client will download less than 100kB per day on average (worst case: 20MB cache, 1.25MB daily download: less than a typical MP3 song). Even adjusting for the fact that our trace represents a good fraction, but a fraction nevertheless, of the search advertising market, and doesn't include contextual advertising, this load poses little concern.

We arrive at these estimates as follows: The Bing trace we used (for over 2M users in the USA sampled on Sep. 1, 2010) classifies users and ads into 128 interest categories. On average, each user is mapped to 2 interest categories on a given day (9 categories in the 99^{th} percentile case). Using 2–4 coarse-grained geographic regions per state, we obtain several tens of thousand distinct interest-region-gender Privad channels. Remapping Bing ads to these channels results, on average, in slightly less than 2K ads for each channel (10K in the 99^{th} percentile); note, an ad may be mapped to multiple channels. Each ad is roughly 250 bytes of text including the URL. This results in an average unoptimized daily download size of around 1MB (and less than 25MB in the worst case). Compressing ad content (in bulk) reduces download size by a factor of 10.

Of these, only the subset matching the user's other demographic attributes need to be stored in the client's local cache. Using the Bing trace's age-group classification alone, we get a factor of 5 reduction in storage. Occupation, education, marital-status etc. may further reduce storage requirements but we lack data to estimate these. Cached ad data can then be used to further reduce client network traffic. This requires a slight modification to the pub-sub protocol to periodically transfer a bitmap of active/inactive ads on the channel. Based on two weeks of trace data, we find that 54% of ads on a channel were seen the previous day (and around 70% within the previous 4 days; there is little added benefit for caching beyond 4 days). Thus with a warmed up 1MB cache, the client needs to download on average 100kB (1.25MB worst case) of compressed ad content plus a few tens of kilobytes of periodic bitmap data per day. Privad does not change the number of ads viewed by the user; based

on the Bing trace we estimate the client's upload traffic will be less than 20kB per day on average.

Consequently, we estimate the broker will send around 100kB and receive around 20kB per client per day, while the dealer acting as a proxy will send (and receive) around 120kB per client per day. While broker network overhead is more than today, the Privad broker trades-off network for lower processing overhead. There is, however, no simple comparison of Privad broker processing overhead with that of existing systems. Todays systems are *synchronous*: they request a small number of ads frequently, and ad selection plus auction plus ad delivery must occur in milliseconds. Privad is *asynchronous*: a large number of ads are requested infrequently, and these do not have to be delivered immediately (overhead quantified in Section 6). Thus comparing overall broker costs depends, among other factors, on the reduction in broker processing overhead and corresponding reduction in datacenter provisioning costs, versus bandwidth costs. As for the dealer, the network overhead works out to less than 88MB per user per year. Assuming the dealer leases datacenter resources at market prices, this amounts to less than $0.01 per user per year (based on current Amazon EC2 pricing [2]).

5 Implementation and Pilot Deployment

We have implemented the full Privad system and deployed it on a small scale. The system comprises a client implemented as a 210KB addon for the Firefox web browser, a dealer, and a broker. Out of the 11K total lines of code, the dealer consists of only 700 lines — well within limits of what can be manually audited.

We have deployed Privad with a small group of users comprised primarily of 2083 volunteers[2] we recruited using Amazon's Mechanical Turk service [1]. The primary purpose of the deployment is to convince ourselves that Privad represents a complete system. To this end the deployment exercises all aspects of Privad including user profiling (by scraping the user's Facebook profile and Google Ad Preferences), pub-sub ad dissemination, GSP auctions, view/click reporting, and basic click-fraud defense. For test ad data we scrape and re-publish Google ads through our system; since we lack targeting information for these ads, we target randomly. The system has been in continuous operation since Jan 1, 2010, with over 271K ads viewed and 238 ads clicked as of Jan 6, 2011.

The primary implementation challenge is the effort required to scrape webpages for profiling purposes. Facebook's and Google's layout changed on multiple occasions during our deployment, which required us to update the client code (using the addon's autoupdate mechanism). We are presently working on a higher-level language (and interpreter) for scraping webpages that will allow us to react more quickly to website changes.

[2]Users were offered an average one-time reward of $0.40 (for the 1 minute it took on average to install the addon) with mechanisms in place to prevent cheating. While users were required to leave the addon installed for at least a week to get paid, most users either forgot about it or chose to leave it installed for longer. As of Jan 6, 2011, 429 users still have the addon installed.

6 Experimental Evaluation

We use microbenchmarks to evaluate our system at scale.

Broker: We benchmark first the performance of subscribe and report messages at the broker since they involve public-key operations. Without optimizations, as expected, performance is bottlenecked by RSA decryptions. While crypto optimizations could be offloaded to hardware, since the broker is in any event untrusted, we additionally have the option of offloading to idle (untrusted) clients in the system (without impacting privacy guarantees). With this optimization, the broker needs only perform symmetric-key (AES) and hashing (SHA1) operations, which can be done at line speed using dedicated hardware [21]. Our software-based implementation achieved a throughput of 6K subscribe and report requests per second on a single core, can publish 8.5K ads per second, and perform around 30K auctions per second. We note that request throughput in our broker is in the same ballpark as production systems today (based on the traces mentioned earlier); although this is somewhat of an apples-to-oranges comparison since brokers in Privad are much simpler.

In all cases the measured performance did not depend on the number of subscriptions or unique ads since all lookups at the broker are $\mathcal{O}(1)$; all runtime state (subscriptions, ads) is cached in memory and backed by persistent storage. The broker is designed with no shared state so it can trivially scale out to multiple cores.

Dealer: Our dealer can forward 15K requests per second (on a single-core 3GHz workstation) in both directions, which is sufficient for handling nearly 200K online clients (based on request rates from our deployment). The bottleneck is due to client-side polling which arises from implementing Privad's asynchronous protocols on top of a request-response based transport (HTTP). With the emerging WebSockets standard [16], we believe we can eliminate this polling and support well over a million clients per dealer core.

Client: Finally we focus on how Privad improves a user's web browsing experience by eliminating network round-trips in the critical path of rendering webpages. Figure 6 compares Privad performance to existing ad networks. The figure compares the delay added for both populating ad boxes (on the 20 most popular sites as ranked by Alexa), and for completing the redirect to the advertiser webpage after a click. For Privad, we measured the time taken to populate ad boxes as we scale the number of (relevant) ads cached in the client

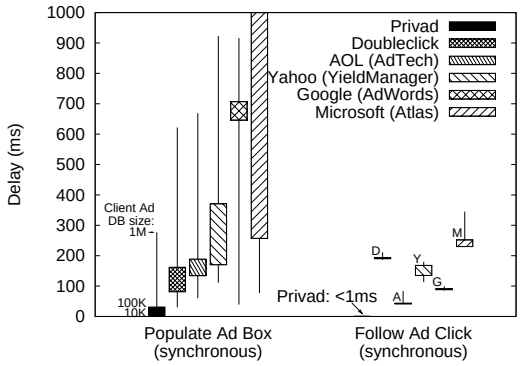

Figure 6: Privad eliminates network RTTs for showing ads, and reporting clicks. Whiskers for Privad show performance as the number of (relevant) ads in the client's database scales to 1 million. Whiskers and boxes for existing ad networks show minimum and maximum latencies, and quartiles.

database. As mentioned, we estimate the typical number of cached ads to be between 10K (average) to 100K (worst case); we benchmark with a factor of ten margin. As one might expect, our client implementation outperforms existing ad networks since displaying ads requires only local disk access. Our client can populate ad boxes, based on keywords or website context, in 31ms. In existing networks, we found the delay was dominated by the ad selection process; downloading the actual ad content (e.g. 30kB flash file) took less than 2ms. Doubleclick, which to our knowledge does not perform demographic or context sensitive advertising, took 129ms in the median case, and Google, which does perform context sensitive advertising, took 670ms. With regards to reporting clicks, existing ad networks must perform a synchronous redirect through the ad network, which consumes several RTTs. Since Privad reports clicks asynchronously (when browser is idle), the redirect is unnecessary, thus allowing much faster advertiser page-loads.

Our client scrapes webpages, pre-fetches ads, conducts auctions, and sends reports in the background. Messages that require public-key encryptions take between 68ms (on a workstation) to 160ms (on a netbook) to construct, but since they are performed when the browser is idle, they are imperceptible to the user. The client uses negligible memory since ads are stored on disk; there is no appreciable change in the browser's memory footprint whether the client is enabled or disabled. During our 12 month deployment, we have not received any negative feedback, performance related or otherwise, from users[3].

[3] or, for that matter, positive feedback.

7 Privacy Analysis

Broadly speaking, Privad uses technological means to protect user privacy. Privad provides privacy through *unlinkability* [27] (described below), and uses the dealer mechanism to ensure this. It is worth considering briefly alternative design points that we opted against.

Considering it is believed to be impossible to design systems that are secure against covert channels and collusion [17, 25], neither we, nor privacy advocates expect bulletproof privacy. Privacy advocates instead have the much softer requirement that "individuals [be] able to control their personal information", and if privacy is violated, the ability to "hold accountable organizations [responsible]" [5]. Privad trivially satisfies the first requirement by storing all personal information on the user's computer and assuring unlinkability. In the absence of covert channels or collusion, this prevents any organization from learning about users, thereby preventing privacy violations in the first place. In the presence of covert channels or collusion, the organization's willing and explicit circumvention of technological privacy safeguards strongly implies malicious intent (in the legal sense) to which they can be held accountable.

As a result, the oversight task for privacy advocates is reduced from detecting any kind of privacy violation, including those purely internal to a broker, to detecting collusion and the use of covert channels. As we discuss below, Privad incorporates existing (and future) techniques to disrupt or detect covert channels through the reference monitor mechanism and careful protocol design. Detecting collusion is easier with the dealer mechanism as compared to, say, a mixnet like TOR [6]. Not only does TOR not meet business needs by giving up any visibility into click fraud, TOR's threat model is a poor match for Privad since a single entry node colluding with the broker can compromise the anonymity of all users connecting through that node [3]. In contrast to mixnet nodes, a dealer organization (e.g. datacenter operators) can be contractually bound, and its non-collusionary involvement be monitored by privacy advocates. This model is in use today and is approved for instance by the European privacy certification organization Europrise [10].

Given that Privad relies to an extent on accountability, one might ask why a purely regulatory solution doesn't suffice. There are two problems. First, entrenched players like Google have strong incentives, lobbying power, and the capital needed to maintain the status quo. Indeed many parallels can be drawn to the network-neutrality battle where powerful ISPs successfully resisted new regulations threatening their business model [32]. Second, even if regulations were passed, enforcement would require third-party auditing of all broker operations, which is impractical due to the complexity and scale of these systems. Market forces, such as

competition from a startup offering better ROI to advertisers through deeper personalization (with backing from privacy advocates), can arguably effect change more easily.

In the remainder of this section we first define informally what we mean by user privacy and our trust assumptions. We then address the technical measures pertaining to covert channels. We then consider a series of attacks on the system, the defense to the attack, and a discussion of the extent to which the defense truly solves the attack.

7.1 Defining Privacy

Our privacy goals are based on Pfitzmann and Köhntopp's definition of anonymity [27] which is unlinkability of an *item of interest* (IOI) and some logical user identifier. Privad has three types of IOI; IP address, and interest attributes and demographic attributes. Pfitzmann and Köhntopp consider anonymity in terms of an *anonymity set*, which is the set of users that share the given item of interest — the larger this set, the "better" the anonymity. Personally Identifiable Information (PII) is information for which the anonymity set comprises a single (or a very small number of) elements; e.g., the IP address is PII. Examples of non-PII anonymity sets in Privad include: the set of users that join a pub-sub channel, the set of users that visit a given publisher, and the set of users that view or click a given ad (i.e. probably share some or all of the ad's attributes).

In our definition of privacy we draw a distinction between IOI that contain PII and IOI that do not, as follows:

P1) Profile Anonymity: No single player can link any PII for a user with any attribute in the user's profile.

P2) Profile Unlinkability: No single player can link together more than a threshold number of (non-PII) profile attributes for the same user, which would otherwise allow them to, over time, construct a unique profile that could be deanonymized using external databases.

Existing ad networks, of course, satisfy neither Profile Anonymity nor Profile Unlinkability.

Note that for Profile Unlinkability we use "number of profile attributes" rather than the size of the anonymity set even though the former doesn't per se map directly onto the latter. Different attributes imply different sizes of anonymity sets (e.g., music vs. sports.skiing.cross-country). Ideally, Privad would dynamically guarantee a minimum anonymity set size at runtime, but this is not possible because any such approach is easily attacked with Sybils [7], e.g. a botnet of clients masquerading as members of that set. It is possible, however, to estimate offline the rough expected anonymity set size for an attribute with outside semantic knowledge.

The approach towards privacy in Privad is then as follows: 1) offline semantic analysis by privacy advocates establishes per-message thresholds for Profile Unlinkability; this is enforced at runtime by the monitor as we discuss later in Attack A9. 2) Mechanisms in Privad ensure multiple messages from the same client cannot be linked together, and therefore the system as a whole cannot violate Profile Unlinkability. And 3) since the dealer is the only party that learns PII (IP address) and nothing else about the user, Profile Anonymity is trivially satisfied.

7.2 Trust Assumptions

The user trusts only the reference monitor; the client software, dealer and broker are all untrusted. Privacy advocates are expected to play a watchdog role by validating the reference monitor, monitoring dealer operation, and running honeyfarms to detect covert channels. The broker does not trust clients, dealers, or reference monitors. Attack A4 below discusses malicious dealers including those that may engage in click fraud. Privad does not modify any interactions users or brokers have with publishers or advertisers. The advertiser and publisher, like today, can see the user's browsing behavior on their own site, and trust the broker to perform accurate billing.

7.3 Covert Channels

A malicious broker may distribute a malicious client that attempts to leak data using covert channels. The bandwidth of covert channels is reduced by bounding nondeterminism in messages. Note first of all that the covert channel must come from Privad application message fields, not encapsulating protocol fields such as those in the crypto messages. This is because it is the reference monitor that takes care of crypto and message delivery functions. In addition, it is also the monitor that generates the one-time shared keys (for subscriptions) which otherwise represent the best covert channel opportunity.

Note next that the values of most message fields are driven by user behavior (outside client-control) and are subject to audit by privacy advocates or users. This includes the channel ID in subscriptions, and the type, publisher ID, and ad ID in reports, which together compose all remaining bits in subscribe and report messages. The next best opportunity for a covert channel would come from the user score in the GSP auction message (Figure 4). That is because this is the only client-controlled message field, albeit only 2 or 3 bits in size since the user score need only be in a small range. This bounds the information that can be leaked by a single message.

The Privad protocol and reference monitor make it hard to construct a covert channel across multiple messages. Since messages from the same source cannot, by design, be linked based on content, the attacker must use

some time-based watermarking technique (e.g., [31]). The reference monitor adds arbitrary delay or jitter to messages to disrupt such attempts. For this reason, all Privad protocols are designed to be asynchronous and use soft-state without any acknowledgments.

A computer system cannot completely close all covert channels, but by at least making it possible for privacy-advocates to detect them, and by establishing malicious intent by requiring attackers to circumvent multiple technical hurdles, Privad significantly increases the risk of being caught and thus decreases the utility of covert channels. This is in contrast to today where third-parties can neither detect privacy-violations, nor establish intent when violations are revealed [28].

7.4 Attacks and Defenses

This section outlines a set of key attacks on user privacy. Space constraints prevent us from discussing in detail attacks on advertiser and broker privacy. We do however briefly note the following. Broker privacy, in the form of trade secrets for profiling mechanisms, is maintained because client software is a black-box that does not need to be audited; and the broker can use the same legal and technical mechanisms used by desktop software companies today. Advertiser privacy is weakened because it is slightly easier to learn an ad's targeting information as compared to today's systems. Privad does not however change the ease with which an attacker can learn an advertiser's bids.

7.4.1 Attacker at Client

Attack A1: The attacker installs malware on a user's computer which provides the profile information to the attacker or otherwise exploits it.

Defense D1: Privad does not protect against malware reading the profile it generates. Our general stance is that even without Privad, malware today can learn anything the client is able to learn, and so not protecting against this threat does not qualitatively change anything. Having said that, obviously the existence of the profile does make the job of malware easier. It saves the malware from having to write its own profiling mechanisms. It also allows the malware to learn the profile more quickly since it doesn't have to monitor the user over time to build up the profile.

Ultimately what goes into the profile is a policy question that privacy advocates and society need to answer. Clearly information like credit card number, passwords, and the like have no place in the profile (though malware can of course get at this information anyway). Whether a user has AIDS probably also does not belong there. Whether a user is interested in AIDS medication, however, arguably may belong in the profile.

Indeed, there are pros and cons to keeping profile contents open. On the pro side, this makes it easier for privacy advocates to monitor the client and to an extent broker operation. On the con side, it makes life easier for malware. One option, if the operating system supports it, is to make the profile available only to the client process (e.g. through for instance SELinux [24]). This would protect against userspace malware, but not rootkits that compromise the OS. Another option is to leverage trusted hardware (e.g. [30]) when available. How best to handle the profile from this perspective is both an ongoing research question and a policy question.

7.4.2 Attacker at Dealer

A2: The attacker attempts to learn user profile information by reading messages at the dealer.

D2: The dealer proxies five kinds of messages: subscribe, publish, auction request and response, and reports. Of these, the dealer cannot inspect the contents of subscribe, report, and publish messages since the first two are encrypted with the broker's public key, and the last is encrypted with a symmetric key that is exchanged via the encrypted subscribe message. Auction messages, which are unencrypted, contain a random single-use Iid that identifies the ad at the broker and the client (exchanged over the encrypted publish message), but is meaningless to the dealer.

A3: The attacker injects messages at the dealer in order to learn a user's profile information.

D3: The dealer cannot inject a fake publish message since it would not validate at the client after decryption. If the dealer injects a fake subscribe message, all resulting publish messages would be discarded by the client since the client would not have a record of the subscribe or the associated key. The dealer cannot inject fake auction messages since the client would not have a record of the Iid. The dealer could reorder the auction result, but would not learn which ad the client viewed or clicked since reports are encrypted. The dealer injecting fake reports has no impact on the client; it is, however, identical to dealer-assisted click-fraud, which we consider next.

A4: The dealer itself engages in click-fraud, or otherwise does not comply with the broker's request to block fraudulent clients.

D4: The broker can independently audit that the dealer is operating as expected both actively and passively. The broker can passively track view/click volumes, and historical statistics on a per-dealer basis to identify anomalous dealers. Additionally the broker can passively monitor the rate of fraudulent clicks (e.g. using bait ads) on a per-dealer basis. The broker can detect suspicious dealer behavior if after directing dealers to stem a particular attack the rate of fraudulent clicks through one dealer does not drop (or drops proportionally less) than

for other dealers. Finally, the broker can actively test a dealer by launching a fake click-fraud attack from fake clients, and ensuring the dealer blocks them as directed.

A5: A particularly sneaky attack aimed at learning which users send view or click reports for a given publisher (or advertiser) is as follows. The dealer first launches a click-fraud attack on the given publisher (or advertiser). The broker identifies the attack. When a user sends a legitimate report for that publisher (or advertiser), the broker mistakenly suspects the report as fraudulent and asks the dealer to block the client. The dealer can now infer that the encrypted report it proxied must have matched the attack signature it helped create.

D5: First note that this attack applies only in the scenario where there are no other click-fraud attacks taking place other than the one controlled by the dealer (and the dealer somehow knows this). As part of the Privad protocol (Figure 5), however, the dealer does not learn how many attacks are taking place (even if there is only one ongoing attack), or which publishers or advertisers are under attack, or which attack the client was implicated in. Thus there is too much noise for the dealer to reach any conclusions about implicated clients.

7.4.3 Attacker at Broker

A6: The broker attempts to link multiple messages from the same user using passive or active approaches.

D6: We are only concerned with subscribe and reports messages since the dealer mixes auction requests. Privad messages do not contain any PII, unique identifier, or sequence number. The monitor ensures the per-subscription symmetric keys are unique and random. Additionally, the monitor disrupts timing based correlation, for instance by staggering bursts of messages (e.g. when the client starts up, or views a website with many adboxes). Altogether these defenses prevent the broker from linking two subscriptions, or two reports from the same user.

The broker may attempt to link a report with a subscription. The only way to do this is by publishing an ad with a unique ad ID, and waiting for a report with that ID. Privacy advocates can detect this by running honeyfarms of identical clients and ensuring ad IDs are repeated.

A7: During the GSP auction mechanism the broker attempts to link two ads published to the same client through different pub-sub subscriptions, thereby effectively linking two subscriptions.

D7: The property of the mix constructed at the dealer is such that tuples from the same client but for ads on different pub-sub channels are indistinguishable from tuples from two different clients each subscribed to one of the channels. The pub-sub protocol provides the same property. Thus the broker doesn't learn anything new from the auction protocol.

Note the broker can obviously link which ads it sent for the same subscription, but cannot determine which of them actually matched the user. This is because the client submits all ads received on a channel for auction whether or not it matched the user (enforced by the monitor); bogus user scores for non-matching ads prevents the broker from distinguishing between the two.

A8: The broker masquerades as a dealer and hijacks the client's messages thus learning the client's IP address. Possible methods of hijacking the traffic may include subverting DNS or BGP.

D8: The solution is to require Transport Layer Security (TLS) between client and dealer, and to use a trusted certificate authority. The reference monitor can insure that this is done correctly.

A9: The broker creates a channel with a large enough number of attributes that an individual user is uniquely defined. When that user joins the channel, the broker knows that a user with those attributes exists. This could be done for instance to discover the whereabouts of a known person or to discover additional attributes of a known person. For instance, if n attributes are known to uniquely define the person, then any additional attributes associated with a joined channel can be discovered.

D9: It is precisely for this reason that pub-sub channels definitions are static, well-known, and public (Section 3.1). Privacy advocates can look at channel definitions and ensure they meet a minimum expected anonymity set size. Additionally, the monitor can filter out channel definitions when the attributes for that channel exceed some set threshold.

Similar restrictions apply to the set of profile attributes an ad can target, with one difference. In the context of second-price auctions, the broker needs to necessarily link adjacent ads. Thus the monitor needs to enforce that the sum of attributes of the two ads involved in a click-report is below the threshold.

Note the ability to link two ads applies only to clicks. View reports do not contain second price information since otherwise a page with many ads would allow the broker to link each consecutive pair of ads, and therefore a whole chain of ads. While the same problem exists if the user were to click on the whole chain of ads, since clicks are rare this is not a big concern.

8 Related Work

There is surprising little past work on the design of private advertising systems, and what work there is tends to focus on isolated problems rather than a complete system like Privad. This related work section focuses only on systems that target private advertising per se, and mainly concentrates on the privacy aspects of those systems. In particular, we look at Juels [19], Adnostic [29], and Nurikabe [23].

Juels by far predates the other work cited here, and indeed is contemporary with the first examples of the modern advertising model (i.e. keyword-based bidding). As such, Juels focuses on the private distribution of ads and does not consider other aspects such as view-and-click reporting or auctions. Privad's dissemination model is similar to Juels' in that a client requests relevant ads which are then delivered. Indeed, Juels' trust model is stronger than Privad's. Juels proposes a full mixnet between client and broker, thus effectively overcoming collusion. We believe this trust model is overkill, and that his system pays for this both in terms of efficiency and in the mixnet's inability to aid the broker in click fraud.

Like Juels and Privad, Adnostic also proposes client-side software that profiles and protects user privacy. When a user visits a webpage containing an adbox, the URL of the webpage is sent to the broker as is done today. The broker selects a group of ads that fit well with the ad page (they recommend 30), and sends all of them to the client. The client then selects the most appropriate ad to show the user. The novel aspect of Adnostic is how to report which ad was viewed without revealing this to the broker. Adnostic uses homomorphic encryption and efficient zero-knowledge proofs to allow the broker to reliably add up the number of views for each ad without knowing the results (which remain encrypted). Instead, they send the results to a trusted third-party which decrypts them and returns the totals. By contrast to views, Adnostic treats clicks the same as current ad networks: the client reports clicks directly to the broker.

The privacy model proposed by Adnostic is much weaker than that of Privad. Privad considers users' web browsing behavior and click behavior to be private, Adnostic does not. Indeed, we would argue that the knowledge that Adnostic provides to the broker allows it to very effectively profile the user. A user's web browsing behavior says a lot about the user interests and many demographics. Knowledge of which ads a user has clicked on, and the demographics to which that ad was targeted, allow the broker to even more effectively profile the user. Finally, the user's IP address provides location demographics and effectively allows the broker to identify the user. Adnostic's trust model for the broker is basically honest-and-*not*-curious. If that is the case, then today's advertising model should be just fine.

Nurikabe also proposes client-side software that profiles the user and keeps the profile secret. With Nurikabe, the full set of ads are downloaded into the client. The client shows ads as appropriate. Before clicking any ads, the client requests a small number of click tokens from the broker. These tokens contain a blind signature, thus allowing the tokens to later be validated at the broker without the broker knowing who it previously gave the token to. The user clicks on an ad, the click report is sent to the advertiser along with the token. The advertiser sends the token to the broker, who validates it, and this validation is returned to the client via the advertiser.

Nurikabe has an interesting privacy model. They argue that, since the advertiser anyway is going to see the click, there is no loss of privacy by having the advertiser proxy the click token. By taking this position, Nurikabe avoids the need for a separate dealer. Our problem with this approach is that Nurikabe basically gives up on the problem of privacy from the advertiser altogether. It cannot report views without exposing this to the advertiser, thus reducing user privacy from the advertiser even more than today. View reporting is important, in part because it allows the advertiser to compute the CTR and know how well its ad campaign is going. Nurikabe also gives up any visibility into click fraud. Nurikabe mitigates click fraud only by rate limiting the tokens it gives to every user. As a result, the attacker need only Sybil itself behind a botnet and solve CAPTCHAs to launch a massive click-fraud attack which cannot be defended. Finally, in [13] the authors find through ad measurements that there are simply far too many ads (with too much churn) to be able to distribute them all to all clients.

Some aspects of Privad have previously been explored in [13, 15]. The seed idea behind Privad was planted in [15], a short paper revisiting the economic case for advertising agents on the endhost (i.e., distinguishing "adware" from "badware"), which presents a rough sketch of privacy-aware click reporting. In [13] we use measurement data to guide our design and explore the feasibility of building such a system. This paper presents the resulting detailed design, experimental evaluation, and security analysis of a full advertising system.

9 Summary and Future Directions

This paper describes a practical private advertising system, Privad, which attempts to provide substantially better privacy while still fitting into today's advertising business model. We have designs and detailed privacy analysis for all major components: ad delivery and reporting, click fraud defense, advertiser auctions, user profiling, and optimizations for scalability.

We are actively working on getting a better understanding of a number of Privad components. Foremost among these are how best to do profiling, how best to run auctions, the bait approach to click-fraud, and privacy from the advertiser. Another important problem is how to allow brokers and advertisers to gather rich statistical information about user behavior in a privacy-preserving way. Towards this end, we are looking at distributed forms of differential privacy. We are also working with application developers to deploy at Internet scale to give researchers a platform for experimenting with real users and advertisements.

Besides pursuing the technical aspects of Privad, we have discussed Privad with a number of privacy advocates and policy makers, and have applied for a Europrise privacy seal. We hope that Privad and other recently proposed private advertising systems spur a rich debate among researchers and privacy advocates as to the best ways to do private advertising, the pros and cons of the various systems, and how best to move private advertising forward in society.

References

[1] Amazon Mechanical Turk. http://www.mturk.com.

[2] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2), Sept. 2010. http://aws.amazon.com/ec2/.

[3] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-Resource Routing Attacks Against Tor. In *Proceedings of the 2007 Workshop on Privacy in the Electronic Society (WPES)*, Alexandria, VA, Oct. 2007.

[4] A. Chen. GCreep: Google Engineer Stalked Teens, Spied on Chats. Sept. 2010. http://gawker.com/5637234.

[5] J. Chester, S. Grant, J. Kelsey, J. Simpson, L. Tien, M. Ngo, B. Givens, E. Hendricks, A. Fazlullah, and P. Dixon. Letter to the House Committee on Energy and Commerce. http://tinyurl.com/y85h98g, Sept. 2009.

[6] R. Dingledine, N. Mathewson, and P. Syverson. TOR: The Second-Generation Onion Router. In *Proceesings of USENIX Security Symposium '04*.

[7] J. R. Douceur. The Sybil Attack. In *Proceedings of IPTPS '02*.

[8] B. Edelman, M. Benjamin, and M. Schwarz. Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords. *American Economic Review*, 97(1):242–259, Mar. 2007.

[9] J. Elson, J. R. Douceur, J. Howell, and J. Saul. Asirra: A CAPTCHA that Exploits Interest-Aligned Manual Image Categorization. In *Proceedings of CCS '07*.

[10] Europrise. European Privacy Seal DE-080006p. http://tinyurl.com/2dckmpx.

[11] G. Gross. FTC Sticks With Online Advertising Self-regulation. *IDG News Service*, Feb. 2009.

[12] S. Guha, B. Cheng, and P. Francis. Challenges in Measuring Online Advertising Systems. In *Proceedings of IMC '10*.

[13] S. Guha, A. Reznichenko, K. Tang, H. Haddadi, and P. Francis. Serving Ads from localhost for Performance, Privacy, and Profit. In *Proceedings of HotNets '09*.

[14] H. Haddadi. Fighting Online Click-Fraud Using Bluff Ads. *SIGCOMM CCR*, 40(2):22–25, Apr. 2010.

[15] H. Haddadi, S. Guha, and P. Francis. Not All Adware is Badware : Towards Privacy-Aware Advertising. In *Proceedings of 9th IFIP conference on e-Business, e-Services, and e-Society*, Nancy, France, Sept. 2009.

[16] I. Hickson. The WebSocket API. http://dev.w3.org/html5/websockets/.

[17] N. Hopper, J. Langford, and L. V. Ahn. Provably Secure Steganography. In *Proceedings of Crypto '02*.

[18] A. Jesdanun. Ad Targeting Based on ISP Tracking Now in Doubt. *Associated Press*, Sept. 2008.

[19] A. Juels. Targeted Advertising ... And Privacy Too. In *Proceedings of the 2001 Conference on Topics in Cryptology*, pages 408–424, London, UK, 2001.

[20] A. Juels, S. Stamm, and M. Jakobsson. Combating Click Fraud via Premium Clicks. In *Proceesings of USENIX Security Symposium '07*, pages 1–10.

[21] M. Kounavis, X. Kang, K. Grewal, M. Eszenyi, S. Gueron, and D. Durham. Encrypting the Internet. In *Proceesings of SIGCOMM '10*.

[22] B. Krishnamurthy and C. E. Wills. Cat and Mouse: Content Delivery Tradeoffs in Web Access. In *Proceedings of WWW '06*.

[23] D. Levin, B. Bhattacharjee, J. R. Douceur, J. R. Lorch, J. Mickens, and T. Moscibroda. Nurikabe: Private yet Accountable Targeted Advertising. Under submission. Contact johndo@microsoft.com for copy, 2009.

[24] P. Loscocco and S. Smalley. Integrating Flexible Support for Security Policies into the Linux Operating System. In *Proceedings of the 2001 USENIX Annual Technical Conference*, Boston, MA, June 2001.

[25] I. S. Moskowitz and M. H. Kang. Covert Channels - Here to Stay? In *Proceedings of the 9th Annual Conference on Computer Assurance (COMPASS)*, pages 235–243, Gaithersburg, MD, July 1994.

[26] K. Park, V. S. Pai, K.-W. Lee, and S. Calo. Securing Web Service by Automatic Robot Detection. In *Proceesings of USENIX Annual Technical Conference '06*.

[27] A. Pfitzmann and M. Köhntopp. Anonymity, Unobservability, and Pseudonymity — A Proposal for Terminology. *Designing Privacy Enhancing Technologies*, 2001.

[28] B. Stone. Google Says It Inadvertently Collected Personal Data. *The New York Times*, May 2010. http://tinyurl.com/2946cql.

[29] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy Preserving Targeted Advertising. In *Proceedings of NDSS '10*.

[30] Trusted Computing Group. TPM Specification Version 1.2. http://www.trustedcomputinggroup.org/.

[31] X. Wang, S. Chen, and S. Jajodia. Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet. In *Proceedings of CCS '05*.

[32] E. Wyatt. U.S. Court Curbs F.C.C. Authority on Web Traffic. *The New York Times*, Apr. 2010. http://tinyurl.com/yamowhd.

[33] B. Yee, D. Sehr, G. Dardyk, J. B. Chen, R. Muth, T. Ormandy, S. Okasaka, N. Narula, , and N. Fullagar. Native Client: A Sandbox for Portable, Untrusted x86 Native Code. In *Proceedings of Oakland '09*.