Private-by-Design Advertising Meets the Real World

Alexey Reznichenko Paul Francis
Max Planck Institute for Software Systems (MPI-SWS), Germany
{areznich,francis}@mpi-sws.org

ABSTRACT

There are a number of designs for an online advertising system that allow for behavioral targeting without revealing user online behavior or user interest profiles to the ad network. However, none of the proposed designs have been deployed in real-life settings. We present an effort to fill this gap by building and evaluating a fully functional prototype of a practical privacy-preserving ad system at a reasonably large scale. With more than 13K opted-in users, our system was in operation for over two months serving an average of 4800 active users daily. During the last month alone, we registered 790K ad views, 417 clicks, and even a small number of product purchases. In addition, our prototype is equipped with a differentially private data collection mechanism, which we used as the primary means for gathering experimental data. The data we collected show, for example, that our system obtained click-through rates comparable with those for Google display ads. In this paper, we describe our first-hand experience and lessons learned in running the first fully operational "private-by-design" behavioral advertising and analytics system.

Categories and Subject Descriptors

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

Keywords

online advertising; privacy-by-design; prototype; deployment; experiments

1. INTRODUCTION

Several research projects have proposed alternative "private-by-design" advertising models in an attempt to reconcile behavioral targeting and user privacy (Adnostic [21], RePriv [12], Privad [15], MobiAd [16], and PiCoDa [18]). Each of the proposed systems makes largely unsubstantiated claims that they provide both good privacy and high utility at reasonable cost. The goal of this paper is to test that claim, at least for the Privad system, by building, deploying, and evaluating a fully functional prototype. Our deployment delivered functional ads in the sense that the ads were targeted to user interests, displayed on publisher webpages, linked to real shopping websites, and in fact led to actual purchases. Side-by-side with Privad, we also deployed a distributed differentially-private user analytics system, PDDP [7], that served as our primary means of gathering experimental data.

By bundling our system with a popular Firefox addon, we deployed it to over 13K opted-in users. Over a period of two months, the system was used daily by over 4800 active users on average, with more than 2000 users online at peak. In October 2013 alone, our backend received 1.1M ad requests and generated 9.5M ads. During that time, we registered 790K ads views, 417 ad clicks, and 4 product purchases.

While minuscule by commercial standards, our deployment was big enough to gain real insights into the Privad and PDDP designs, both positive and negative. Perhaps the most surprising of the positive insights is that our PDDP results indicate that the system's click-through rate (CTR) compares well with Google display ads CTR (§ 5.2). This is especially impressive given that ad generation in our system was fully automated, in contrast to Google where ads are designed by hand and fine-tuned over time.

Among the negative insights, our experience suggests that the privacy loss predicted by differential privacy is prohibitively pessimistic. Based on the relatively small number of queries we made to our system (159 distinct queries generating 790K answers from 9395 unique clients), differential privacy's worst-case stance would suggest that a *substantial* proportion of our user base could have experienced privacy loss. In reality, not only was no individual user information leaked through PDDP, even if we had generated malicious queries based on auxiliary information, at best we could have learned one or two things about one or two users (§ 5.4). This large discrepancy between the differential private model of privacy loss, and actual privacy loss, needs somehow to be addressed by the privacy research community.

Our experience also exposed several shortcomings in the Privad privacy mechanisms (§ 3.7). First, Privad did not adequately consider user profiling mechanisms. The practical profiling mechanism we used required additional privacy protections (white-listing of user search terms and scraped product names). Second, Privad didn't adequately take into account issues that arise when the user base is relatively small, such as potential user fingerprinting through reported timestamps.

CCS'14, November 3–7, 2014, Scottsdale, Arizona, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2957-6/14/11 ...$15.00.
http://dx.doi.org/10.1145/2660267.2660305.

Nevertheless, the high-level take-away is generally positive: we found no fundamental problems, either with privacy or with utility. We believe that ultimately a commercial deployment is needed to prove the viability of these private-by-design advertising and analytics technologies. In particular, with our limited resources, we were not able to integrate with existing auction systems or ad exchanges, nor gain any experience with click-fraud detection.

Altogether, this paper makes the following contributions: 1) It presents the design and analysis of the first deployed fully-functional private-by-design ad system. While it does find (and fix) a number of weaknesses in the previous design, it also finds nothing to disprove the viability of private-by-design behavioral advertising. 2) It presents our experience in running a relatively large-scale user-centric research experiment with differentially private analytics as the primary means of gathering system and user data. It shows that such an approach is feasible. 3) It analyzes the practical implications of the privacy deficits accumulated as a result of differentially private data collection. It concludes that the differentially private model is overly pessimistic for understanding privacy loss in our deployment, and that the noise-adding mechanism alone is too weak to be practical for long-running analytics.

2. BACKGROUND

In this section, we give a broad overview of the main system components in the Privad architecture and describe the privacy guarantees provided by the system. We then outline the design of the PDDP system that leverages the same underlying architecture to enable differentially private data collection in distributed settings.

2.1 Basic Privad Architecture

A number of components in the Privad architecture play the same role as they do in today's ad deployments. These include *users*, *publishers*, *advertisers*, and a *broker* (ad network): users browse publisher webpages, and advertisers provide ads to brokers for display on those webpages. Privad defines two new components, the *client* and the *dealer*, and substantially modifies the role of the broker: user profiling and ad serving are delegated to the client software, which runs on the user's device, rather than in the cloud (i.e., at the broker) as it is done in today's deployments. The client monitors user behavior (i.e., the user's searching, browsing, purchases, and so on) and over time builds up a user profile. It then uses this behavioral profile to privately fetch ads from the broker and locally decide which ads should be presented to the user. The client and the broker are separated by a proxy-like dealer, which strips away the network layer address of all clients' messages. The dealer can also coordinate with the broker to identify and discount fraudulent clicks as well as block clients suspected of click-fraud.

The broker in Privad is assumed to be honest-but-curious. While this may be close to reality (brokers like Google can generally be trusted to do what they claim they are doing), the system design nevertheless strives to prevent the broker from obtaining high-value information through simple but hard-to-detect cheating.

The fundamental design principle in Privad is that private information about each user is kept on that user's computer, not in the cloud [15]. In a sense, users are still tracked. However, the tracking is done by a software agent running on the user's machine, and the information it gathers (the user profile) never leaves the user's machine. The challenge is to utilize the user profile to deliver targeted ad content while revealing the minimum amount of information from the user profile. Concretely, the privacy goals of the Privad system are formulated as follows [14]:

- Anonymity: the broker cannot associate any unit of learned information with any user Personally Identifiable Information (including network address), and
- Unlinkability: the broker cannot associate separate units of learned information with a single (anonymous) client. This prevents the broker from building up a user profile, and then associating it with a known user using externally gathered knowledge.

Figure 1 illustrates the basic Privad architecture and message exchanges between principal components.

User profiling software runs at the client. This software monitors the user's activity, such as search terms, browsing behavior, purchases made, and so on. When the profiling software identifies a user interest, it anonymously requests a set of ads for the given interest category *type* (i.e., ads for products or services matching the user interest). The request must be generic enough that a substantial set of clients can have legitimately made the request. A set of matching ads, each with an identifier A and associated *targeting* information, are transmitted to the client. The client software then filters out ads that do not match the profile and locally stores the rest. When an adbox is presented to the client, for instance on a webpage, the client selects among the stored ads those that best match the user profile, and inserts them in the adbox displayed to the user. The client anonymously reports the view (and click) for the ad A together with the webpage *URL*.

Privad relies on two communication channels between the client and the broker, one for ad delivery, the other for view and click reporting. All message exchanges follow the same basic protocol: the client's request is encrypted with the broker's public key and contains a one-off symmetric key generated by the client, which is later used to encrypt the broker's response (e.g., the stream of ads sent to a client). Since the encryption is opaque for the dealer, it blindly forwards the messages without learning anything about the clients. As long as the broker and the dealer do not collude the system can offer privacy guarantees: the dealer prevents the broker from learning the client's identity or from linking separate messages from the same client.

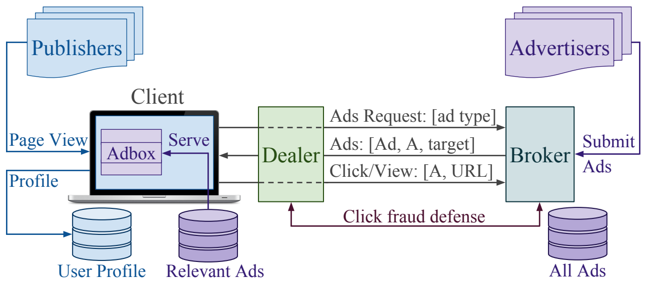


Figure 1: The Privad architecture. [x] denotes encryption of x.

2.2 PDDP Overview

Privad is carefully designed to provide only the minimum information needed by the broker, advertisers, and publishers to run the ad business: requests for ads and anonymous reports of clicks and views (i.e., which clicks and views occurred on which ads at which publishers). This minimum information, however, is not sufficient if the goal is to get deeper insight into user behavior and system performance. For instance, key players may want to know more, in the aggregate, about what activity leads the profiler to detect an interest in the first place. They may wish to know the level of interest, or correlations between interests. With centralized (non-private) tracking, all required information is available locally, and can simply be mined. With Privad, this information is all tucked away on user computers, which precludes broad statistical analysis of user data. Moreover, once Privad is deployed, it would be virtually impossible for the system designers themselves to debug the system without infringing on user privacy. To address this issue, Privad needs to provide support for privacy-preserving statistical queries over distributed user data.

One approach to supporting privacy-preserving statistical queries is to add noise to the answers of queries, in such a way that the privacy of individual users is protected. An instance of this approach popular in the research community is *differential privacy (DP)* [10, 11]. Specifically, DP adds noise to the answers for queries to statistical databases so that the querying system cannot infer the presence or absence of a single user or a set of users. DP provides a provable foundation for measuring privacy loss regardless of what information an adversary may possess.

The traditional deployment model for DP assumes a centralized database. The system operating the database is trusted with its content, and is also trusted to add noise to the information released from the database. The non-tracking advertising scenario is different in several respects. First, there is no trusted centralized database; individual clients maintain their own data. Second, the information is distributed among potentially millions of clients. Therefore, the non-tracking advertising settings call for a practical mechanism that provides some form of *distributed differential privacy.*

As it turns out, the existence of the dealer in Privad, trusted not to collude with other components of the system, can be leveraged to accommodate the Practical Distributed Differential Privacy (PDDP) system [7]. Figure 2 shows how PDDP can be deployed on top of the Privad dealer. In Privad, an analyst (e.g., a broker, an advertiser), who wishes to make statistical queries over some number of Privad clients, can formulate a query and transmit it to the dealer, which in turn forwards it to the required number of clients. Each query comes with a number of buckets that specify possible answer ranges. A client locally executes the query, and for each bucket it produces a binary value indicating whether the query result fell within the range of that bucket. Then, the resulting bit vector is encrypted with the analyst's public key (using Goldwasser-Micali bit-cryptosystem [13]) and uploaded to the dealer. Meanwhile, the dealer and clients, using the XOR homomorphic property of the GM cryptosystem, *collaboratively and blindly* generate noisy answers that mimic a number of additional client responses to produce the required amount of differentially private noise. Finally, the dealer mixes received client answers with noisy answers,

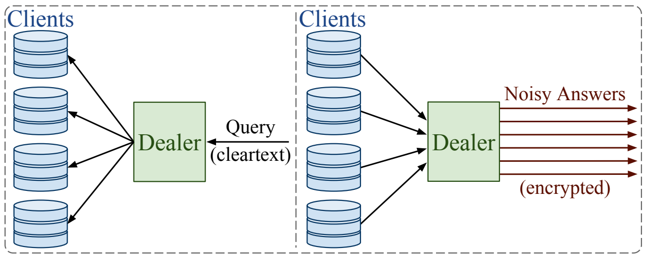


Figure 2: DP in the context of non-tracking advertising

and forwards everything together to the analyst, who then decrypts the received answers and computes the statistical result under the differentially private guarantee.

A major issue with DP in practice is that systematically repeated queries can be used to eliminate the noise and reveal the true answer. Traditional DP systems deal with this through the notion of a *budget*. Each query deducts from the budget, and when the budget is spent, the additional queries are simply not allowed. However, this approach is not practical in the advertising context, which requires longitudinal analytics. Rather than setting a hard limit on the cumulative privacy loss, PDDP treats it as an ongoing measure referred to as the *privacy deficit*. The notion of the *privacy deficit* makes it possible to address a number of open issues. First, it provides a quantitative measure of the privacy cost incurred as a result of a meaningful statistical analysis of the user population. Second, as described in Section 5, the accumulated deficit can be analyzed under a worst-case scenario to determine whether it theoretically allows a malicious analyst to discover a number of sensitive user attributes. Indeed one question we address in this study is "How far from actual reality is differential privacy's worst-case model?"

3. PROTOTYPE DETAILS

While Privad establishes a basic private-by-design architecture described in the previous section, to put together a fully functional ad system prototype we had to fill in a number of gaps. Due to the experimental nature and small scale of our system, we cannot work directly with advertisers or ad networks. Instead, we use product information from major shopping engines to as a proxy for creating Privad ads. Given such product-oriented ads, for profiling and targeting we focus exclusively on the user purchasing intent. We rewrite Google ad iframe requests and repurpose resulting adboxes to render Privad ads, which prevents exposing users to more ads than they would normally see.

In the rest of this section, we describe in detail the challenges we tackled while building a private-by-design ad system prototype without support from the ad industry. We also discuss ways in which the private system design evolved to meet practical concerns. Then, we report privacy issues identified and addressed along the way. Finally, we describe implementation details specific to our prototype.

3.1 User Profiling

Since we can only generate product-related ads, our main goal with respect to user profiling is to identify and capitalize on the user transactional (purchasing) intent [17]. Therefore, we focus on two main signals: user browsing activity on shopping websites (product-based targeting) and

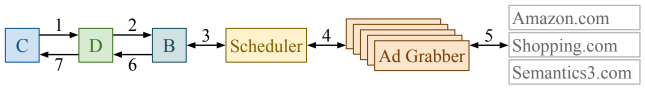
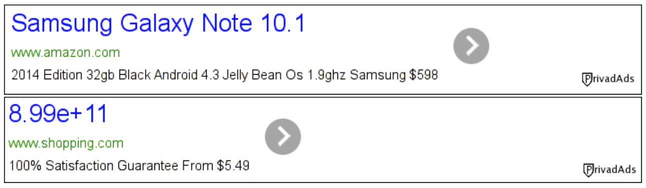


Figure 3: Ad generation pipeline. C is client, D is dealer, B is broker


Samsung Galaxy Note 10.1
www.amazon.com
2014 Edition 32gb Black Android 4.3 Jelly Bean Os 1.9ghz Samsung $598
PrivadAds

8.99e+11
www.shopping.com
100% Satisfaction Guarantee From $5.49
PrivadAds

Figure 4: Examples of auto-generated Privad ads

product-related searches on major search and shopping engines (search-based targeting). Towards this end, we compiled a whitelist of shopping websites containing almost 14K entries by crawling retailers on Shopping.com that also appear in Alexa's top 1M [1]. The whitelist also includes Alexa's top 500 websites for a number of top-level shopping categories, as well as the 5 largest search engines. Additionally, we built a dictionary of product-related terms by crawling a random set of ca. 80 million products offered on Google Products, Amazon and Shopping.com. From the collected set of 10M terms appearing in product titles, we removed terms with fewer than 100 occurrences, resulting in 180K whitelisted keywords.

Each time a user issues a search query on one of the whitelisted websites, Privad client captures the query, extracts search terms and filters stemmed keywords through the product-related dictionary. The resulting list of keywords is then used to establish an ad channel. Additionally, the identified keywords are cached and used to deduplicate future product searches.

We profile user shopping activity by monitoring browsing behavior on whitelisted websites and applying customized scrapers to identify specific products the user is interested in. The fundamental challenge of this approach is the effort required to build specialized product scrapers for the majority of popular online retailers. In our experimental prototype, we sidestep this challenge by leveraging scraping functionality developed by InvisibleHand [4]. InvisibleHand tries to identify a product the user is browsing for, and then displays a notification if there are better deals available for the product. At the moment, InvisibleHand scrapers identify products on 670 shopping websites. As described in Section 3.6, neither the whitelists nor the scraping functionality is hard-coded in the client. Instead, clients periodically check with the broker and download updated lists of scrapers and whitelisted domain names as soon as they become available.

3.2 Ad Generation

The experimental nature of our system dictates that we neither work with any commercial advertising companies nor generate any revenue by displaying ads. Instead, we create mock-up[1] ads using product listings from three major shopping engines: Amazon.com, Shopping.com and Semantics3.com.[2]

The series of steps performed to generate Privad ads is shown in Figure 3. Once the Privad client detects a new product or product-related search, it uses product title or whitelisted search keywords to establish a new interest channel and request ads from this channel. We do not rely on any predefined interest hierarchy to map profiling information to channels. On the contrary, the interest channels are generated at runtime and are fully defined by the associated product information. As such, we distinguish between two channel types according to targeting parameters (product and search targeted channels). Targeting information associated with each channel is used to request a number of ads from the broker in an anonymous, privacy-preserving manner. The ad request (step 1 in Figure 3) is relayed by the dealer to hide the client's identity. The encryption mechanism described in Section 3.4 is used to prevent the dealer from eavesdropping.

The dealer batches requests from multiple clients into a single RPC request, which is uploaded to the broker once every 30 seconds (step 2). Upon receiving an ad request, the broker forwards targeting information to the *ad grabbing* service (in step 3), which then uses it to make a product search on one of the three shopping engines (steps 4 and 5). Up to 20 most relevant product offers from the result set are converted into textual ads and returned to the broker. The conversion is straightforward and consists of removing stop words and excessive punctuations from the product title and description, distributing the remaining terms together with a price tag over ad head and body (25 and 70 characters long), and adding a short display URL. Most of the auto-generated ads produced this way are intelligible and look almost indistinguishable from AdSense ads. However, in some cases products-to-ads conversion fails to produce meaningful output (see examples in Figure 4). Finally, the broker bundles the resulting set of text ads into a single ad channel and ships it back to the dealer (step 6). The ad channel is stored at the dealer until it is eventually retrieved by the client (step 7).

Shopping engines proved to be the major bottleneck in the ad generation pipeline (with requests to Shopping.com taking on the order of several seconds). Additionally, they tend to impose a limit on the number of allowed search requests per IP. To address this scalability challenge we replicated the ad grabbing service on a number of machines and placed a simple round-robin load-balancer (Scheduler in Figure 3) in between the Ad Grabbers and the broker.

3.3 Ad Selection and Placement

Since we lack real publishers, we render Privad ads in the existing Google AdSense adboxes (i.e., adboxes that contain contextual ads and appear on publisher websites). While we modify Google ad frame requests to display more textual advertising (as opposed to flash and image ads), we avoid exposing users to more ads than they would normally see. Google allows publishers to specify style parameters for textual ads so that ads have the same look and feel as the publisher website. Instead of fully re-writing adbox html

[1]The generated ads look like legitimate Google ads and link to real products. We call them 'mock-up' only because they are not handcrafted.

[2]While we initially started out with Google.com/shopping as our third product provider, over the course of the prototype development Google's Search API for Shopping was deprecated and then eventually sunset.

code, we leverage this functionality by identifying and surgically replacing only relevant ad content (head, body, click URL, etc.). As a result, apart from a small logo indicating a Privad adbox, Privad ads look almost indistinguishable from AdSense text ads. However, while preserving ad style preferences, this approach is ad hoc in nature and depends on cues in the html code characteristic to various ad elements; once html code is modified our ad serving modules might fail to render Privad ads.[3]

Our client implementation allows us to experiment with different placement strategies: control mode (only Google ads), full-on Privad mode (only Privad ads), mixed mode (a mixture of Privad and Google ads in the same multi-slot adbox) and random mode (uniform distribution of adboxes filled with either Privad or Google ads). Additionally, our client respects user preferences and does not request or display any ads in Private Browsing Mode, since uploading ad requests or view/click reports in PBM would clearly violate user expectations. Also, it does not display Privad ads if it detects any adblocking mechanisms (browser addons or DNS-based blocking).

Unfortunately, we lack bid information to run a full-fledged second price auction. Instead, we conjecture that the click probability, and hence the user score [19], is inversely related to the amount of time passed since an interest was detected. In other words, the *click probability* on an ad from a particular channel *decreases over time*. To verify this observation, the Privad client allows experimentation with three ranking mechanisms according to the age of an ad channel: most recent first, uniform random and binomial (pick the most recent with probability 1/2, second most recent with 1/4 and so on).

3.4 Message Exchange

Following the original Pub-Sub model [14], the dealer uses an asynchronous relay protocol to forward messages between the clients and the broker. Each new Privad client is bootstrapped by requesting a unique client id from the dealer. This Uid then is sent to the dealer alongside the encrypted message payload. For every incoming request that requires an explicit reply from the broker, the dealer generates a unique request ID (Rid). It replaces Uid with Rid in the client's request and also stores the mapping between them. Finally, multiple client requests are batched together and uploaded to the broker at regular intervals. The broker then attaches the Rid to every response it sends back, which the dealer uses to look up the intended client and save the broker's response tagged with the client's Uid. A client uses its Uid to periodically poll the dealer for any new messages from the broker.

As opposed to the original Privad design, which relies on a predefined set of channels to map user attributes to ads, in our prototype channels are established on the fly in response to an ad request (as long as the result set of the corresponding product search is non-empty). This allowed us to reduce the original Pub-Sub ad dissemination mechanism to an asynchronous request-response, which operates as follows. Client requests take the form ($E_{PK_{broker}}(K_{shared})$, $E_{K_{shared}}(request)$) where the actual message payload is encrypted with a randomly generated 128-bit AES key K_{shared}, and the symmetric key itself is encrypted with the 1024-bit public RSA key of the broker PK_{broker}. Randomized padding is added to defend against dictionary attacks. The response from the broker is then $E_{K_{shared}}(reply)$ encrypted with the symmetric key from the corresponding request. The request and response messages of this form serve as building blocks for all client-broker communications, which include ads delivery, click and view reporting, as well as the new communication channels required for distributing scrapers, website and keywords dictionaries and experimental configurations (Section 3.6).

[3]Unluckily, as described in Section 4.1, this is exactly what happened during our deployment.

3.5 Privad Implementation

We built a fully functional Privad system. Following the architecture described in Section 2.1, our prototype comprises three principal components: the client, the dealer and the broker. The client is implemented as a 154KB Firefox addon written entirely in JavaScript (8.5K lines of code not counting the pidCrypt library and autogenerated RPC client code). All backend components are implemented in Java, totaling 14K lines of code with the dealer, broker, and ad generation infrastructure taking roughly equal parts. All Privad datatypes and interfaces between system components are defined in 600 lines of Apache Thrift IDL [3], producing 48K and 6K lines of Java and JavaScript code respectively.

We chose to implement the client as a browser addon to enable us to scrape highly-dynamic AJAX web applications, which would have been impossible with a standalone daemon or local browser proxy. Concerned with Javascript performance for cryptographic operations, we delegated all CPU-intensive processing to two independent web workers (one responsible for Privad-related functionality, the other for PDDP). These background Javascript threads have no access to the DOM and communicate with the main browser thread via asynchronous message passing. Also, they serve as the single gateway between the client and the dealer. By outsourcing cryptographic operations, data serialization and network communication to background workers, we ensured that there was no negative impact on the user's browsing experience.

All client-dealer communication is performed over HTTP to accommodate clients behind firewalls and proxies. We use JSON since it's the only format currently supported by the JavaScript Thrift library. While the dealer is written as a Jetty [20] server handler, other backend components are built on top of Thrift servers and communicate using binary Thrift format.

3.6 Client Details

In addition to the two communication channels between the client and the broker (ad delivery and view/click reporting) required by vanilla Privad, we also introduced a distribution and update mechanism for product scrapers, shopping websites and product term whitelists. To keep their whitelists and scrapers up-to-date, clients periodically issue and upload a request containing the hash of the currently active whitelist. As long as the hashes of the client's and broker's lists match, the request is ignored by the broker. If hashes do not match (e.g., entries were added or removed from the whitelist), the new whitelist is sent in response.

A similar mechanism is used to disseminate experiment configurations. In addition to the hash of the configuration in place, clients also include their configuration class in the request. This class is randomly selected from 16 available values during the client's first launch. By dividing client's population into 16 groups, we are able to run multiple experiments in parallel. An experiment configuration contains a number of parameters that specify start and end timestamps and regulate ad placement strategy (none, everywhere, mixed, random), channel ranking mode (random, most recent, binomial), targeting mode (product-based, search-based, random) and various channel-related attributes (max channels in place, max channel lifetime, max view opportunities, etc.).

3.7 Practical Privacy Issues

Running a 'real' ad system requires a number of functions not anticipated in Privad's design. Consequently, to build an operational private-by-design system, we had to address several practical privacy-related concerns that arose as a result of the added functionality.

Search terms in ad requests. The original Privad design envisioned that relatively broad product or interest categories would be conveyed in ad requests. In practice, however, we had to use search terms and product names derived from potentially error-prone web page scraping. To mitigate possible privacy loss through these search terms and product names, we implemented the whitelist described in Section 3.1. In spite of this, in rare cases, it may be possible to identify users through the ad request. This is an unanticipated problem that needs further consideration.

Timestamps in ad requests and reports. In order to select relevant configuration parameters when serving a client's ad request, the broker needs to know both the client's configuration class and the timestamp at which the request was made. Moreover, view/click/conversion reports are also timestamped, which allows us to find the delay between the interest detection and the subsequent view/click/conversion events as well as the temporal distribution of these events. However, revealing unobscured client timestamps constitutes a major privacy leak. First, the broker may exploit these timestamps and try to fingerprint clients based on their clock skew. Second, sending timestamps in the client's local timezone breaks channel unlinkability (e.g., when there are very few online users in a particular timezone sending view/click reports for ads from different channels).

We prevent this privacy leakage by uploading client timestamps converted to the same timezone (UTC) across all clients. To be able to compute event distribution over time, we add a timestamp in the client's timezone and an event subtype (ad request, view, click, conversion) as the meta-info of the encrypted message uploaded to the dealer, and store it there without forwarding it to the broker. To hide potential clock skew, we currently use timestamp granularity of 5 minutes. Additionally, it's possible to add some amount of noise to the timestamps, introduce longer (currently only 30 seconds) upload cycles and jitter (i.e., delay random messages for several upload cycles) at the dealer.

Publisher info in view reports. Using the publisher domain from the view reports, the broker can track all websites where an ad (or ads from the same channel) were displayed, as long as ad and channel ids are unique across all clients. However, in a commercial Privad deployment the broker must not generate unique ids for subscriptions to the same channel, and it should be easy to detect when this assumption is violated. The problem will come up only when there are very few users subscribed to a channel. Reporting publisher domain together with a view timestamp also breaks channel unlinkability (e.g., when there are multiple adboxes on a page filled with ads from different channels). In our prototype, only click and conversion reports contain publisher info. We break the view report into two parts, one containing ad specific information and the other publisher data, and upload them to the broker independently with a random delay.

Adbox id in view reports. To discover which ads are displayed together, a randomly generated adbox id is included in view reports. Unlinkability then can only be ensured, if we populate each adbox exclusively with ads from a single channel.

3.8 PDDP Implementation

Following the design described in [7], we built the PDDP private analytics subsystem by retrofitting Privad components with additional PDDP functionality. In our implementation, the broker takes the role of an analyst, the dealer acts as the PDDP proxy, and the Privad addon as a client.

The query processing funnel contains the following steps. First, the broker submits a PDDP query to the dealer. A query includes a number of SQL statements, buckets definitions (ids, and lower and upper bounds), privacy parameter ϵ, and start and end timestamps. Additionally, it can specify the required number of answers and the target configuration class. The dealer verifies that the query does not modify the client database (queries containing keywords like 'create', 'pragma', 'delete', etc. are rejected),[4] and adds it to the list of pending queries.

For every pending query the dealer maintains a set of clients who already uploaded an answer to the query. Clients periodically poll the dealer and retrieve a new (random) PDDP query that they have not yet answered. Upon receiving a query, the client executes it over its local database and produces a list of numerical answers, which it then maps to buckets by assigning a '1' or a '0' to each bucket, depending on whether or not one of the answers fell within the range of the bucket. Then the client encrypts each per-bucket binary value using the broker's Goldwasser-Micali (GM) [13] public key. The resulting set of bucket ids together with encrypted bits make up a PDDP answer that is submitted to the dealer.

After receiving a client's answer, the dealer validates the answer (Jacobi symbol of a valid GM-encrypted value equals to '+1') and stores it locally. Once the dealer collects the required number of answers or the query expires, it adds a number of randomly flipped bits or *coins* to each bucket to ensure differential privacy. Given privacy parameter ϵ and the number of answers c, the minimum number of per-bucket coins required to achieve (ϵ, δ)-differential privacy is $n = \lfloor 64 \ln(2c)/\epsilon^2 \rfloor + 1$ [7]. Finally, the dealer shuffles clients' answers and random coins together and uploads the resulting set of (bucket id, encrypted bit)-pairs together with the value of n to the broker. Upon receiving this message, the broker decrypts and sums up all binary values for each bucket id. It then subtracts $n/2$ from each per-bucket sum to compute a (noisy) per-bucket $count = \sum_{i=1}^{c+n} bit_i - n/2$

[4]However, we allow PDDP queries to store intermediate results as key-value pairs in a dedicated table, which is wiped clean after every query execution.

(i.e., the number of clients that fall within this bucket, under the guarantees of differential privacy).

By answering a PDDP query a client ultimately reveals a bit of private information, which over a large number of trials can potentially allow an attacker to average out noise and discover private user attributes. In other words, each PDDP query has an implicit privacy cost associated with it, which clients pay when they answer the query. Over time, this leads to accumulation of a *privacy deficit* (i.e., privacy loss across all queries). We keep a record of the per-client privacy deficit at the dealer. To err on the conservative side, we make no assumptions about possible correlations between buckets and effectively treat individual buckets as separate queries bundled together. Thus, for every client that contributed an answer the dealer adds $(\epsilon, 1/c)\times$ *number of buckets* to the client's privacy deficit. While this is an overly pessimistic approach to tracking deficit, it allows us to have both overlapping bucket ranges and queries producing multiple results that are mapped to multiple buckets. To further reduce the privacy cost, we implemented PDDP queries keyed on experimental configuration classes to target only relevant groups of users.

Dogfooding PDDP enables us to collect various advertising related metrics that cannot be conveyed through Privad without breaking its privacy guarantees. For example, using PDDP we can find per-user click-through performance of Google AdSense ads and compare it with Privad's. Moreover, using various client-side stats, PDDP allows us to peek beyond simple views and clicks and analyze user engagement with the advertising content.

In addition to storing information related to core Privad functionality (experimental configuration, captured searches and products, ad requests, active ad channels, view and click stats, etc.), we also collect a number of additional metrics. These include performance stats for several types of Google ads (text, banner, flash), user engagement (time spent actively browsing a landing page), browsing session and click-chains (series of visited URLs after an ad click). Our client also captures user's shopping activity (products placed in the shopping cart, purchases made), browsing, and bookmarking behavior. Additionally, we store general user information including geographical location and timezone, OS, language, adblocking addons, as well as overall browser usage. All captured information is stored in a local SQLite database using Firefox's Storage API, thus allowing the PDDP analytics system to query for that information in a differentially private manner.

4. LARGE SCALE DEPLOYMENT

One major challenge in deploying the Privad prototype is incentivizing users to install it. Since Privad does not provide immediate tangible benefits for the end users, the most viable deployment model is bundling with existing freeware applications with a well-established user-base. In this section, we describe our experience in deploying Privad by bundling it with a popular Firefox addon and present various deployment statistics collected by the backend servers.

4.1 Deploying Privad at Scale

We deployed Privad by bundling it with Google Docs Viewer[5] – a Firefox addon that uses Google Docs to render online documents (pdf, doc, ppt, etc.) in the browser without downloading them. We decided to bundle with Google Docs Viewer mainly because the addon is actively supported and extended, and therefore maintains a sizable population of almost 80K daily active users.

[5]No affiliation with Google Inc.

Experimental ethics. Participation in the Privad experiment follows an opt-in model.[6] When users update their addon to the version containing the Privad bundle, they are presented with a participation request dialog and are free to choose to join the experiment or not. The participation request contains a link to a webpage, which provides a comprehensive description of the experiment and informs the users that a fraction of the Google AdSense ads will be replaced with Privad ads during the experiment. Each adbox containing Privad ads is clearly labeled with a distinct PrivadAds icon, which when clicked leads to the experiment homepage. In case an opted-in user is no longer willing to participate in the study, the Privad client provides an easy way to opt-out.

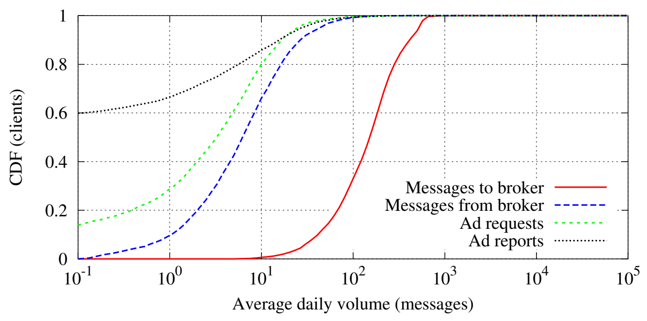
The system was in continuous operation for more than two months in September and October 2013 with a two-week gap during which it did not serve any Privad ads. This hiccup was caused by a major Google AdSense redesign [5], which changed the html code that ad servers produce to display AdSense ads. As a result, Privad's ad rendering modules were no longer functioning properly and we had to push an update to address issues triggered by the new AdSense design.

Overall, 13K users opted into the study[7] and after an initial bootstrapping period the system was used daily by over 4800 users on average, with more than 2000 users online at peak. In October alone, the Privad backend received 1.1M ad requests, generating 960K channels with 9.5M ads in total. We registered 790K ads views, 417 ad clicks and 4 Amazon purchases (including a "Flower Power Hippie" Halloween costume, among others). During that time the dealer served on average 7.9M daily RPC requests, and forwarded 950K messages from clients to the broker and 60K messages in the reverse direction on a daily basis. In terms of the network utilization, this corresponds to 1.2 GB and 115 GB of daily traffic received and sent to clients. On average, the broker processed 32K search-based and 5.5K product-based daily ad requests, generating 280K and 39K ads respectively. At peak, the load on broker reached 286 requests per second.

To measure the communications overhead at the clients we parsed server-logs generated by the dealer in October and selected 6.5K Uids that appear in logs on at least 7 different days. Figure 5 plots the distributions of per-client daily volume of messages exchanged with broker, including ad requests and reports, as well as the daily bandwidth consumption. While detailed information logged by the dealer must not be directly available to the broker, the aggregate stats presented here can safely be made publicly available as part of a monthly operational summary. Surprisingly, we found that the median value for the number of daily ad requests is around 3.2, 11% of the users in our sample never

[6]Mozilla's No Surprises policy requires a opt-in with non-default user action to activate an *"unexpected feature"*, such as Privad.

[7]No pings are sent to the backend before a user has opted in, as a result we do not know the exact opt-in rate. But based on the number of daily Google Docs Viewer users, we estimate it at around 1 in 15.



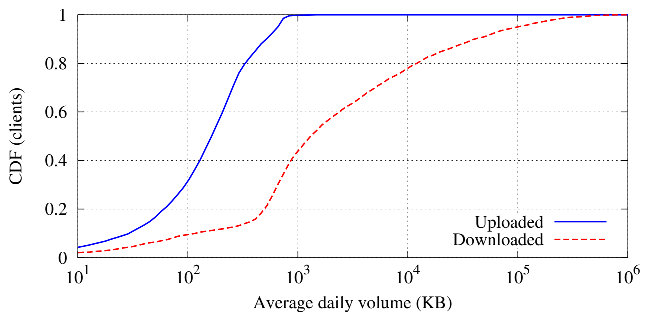


Figure 5: CDFs of per-client daily communications overhead

requested any ads, and almost 60% did not generate any ad views. To uncover the reasons for the observed behavior, we turned to PDDP analysis, as described later in Section 5.1.

4.2 Privad Advertising

In this section, we report various advertising related stats computed using reports collected by the Privad broker (i.e., not with PDDP). In total, 87% of all requests generated a channel (with 89% and 79% for search- and product-based requests respectively), producing on average 9.8 ads per channel. Overall, we calculated Privad CTR at 0.05%. While this value may seem discouragingly low, as we discovered using PDDP analysis (see Section 5.2), in terms of advertising performance Privad ads are comparable with text ads on Google Display Network.

With slightly more than 400 clicks, we did not observe stark variations in CTRs corresponding to different configuration parameters (e.g., placement mode, channel lifetime, channel selection mode, etc.; all produced roughly equal CTRs). However, we found that the CTR for product-targeted ads is 2.6 times higher than that for search-targeted ads (0.12% versus 0.046%). We also found that more than 70% of the Privad clicks were registered in single-slot adboxes (with corresponding CTR of 1.1%), with another 14% in the first position in multi-slot adboxes (where we observed an exponential decrease in CTR in lower positions).

Figure 6 plots the distributions of delays between an ad request and the first ad view for the generated channel, and between and ad view and the corresponding ad click. The former is an inherent attribute of the private-by-design architecture and depends on the delays between system components. Additionally, the request-view delay is affected by the channel selection and ad placement policy at the client and is subject to the availability of the adboxes. Overall, our prototype was able to generate, deliver and display an ad within an hour of establishing a new interest for 60% of

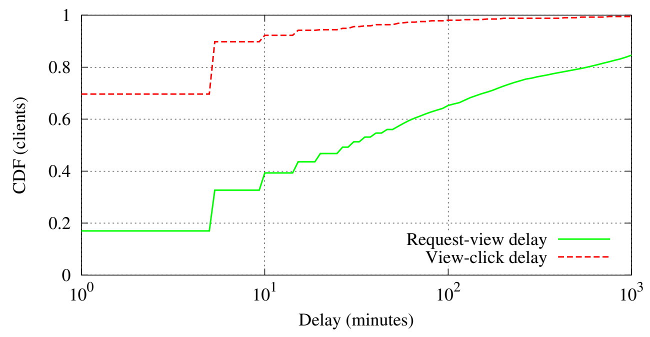


Figure 6: CDFs of delays between ad requests and ad views, and between ad views and ad clicks

all channels. With the most aggressive configuration parameters (channel selection = 'most recent' and ad placement = 'everywhere') the 60th percentile of the delay is below 30 minutes. We believe this to a large extent comes from the time it takes for an adbox to show up.

Additionally, using PDDP we measured the delay between sending an ad request and receiving ads from the generated channel (this information is only recorded locally at the client). This delay for all responded clients was <10 minutes. There are several contributing factors to the request-response delay: length of the upload cycle at the dealer, polling interval at the client and the ad generation latency. While the first two are configurable, the latter is implementation-specific. In our experiments, the observed average ad generation latency was below 2 seconds (with 99th percentile of 7 second).

While, as Figure 6 shows, the majority of the clicks happen within minutes after an ad view (the 90th percentile is less than 10 minutes), we do not have enough data points to establish the relationship between CTR and time elapsed since identifying a new user interest. Nonetheless, we found that CTR is affected by the number of times an ad was shown (so-called, 'opportunities to see'), dropping significantly after the third ad view.

Overall, we find the achieved CTR to be encouraging given the fact that we fully relied on shopping engines to match ad requests to relevant products and that our ad content was produced automatically from resulting products (which, despite our best effort, sometimes did not look as intelligible and appealing as handcrafted ads). Naturally, we could have seen higher CTR by rendering our ads only in top positions. Apart from this, we believe that Privad CTR can be increased by improving targeting heuristics (e.g., reducing noise in search-based targeting), investing in better request matching algorithms and serving advertising content designed by hand, not auto-generated.

5. COLLECTING DATA WITH PDDP

As opposed to the aggregate performance stats maintained by Privad using view and click reports, PDDP provides a privacy preserving mechanism to collect per-user stats and perform user-centric analysis. In this section, we describe our experience exploring the extent to which a differentially private data collection system can be used to understand what is going on behind the scenes in the private-by-design ad deployment.

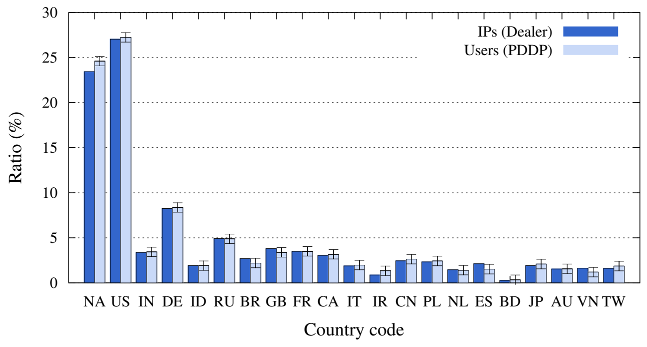


Figure 7: Geographical distribution. Error bars correspond to two standard deviations = 0.5% (95% confidence interval). NA represents countries not included in the top-20 list.

Towards this end, we start out by building confidence in the differentially private results by collecting attributes that characterize the Privad's user population and comparing them with the data available at the server-side. We then exercise PDDP functionality to get better visibility into the client-side and understand the reasons for the observed view and click rates. We also analyze advertising performance for Google ads and compare it with Privad's. We examine the difference between search and display ads in terms of the before- and after-click user behavior. Finally, we look at the privacy deficits accumulated as a result of our analysis and study the privacy implications for the end users.

5.1 Aggregate User Characteristics

We start our PDDP data collection with a simple query retrieving the user geographical region. To obtain the geographical data the clients call the Maxmind GeoIP API. This value is updated whenever the browser restarts and is stored in the client's SQLite database. The corresponding PDDP query is a simple one-line select statement with buckets enumerating the top 20 most represented countries (in terms of the volume of IPs that appear in the dealer's logs). The query was active during a 24-hour time interval. In total, we collected 4604 answers with 585 random coins added to each bucket,[8] which corresponds to a standard deviation of 12.09.

Figure 7 plots two distributions of Privad users over a list of countries. One is based on the noisy answers from the second PDDP query, the other is computed using 4607 IPs extracted from the dealer's logs from the same 24-hour period. The tallest bar on the figure corresponds to the US users, and the more sizable European population is spread over a number of countries including Germany, Russia, Great Britain, France and others. Almost all values computed from back-end data lie within the 95% confidence interval of respective PDDP values, with the exception of the Spain and NA ratios. The reason for this minor mismatch is likely to be that local GeoIP info is captured as soon as the browser starts and is not updated until the next browser restart, therefore it can be somewhat stale by the time a client received the PDDP query. For example, if a user enabled a proxy or joined a different network, the IP address recorded in the dealer's log can be different from the one used to retrieve local GeoIP info. Overall, we find that the back-end

[8]For all PDDP queries in our analysis we use $\epsilon = 1$.

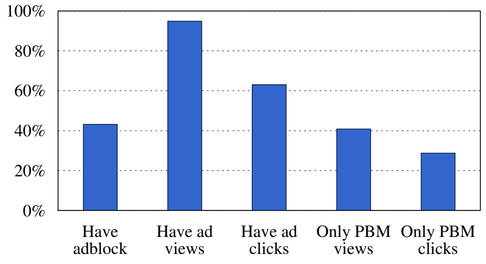


Figure 8: User attributes collected with PDDP. 95% confidence interval is $\pm 0.6\%$. PBM stands for private browsing mode.

data confirms trends discovered using the differentially private mechanism, which serves as a practical validation of PDDP results.

We use the rest of this section to describe our experience in exercising the PDDP functionality to the widest extent possible in order to learn everything we could about our deployment.

To build a better picture of Privad users, we used PDDP to query the operating system installed on the user machine (also available as a part of Firefox API). We found that among the 4428 clients who responded to the query, 64.2% run Windows, 21.6% use OSX and 14.4% have a Linux installed. Clearly, with the ratio of Linux users significantly higher than in the general population (1.73% according to [22]), this sample is representative of a set of technically savvy power users.

Finally, we executed a query to find the percentages of adblocked users as well as users with views and clicks for all types of ads (both Google and Privad). The query was active for 96 hours and accumulated a total of 5909 answers. Among the users who submitted their answers ca. 709 spent less than two weeks with the system and their values were not included in the query results. Figure 8 shows the distribution of the remaining user answers over queried attributes. Consistent with the previous observation about the tech-savvy user population, we found the ratio of adblockers among Privad users to be twice the rate for Firefox users reported by [8]. Surprisingly, despite a large number of adblocked users, the vast majority still receive ad views (mainly because popular ad-blocking addons consider Google Search ads to be "acceptable" [2] and by default do not remove them; by contrast, *no* Privad ads are displayed if our client detects an adblock). Moreover, more than 63% of users in the sample have ad clicks (for users who block ads this ratio is only marginally smaller – 60%). Additionally, 40% and 28% of users have views and clicks registered only during private browsing mode (PBM). This suggests that more than a third of Privad users have PBM enabled most of time, during which Privad client will neither request nor display any ads.

Overall, our PDDP-enabled analysis reveals a technically advanced user base, where a large fraction have ad-blocking addons and browse in private mode. The outcome is far from unexpected, given our "bundling" deployment mechanism. On top of that, the users who opt into a study of private-by-design advertising are likely to be aware of the privacy concerns related to online advertising and to use all means available to minimize their privacy exposure on the web.

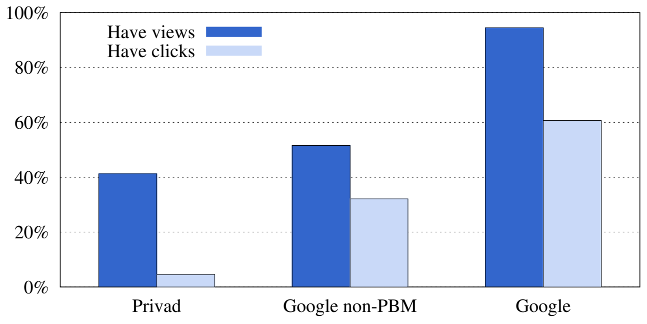


Figure 9: Ratios of users with views and clicks. Timespan = 96 hours, total answers = 5584, 95% confidence interval = ±0.7%.

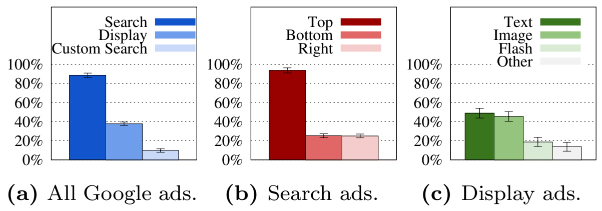


Figure 10: Breakdown of users with at least one Google click by type of the clicked ad. Timespan = 24 hours, total answers = 4624.

5.2 Comparing Privad and Google Performance

In this section, we present the results of PDDP analysis performed to compare and contrast Privad and Google ads performance. Figure 9 plots the ratios of users with views and clicks for both Privad and Google ads. It shows that less than 5% of sampled users have Privad clicks, whereas more than 30% have non-PBM Google clicks. We did not store any details about events that occurred in PBM apart from the type of event (i.e., 'view', 'click', 'search', etc). But since no Privad ads are displayed in PBM, all ad-related PBM events are attributed to Google. In total, we found that more than 60% of all queried users have registered Google clicks. Due to lack of information about events in PBM and to be apples-to-apples with Privad, unless otherwise stated, we exclude PBM views and clicks from further analysis.

To learn which Google ads are most popular, we break down users with clicks into groups according to the type of the clicked ad. As Figure 10a shows, among the approximately 1387 users with Google clicks in this sample, the vast majority registered a click on a search ad. Display with almost 40% of all clickers is runner-up. Figure 10b further breaks down search ads according to location, where top ads (displayed above organic search results) take the lead with more than 90% of 1227 users with search ad clicks. Finally, among display ads (Figure 10c), text and image have roughly equal shares each with 40% of users with display clicks (ca. 524 users in the sample).

To estimate the distribution of per-user Google click-through rates, we issued a query computing the ratio of views to clicks that occurred in October 2013 for all Google ads (including ads displayed in PBM). Since every bucket in a PDDP query incurs a penalty in terms of the added privacy deficit, we generally strove to use as few buckets as possible. In particular, in order to cover the whole range of possible CTR values in this query we used exponentially increasing bucket sizes (with every bucket covering twice the range of its precursor). Among 5392 user who responded to the query, 865 did not have any views, CTR values for the remaining ca. 4527 users are distributed as shown in Figure 11. The first histogram on the figure represents the ratio of users with no clicks, while the last – the ratio of users with CTR values ≥ 6.4%. Using the lower endpoint of a bucket range as a conservative estimate of the CTR value for a user within the bucket, we compute a lower bound on the average Google CTR across all users – 1.53%.

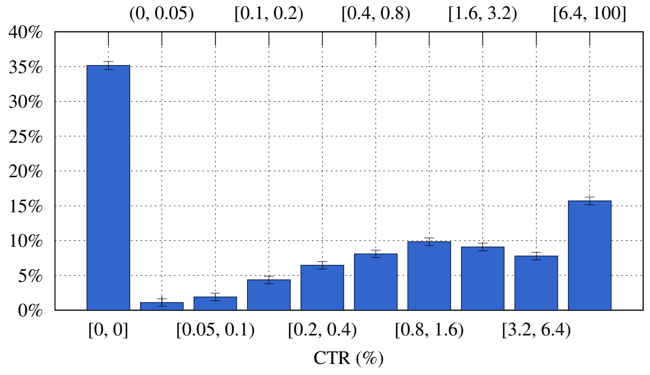


Figure 11: Distribution of per-user Google click-through rates. Timespan = 96 hours, total answers = 5392. Error bars denote 95% confidence interval. Labels along X-axis represent corresponding bucket ranges.

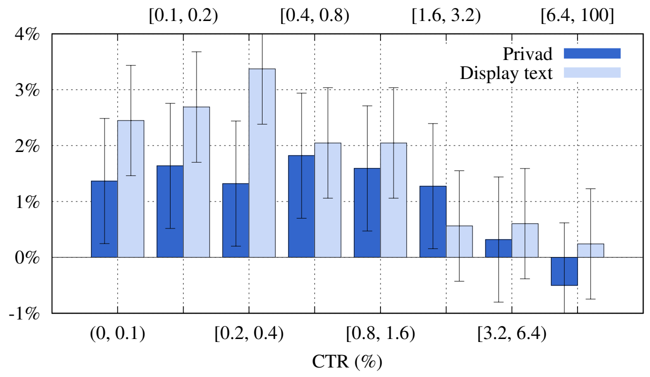


Figure 12: Distributions of per-user click-through rates for Privad and Display text ads. Timespan = 96 hours, total answers = 6317. Zero-CTR ratios (90 ± 1.5% for Privad, 85 ± 1.3% for display text ads) not shown.

Privad is an inherently *asynchronous* system designed for behavioral advertising, not search advertising. The prototype we built generates and displays only contextual ads. Therefore, it is most reasonable to compare performance of Privad ads and text ads on the Google Display Network (which we refer to as *display text ads*). To do so, we executed a query computing per-user CTR values for both types of ads. The results of this query are presented in Figure 12, excluding the ratios corresponding to zero-CTR. While dominated by differentially private noise, both CTR are generally quite similar.

To verify this observation, we used PDDP to compare the Privad and display text CTR values of each individual user. We collected 4844 answers with 588 coins added to each bucket (stdev = 12.1), among which 2994 had no Privad

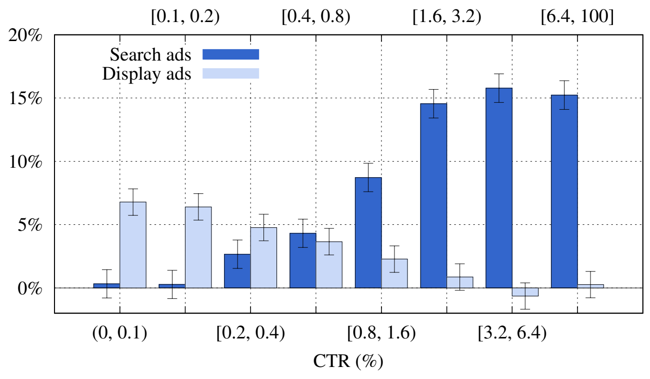


Figure 13: Distributions of per-user click-through rates for Search and Display Google ads (all subtypes). Timespan = 72 hours, total answers = 5272. Zero-CTR ratios ($40 \pm 1.2\%$ for Search, $74.4 \pm 1.3\%$ for Display) not shown.

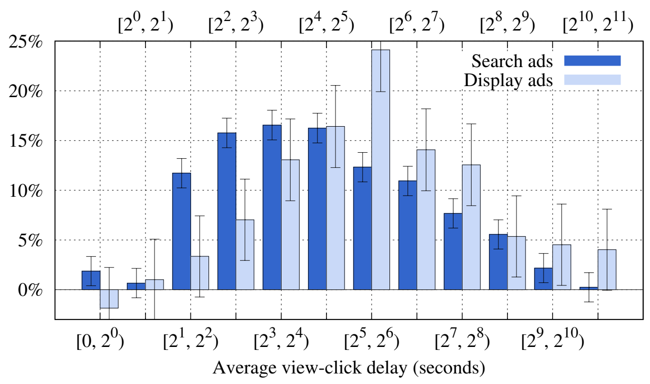


Figure 14: Distributions of per-user average view-click delay for Search and Display text ads. Timespan = 48 hours, total answers = 5239.

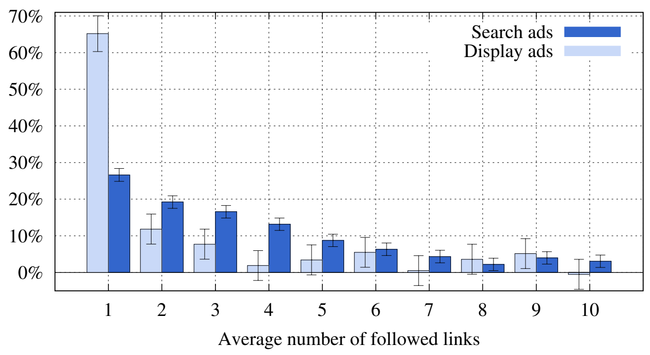


Figure 15: Distributions of per-user average click-chain length for Search and Display text ads. Timespan = 48 hours, total answers = 4895.

or display text views, and 1460 had no clicks. Among the remaining 391 users, we found that in 210 cases the display text CTR was higher, and in 181 the Privad CTR was higher.

In addition to detecting views and clicks, Privad clients use a number of heuristics to identify conversion (i.e., purchases made after a click). Towards this end, clients tag all pages visited after an ad click with the ad id and try to detect online payments (e.g., credit cards numbers in post request parameters that pass through Luhn's validation) issued on pages tagged with a valid ad id. To compare how Privad, with 4 actual conversions, fares against Google, we executed a PDDP query counting the number of detected Google conversions. From 6022 users who responded, 107 users made a purchase following a click on a Google ad (stdev = 12.3), 23 made a conversion while in PBM and 29 had more than one conversion. However, the majority of conversions (ca. 92) were generated by search ads, with display ads accounting for only 7 conversions (well below the noise level).

In general, using PDDP we found that both in terms of the click-though rates and conversions Privad ads perform on a par with text ads on Google Display Network. While recommendation systems and ad targeting are well studied in as long as they are performed in the cloud, it still remains a challenging and open question how to do targeting on the client using only the local profile information. The preliminary results reported here suggest that even with a few simple heuristic for targeting from the localhost Privad is as effective as the ad network, which has a global view of the user profiles and employs complex machine learning algorithms.

5.3 Comparing Search and Display Performance

As mentioned in the previous section, our PDDP analysis reveals a difference in the number of conversions attributed to search and display ads. The dissimilarity between these ad types is even more prominent when we compare the distributions of per-user click-through-rates (Figure 13). More than half of roughly 2200 users with search ad views have CTRs $\geq 0.8\%$, whereas display ad CTRs for almost all users do not exceed this value. Also, all search clicks were generated by text ads, but half of the display clicks comes from text ads and half from image ads.

After observing a stark difference in performance between search and display ads, one of the remaining questions was whether these two ad types differ in terms of the before- and after-click user behavior. In other words, we want to find out how long users linger before clicking on an ad and whether they remain interested in and engaged with the landing page content a long time or leave soon after the click. Our primary goal here was to exercise the PDDP functionality to the widest extent possible without making any significant claims regarding observed results. Therefore, we compare search and display focusing only on text ads. In order to increase size of the population with display text clicks, we include Privad clicks in this category, since both display text and Privad ads are visually similar and share the same performance characteristics.

Figure 14 plots the distribution of the average per-user delay between an ad view and the subsequent click on the ad. The mean delay for display ads is around 30 seconds to 1 minute, while for search ads it is between 8 and 30 seconds. In part, this difference could be explained by the fact that we register a view at the time of the page load, and not when an ad is within the visible area of the browser window. And while top search ads appear immediately on the screen, oftentimes a user has to scroll down to see a number of display ads hidden outside of the visible area.

Another metric we use to analyze post-click behaviour is the length of the "click-chain" – the number of pages visited by following links on the advertiser's webpage. The distributions of the average click-chain length are shown on Figure 15. On average, the majority of users visit only the

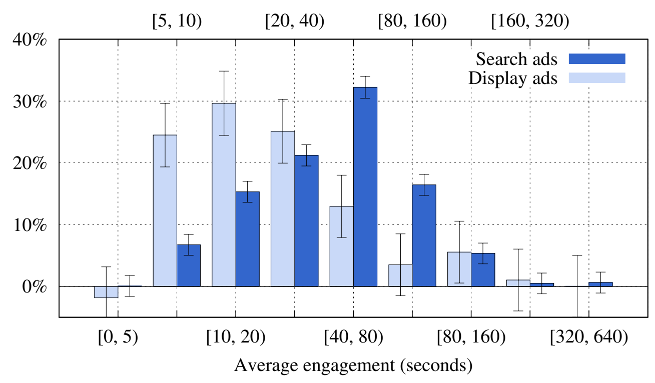


Figure 16: Distributions of per-user average engagement for Search and Display text ads. Timespan = 48 hours, total answers = 5059.

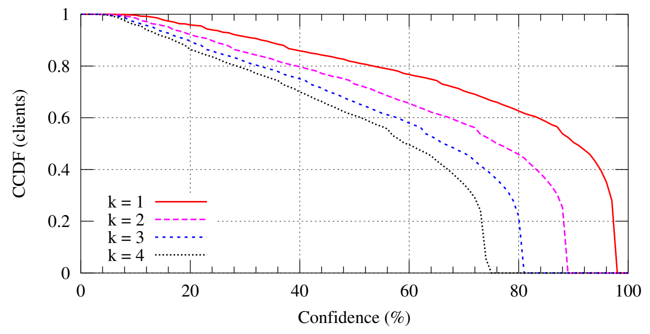


Figure 17: CCDFs of the attacker's confidence of discovering k user attributes given each user's privacy deficit (theoretical worst-case).

landing page of a display text ad (click-chain length = 1) and rarely follow 1-2 additional links (counts corresponding to buckets with more than 3 visited pages are dominated by differentially private noise). The same distribution for search ads is more evenly spread with bucket counts corresponding up to 7 visited pages well above the noise level.

Finally, Figure 16 presents the distributions of time spent actively browsing advertiser's content for both ad types. Again, the figure clearly shows a shift between distributions with the average engagement reaching a peak at 10 to 20 seconds for display ads and 40 to 80 seconds for search ads.

5.4 Privacy Deficit

Overall, we executed 159 PDDP queries, collecting 790017 answers from 9395 unique clients. For all queries we used $\epsilon = 1$ and added a number of coins to satisfy the $\delta < 1/c$ requirement [7]. As already mentioned, we treat individual buckets as separate queries and, therefore, compute the (ϵ, δ)-privacy cost of a query as $(1, 1/c)\times$ *number of buckets*. The cumulative privacy deficit for each client is recorded at the dealer. This dataset shows that the maximum per-client privacy cost of our PDDP analysis is $(\epsilon = 3006, \delta = 0.6)$. Moreover, 20% of the clients who participated in the analysis accumulated a deficit of more than 97% of the maximum. One way to understand this amount of privacy deficit is to estimate how many user attributes it will allow an attacker to learn under a worst-case scenario assumption (i.e., using the isolation attack described in Appendix A).

The maximum (ϵ, δ)-privacy deficit of (3006, 0.6) in our dataset allows an attacker a single query with $t = 3006$ identical buckets and $n = 590$ coins added to each bucket. This enables the attacker to learn a single sensitive user attribute with confidence >97% (alternatively, using two queries with 1503 buckets the attacker can learn two attributes with confidence >89%). Figure 17 plots the attacker's confidence across all users, based on the number of answers produced by each user. This shows that, according to the differentially private model, an attacker could have predicted a single attribute for 40% of the users with 95% confidence (or two attributes with 83%).

We cannot, however, conceive of any query that would have allowed us to learn what Figure 17 suggests. Imagine that we had enough auxiliary information about *one* of the clients to formulate a query that isolated that client from all the others. Then using a malicious 3006-bucket query, we could have learned one thing about our victim. But we would have learned nothing about the other clients that answered the query, even though theoretically those clients experienced privacy loss.

Admittedly, the differentially private threat model, which assumes an omniscient adversary, is prohibitively pessimistic. Relative to our actual usage of PDDP analytics, the predicted privacy loss is a poor reflection of the de facto privacy lost during the experiment. Nonetheless, as it has been long known [9], merely adding zero-mean random noise is inadequate in longitudinal analytics. A number of additional measures must be taken to raise the bar for an attacker. These range from limiting the number of buckets in a query [6] to running taint analysis to ensure that the same attribute is not used to answer repeated queries.

6. SUMMARY AND FUTURE DIRECTIONS

In this paper, we described our experience and challenges involved in building, deploying and operating a private-by-design ad system. Much of this work is empirical in nature, and as such is full of experimental warts and idiosyncrasies (mostly acknowledged). Overall, we believe that the process we went through to see our prototype deployed and used by thousands of users is as much a contribution of this work as the results we obtained. This experiment provided us with ample evidence and helped answer several key questions such as: is the private-by-design technology a non-starter, what can a researcher do to evaluate a private-by-design system, short of an actual start-up, and what compromises are made in the process. We learned several lessons from this experiment.

First, Search ads, which reflect user interests in real-time, clearly perform better than Display ads. Therefore, technologies like Privad, which delay ad delivery, face a serious limitation. On the other hand, compared with Display text ads, Privad performed unexpectedly well. A number of additional improvements can be made to achieve even higher click-through rates.

Second, using PDDP analytics we learned that the population of users who opted into the study was biased towards technically advanced power users, many of whom have ad blocking software, browse in private mode and tend to rarely click on ads. Therefore, we believe that the observed performance is an overly conservative estimate of the click-through-rates in general.

Third, when used with Privad's threat model, which assumes an honest-but-curious adversary, differential privacy produces an excessively pessimistic estimate of privacy loss. In reality, we came nowhere near learning any attributes about any specific individual. Additionally, an expected, but still negative result of this study is that, without additional measures to raise the attack bar, differential privacy is inadequate for long-running analytics.

The next step in Privad's evolution is to build an auction component [19] and deploy the system within an ad exchange. However, we do not believe this is feasible in a purely research setting. Rather, a commercial deployment is needed.

7. REFERENCES

[1] Alexa - The Web Information Company. `http://www.alexa.com/`.

[2] Allowing acceptable ads in Adblock Plus. `http://bit.ly/Ltv9Jn`.

[3] Apache Thrift. `http://thrift.apache.org/`.

[4] InvisibleHand: Home. `http://www.getinvisiblehand.com/`.

[5] Redesigned Text Ads on the Google Display Network. `http://bit.ly/1hJAerq`.

[6] I. E. Akkus, R. Chen, M. Hardt, P. Francis, and J. Gehrke. Non-tracking Web Analytics. In *ACM Conference on Computer and Communications Security*, 2012.

[7] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke. Towards Statistical Queries over Distributed Private User Data. In *NSDI*, 2012.

[8] ClarityRay. About Ad-Blocking. `http://clarityray.com`.

[9] D. E. Denning and P. J. Denning. Data security. *ACM Comput. Surv.*, 11(3):227–249, Sept. 1979.

[10] C. Dwork. Differential privacy. In *Proceedings of the 33rd international conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12. Springer-Verlag, 2006.

[11] C. Dwork. Differential privacy: a survey of results. In *Proceedings of the 5th international conference on Theory and applications of models of computation*, TAMC'08, pages 1–19. Springer-Verlag, 2008.

[12] M. Fredrikson and B. Livshits. Repriv: Re-imagining content personalization and in-browser privacy. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 131–146. IEEE, 2011.

[13] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[14] S. Guha, B. Cheng, and P. Francis. Privad: Practical Privacy in Online Advertising. In *Proceedings of the 8th Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, Mar 2011.

[15] S. Guha, A. Reznichenko, K. Tang, H. Haddadi, and P. Francis. Serving Ads from localhost for Performance, Privacy, and Profit. In *Proceedings of the 8th Workshop on Hot Topics in Networks (HotNets '09)*, New York, NY, Oct. 2009.

[16] H. Haddadi, P. Hui, and I. Brown. Mobiad: private and scalable mobile advertising. In *Proceedings of the 5th ACM international workshop on Mobility in the evolving internet architecture*, pages 33–38. ACM, 2010.

[17] B. J. Jansen, D. L. Booth, and A. Spink. Determining the informational, navigational, and transactional intent of web queries. *Information Processing & Management*, 44(3):1251–1266, May 2008.

[18] K. Partridge, M. A. Pathak, E. Uzun, and C. Wang. Picoda: Privacy-preserving smart coupon delivery architecture. In *PETS Symposium*, 2012.

[19] A. Reznichenko, S. Guha, and P. Francis. Auctions in Do-Not-Track Compliant Internet Advertising. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS)*, Chicago, IL, Oct. 2011.

[20] The Eclipse Foundation. `http://www.eclipse.org/jetty`.

[21] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy Preserving Targeted Advertising. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, Apr. 2010.

[22] Wikipedia. Usage share of operating systems. `http://bit.ly/1daMYZk`.

APPENDIX

A. PDDP ISOLATION ATTACK

Let's assume that an attacker can isolate a single client (for instance, by constructing a query in such a way as to single out one user) and repeatedly pose the same query to this client (or alternatively a single query with a large number of identical buckets). If the system adds at least n coins to an answer, the noise generated in a single trial is $\sum_i^n x_i - n/2$, where x is value of an individual coin (0 or 1). The expected value of the noise in a single trial is 0 (noise is distributed binomially with success probability $p = 0.5$). The average noise value after t trials is $\frac{1}{t}\sum_k^t(\sum_i^n x_i - \frac{n}{2}) = \frac{1}{t}(\sum_i^{nt} x_i - \frac{nt}{2})$, which according to the central limit theorem follows approximately normal distribution $\mathcal{N}(0, n/4t)$. 95% of the values of this distribution lie within two standard deviations of the mean: $[-\sqrt{n/t}, \sqrt{n/t}]$. The noise is effectively cancelled out when the width of this interval is < 1. In other words, if after t trials an attacker observes average PDDP value from $(-0.5, 0.5)$ interval, it infers that the true user value is 0 with 95% confidence (similarly a value from (0.5, 1.5) interval corresponds to the user value of 1). Solving $\sqrt{n/t} < 0.5$ for t, gives that if $t > 4n$ the noise is cancelled out with 95% confidence (for 99% confidence t $> 9n$).