2015 IEEE 35th International Conference on Distributed Computing Systems

Advertiser and Publisher-Centric Privacy Aware Online Behavioral Advertising

Jingyu Hua, An Tang and Sheng Zhong
State Key Laboratory for Novel Software Technology,
Department of Computer Science and Technology, Nanjing University
Email: huajingyu@nju.edu.cn, atang.nju@gmail.com, zhongsheng@nju.edu.cn

***Abstract*—Online behavioral advertising (OBA) has become one of the most successful advertising models on the Internet. Nevertheless, all existing OBA systems are broker-centric in the billing phase, which means it is the broker who exclusively determines advertisers' expenses and publishers' revenues. Consequently, a malicious broker may cheat in their tallying of ad clicks to overcharge advertisers or underpay publishers. Furthermore, as the broker cannot justify the bills, malicious advertisers may deny actual clicks to ask for refunds, and malicious publishers may claim non-existing clicks to demand extra revenue shares. This paper solves these problems by reversing the priority between the broker and the advertisers and publishers. Specifically, when users click on ads, it makes corresponding advertisers and publishers forward click reports of clients to the broker after checking, anonymizing and signing them. The broker then settles accounts with advertisers and publishers fully based on these reports. To guarantee the interests of the broker after the priority reversal, we further propose effective mechanisms for detecting underreporting advertisers and overreporting publishers, respectively.**

***Index Terms*—Online behavioral advertising, privacy-awareness, billing security**

I. Introduction

Advertising is the major source of revenue for most online service providers. In recent years, online behavioral advertising (OBA) systems such as Google's ad networking, achieve a big success as they significantly improve the effectiveness of online advertising.

A. Existing OBA Systems and Their Problems

Typically, existing OBA systems consist of four primary players: advertisers, publishers, brokers and clients [4]. Advertisers want their ads for their products to precisely reach potential purchasers. For this purpose, they specify the profiles of targeted users for each ad, and send them along with the bid amounts to the broker. Publishers hope to display ads on their websites for receiving revenue from the broker. They place ad boxes pointing at the broker on their webpages. When a user views such a page, the client program on his machine is invoked. This program sends the user's profile, which is maintained by tracking his online activities, to the broker. The broker compares it with the ad profiles and returns one or several ads that maximize his expected revenue based on some auction algorithm. The client displays these ads in the ad boxes, and the user may click on them. The corresponding advertiser will be charged for each ad view in the pay-per-view (PPV) model or each ad click in the pay-per-click (PPC) model. The broker has to share part of his revenue with the publisher. Without loss of generality, we consider the PPC model only in this paper.

Sheng Zhong is the corresponding author. This work was supported by NSFC-61321491, NSFC-61425024, and NSFC-61300235.

These OBA systems are subject to at least the following two problems:

First, all existing OBA systems are **broker-centric** in the billing phase, i.e., that it is the broker who exclusively determines how much advertisers have to pay, and how much revenue share publishers could gain. This unfair environment causes many serious problems. First, a malicious broker may cheat in his counts of ad clicks to overcharge advertisers or underpay publishers. We cannot guarantee that the broker is always honest because we cannot verify their applications running on the cloud. In addition, this design, although seems to favor the broker, may as well bring troubles to the broker: as the broker cannot justify his settlement decisions, malicious advertisers may deny actual clicks to ask for refunds, and malicious publishers may claim non-existing clicks to demand greater revenue shares than that the broker has paid. There have been some news saying that brokers like Google are accused of overcharging advertisers [1], [2], [8].

Second, the personalized advertising **raises serious privacy concerns of users** [17]. In the ad delivery, clients have to upload user profiles, which may contain sensitive information. In the click reporting, clients inform the broker what ads that users have clicked on, which may unexpectedly disclose interests of users to the broker. There have been many proposals for addressing this problem. Most of them work by introducing a trusted or semi-trusted (honest-but-curious) third-party server [10], [12], [9], [19] or some trusted hardware (e.g., [4]) between the broker and clients to anonymize messages from clients. Nevertheless, such kind of facilities are rare in the real world. Targeting the privacy leakage in the click reporting, various cryptographic [20] or perturbation techniques [16] are used to conceal the real contents of individual reports. Although these proposals could well protect the user privacy in the process of click reporting, none of them addresses the security problems due to the broker-centric billing. Moreover, these methods significantly increase the difficulty of capturing malicious clients who lie in their reports that have been encrypted or perturbed.

IEEE computer society

B. Our Work and Contributions

We aim to solve the above two problems at the same time in this paper. **Our basic idea is to change the traditional priority-relation between the broker and the advertisers and publishers in the billing phase**, i.e., let advertisers (publishers) themselves instead of the broker determine how much they should pay (get from) the broker. Specially, in the proposed OBA system, when an ad is clicked on, the client is made to send click reports to both the advertiser and the publisher involved. The latter two then check the truth of such reports and forward legitimate ones to the broker after aggregating, anonymizing and signing them. The broker is forced to charge advertisers (pays publishers) based on the click reports signed by advertisers (publishers) themselves.

Compared with the traditional OBA systems, the above design has the following two advantages:

(1) Because the broker is forced to settle accounts based on the click reports verified and signed by advertisers and publishers themselves, he is prevented from arbitrarily overcharging advertisers or underpaying publishers by lying about his own tallying of ad clicks. Moreover, advertisers now cannot deny the legal bills generated based on the click reports signed by themselves because the broker can easily justify the validity of these reports. Similarly, publishers cannot demand more revenue shares than those signed by themselves.

(2) It can well protect users' privacy against the broker in the process of click reporting because all the click reports are anonymized at both advertisers and publishers, and no client is required to contact with the broker directly to report ad clicks. Furthermore, the click reports sent from clients to advertisers and publishers are carefully crafted such that no information about users is leaked to advertisers or publishers other than what can be learned even without these reports. As a result, our design does not raise any new privacy concern at advertisers or publishers.

While the priority reversal has these advantages, it also produces new challenges that could hurt the interests of the broker: malicious advertiser may underreport actual ad clicks (i.e., drop click reports from clients) to reduce their bills, while malicious publishers may overreport ad clicks (i.e., add fake click reports) to drive up their revenue shares. We thereby propose effective detection methods against these malicious behaviors for the broker, who can then punish captured malicious advertisers or publishers to prevent their further attacks.

In summary, we make the following contributions in this paper:

- We propose an advertiser and publisher-centric OBA system. It can not only protect the user privacy in click reporting but also solve the security problems due to the inequality between the broker and advertisers and publishers in traditional systems.
- We propose effective mechanisms to prevent malicious advertisers and publishers from underreporting or overreporting ad clicks after the priority inversion.
- To defend against click-fraud, we extend the most common countermeasures in existing OBA systems to our proposal.
- We use click-through-rates (CTRs) as examples to illustrate how the broker can collect different kinds of global statistics in a privacy preserving way. Note that privacy-preserving statistics gathering is a basic function required by many tasks in the proposed system.
- We experimentally evaluate the effectiveness of the proposed system.

II. Related Work

There have been many studies on the design of privacy-preserving OBA systems. However, none of them considers the security issues due to the billing unfairness. This section makes a quick review of several classic or latest systems that target privacy protection in ad delivery or click reporting.

Juels [15] is the first to study the privacy concerns in targeted advertising. They propose using a mixnet between the broker and clients for privacy-preserving distribution of ads. Their proposal could well protect the user privacy in the ad delivery. Its time cost, however, is so high that prevents the clients retrieving ads on-the-fly. Furthermore, the using of mixnet makes the broker hard to trace back the clients conducting click fraud. Finally, they do not consider the privacy protecting in other stages such as the click reporting.

Privad [9] presents a complete privacy-preserving OBA system. It introduces a *dealer* that works as an anonymizing proxy between the broker and clients to protect the user privacy in ad delivery and click reporting. Moreover, Privad introduces a reference monitor that watches the client behaviors to prevent clients from communicating with the broker through a covert channel. The problem of Privad is that the dealer is a single-point-of-failure in the system. Once the broker compromises or colludes with it, user privacy can no longer be protected. Reznichenko et al. [19] propose auction mechanisms on top of Privad.

Different from Privad, ObliviAd [4] utilizes a broker-side trusted hardware (a secure coprocessor) to anonymize the communication between the broker and clients. The advantage of ObliviAd is that it does not assume any trusted party. However, it requires additional hardware on the broker side. Moreover, this hardware is administrated by the broker and we cannot guarantee that it would not be rigged.

Adnostic [20] provides another privacy-preserving OBA system without using any anonymizing proxy. In this system, behavioral profiling and targeting take place in the users' browser, which is hidden from the ad network. The major contribution is that it proposes using homomorphic encryption and zero-knowledge proofs to allow the broker to count the total number of views for each ad without knowing who viewed which ad. However, the public-key based homomorphic encryptions and zero-knowledge proofs are a bit expensive, and require a trusted third-party that is responsible for decrypting the results.

Instead of using complex homomorphic encryption as Adnostic, Kodialam et al. [16] propose a more efficient

TABLE I
COMPARISONS AMONG EXISTING OBA SYSTEMS (THE SECOND AND THE THIRD COLUMN LIST THE PRIVACY-PRESERVING TECHNIQUES USED IN AD DELIVERY AND CLICK REPORTING, RESPECTIVELY.)

Proposals	Ad Delivery	Click Reporting	Billing Security	Click-Fraud Detection
Juels [15]	Mixnet	×	×	×
Adnostic [20]	Randomly Select	homomorphic encryption	×	√
Privad [9]	Trusted 3rd-Party	Trusted 3rd-Party	×	√
ObliviAd [4]	Trusted Hardware	Trusted Hardware	×	√
Kodialam et al. [16]	×	Perturbation	×	×
Hardt et al. [13]	Profile Generalization	×	×	×
Our proposal	Profile Generalization	Advertisers/Publishers Relaying	√	√

perturbation-based ad-view reporting mechanism that allows the broker to estimate the total number of views for each ad without learning who viewed which ad. However, this approach does not provide the broker with any information about the publisher for each ad impression, which is critical for determining how much revenue he has to share with each publisher. In addition, the authors simply assume that all ads are preloaded to each user, which is obviously impractical in a large advertising network.

Hardt et al. [13] mainly address the problem of achieving privacy-aware personalized ad delivery without using any anonymizing proxy. They propose a flexible framework that enables users to confine the amount of private information shared with the broker by generalizing their profiles. Based on the limited information provided by the client, the broker utilizes an optimization algorithm to return a set of ads that are expected to maximize his revenue under the constraint of communication complexity according to some historical statistics such as CTRs. The client then picks the most matched ones based on its exact profile to display. This proposal makes a good trade-off among privacy, efficiency and broker revenue. The authors, however, do not discuss how to report ad clicks/views for the broker-side accounting.

We summarize and compare these proposals in Table I.

III. Goals and Adversary Model

In this section, we describe the goals of the proposed OBA, and specify the adversary model considered in this paper.

A. Design Goals

The proposed system is composed of two major goals: protecting the user privacy without using any third-party proximizing proxy and guaranteeing the billing security*. We describe the details of these goals below.

As we have mentioned earlier, the current unfair relation between the broker and advertisers and publishers may cause the following problems: (1) a malicious broker may cheat in his tallying of ad clicks to overcharge advertisers or underpay publishers; (2) malicious advertisers may deny the actual clicks on their ads to demand refunds, and malicious publishers may claim non-existing clicks from their webpages to ask for more revenue shares. So, our **security goal** is to address these problems. In particular, it is composed of the following aspects:

1. Malicious brokers are prevented from overcharging advertisers or underpaying publishers by cheating in their tallying of ad clicks.

2. Malicious advertisers are prevented from denying actual clicks on their ads and underpaying brokers.

3. Malicious publishers are prevented from claiming non-existing ad clicks and being overpaid.

4. The broker should have the ability to detect and block click fraud from malicious clients.

Next, we consider the **privacy goal**. According to [9], the current OBA systems compromise client privacy mainly at two stages AD delivery and click reporting. We focus on the later in this paper. For the first stage, we directly use the nice approach proposed by Hardt et al. [13] to protect the user privacy though profile generalization. So, the primary privacy goal in this paper is to achieve **Privacy-Preserving Click Reporting**. In other words, we aim to enable a client to report ad clicks to the broker without disclosing its PII (including the IP address). This task is non-trivial if we are not allowed to use any third-party anonymizing proxy.

In addition, as we will show later, many tasks of the broker in the proposed OBA system require to make use of various historical global statistics such as the CRTs. So, our second goal is to design a **Privacy-Preserving Statistic Gathering Mechanism**, which can let the broker learn the global statistics but prevent him from getting any information on individuals other than what can be deduced from the global statistics.

Due to the page limit, we have to put the privacy analysis of our proposal in the extended version.

B. Adversary Model

We now describe our trust assumptions on each player.

1) Client: We assume that the client software is trusted, which means: (1) it obeys the prescribed operations, and (2) will not leak users' private information to any parties, including the broker through any covert channels. These two assumptions can be guaranteed by utilizing tools such as the reference monitor in Privad [9] and RePriv [7] to mediate messages leaving the browser. In addition, although the client software is considered to be trusted, we assume it may be malicious used by users to launch click-fraud attacks.

2) Broker: We assume that the broker is dishonest and may cheat in their counts of ad clicks to overcharge advertisers or

*Billing security here refers to addressing the security problems due to the broker-centric billing strategy, which has been elaborated in Sec. I-A.

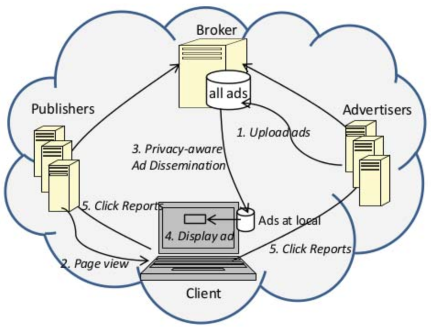


Fig. 1. Overview of the proposed OBA system (this figure is drawn based on Fig.1 of [9])

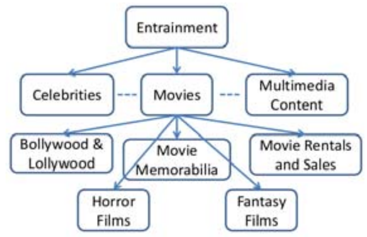


Fig. 2. Example of Google Ads Preferences categories†

underpay publishers. He is also curious about user privacy. Nevertheless, we assume it does not collude with advertisers or publishers. All existing OBA systems make the last assumption explicitly or implicitly.

3) Advertiser: We assume that advertisers are selfishly malicious towards the broker. They may deny users' actual clicks on their ads that the broker claims. They may do anything possible to cut their payments to the broker. However, they do not collude with the broker.

4) Publisher: The same as advertisers, we assume that publishers are selfishly malicious towards the broker. They may claim a higher number of ad clicks through their web pages than the actual value. They also do not collude with the broker to compromise the user privacy.

IV. Protocol Overview

Our system is composed of the same four players existing in today's advertising model: broker, advertiser, publisher and client. At a high level, the operation of this system goes as we show in Fig. 1.

Advertisers submit ads as well as their bids to the broker. The same as today's advertising systems such as Adwords of Google, Advertisers also has to specify the profiles of users targeted by each ad. A user profile usually consists of various attributes such as user interests and demographics (e.g., location, gender, age, salary). Accordingly, the client software maintains a realtime profile for each user by monitoring and analyzing his activity (e.g., webpages browsed by the user, contents of emails, visited locations from GPS). We organize the domain of each attribute into a hierarchical structure. For example, the attribute of user interests can be organized into three levels based on the hierarchical categorization scheme of Google Ad Preferences. As we show in Fig. 2, the values are generalized from the bottom to the top in their structure.

When the user visits a webpage (We assume that this page belongs to Publisher P) including an ad space, the client software requests ads from the broker with his present profile. To limit the privacy disclosure, the client generalizes the profile attributes before sending them to the broker. The broker then utilizes the algorithm proposed in [13] to select a set of ads that are expected to maximize his revenue under the constraint of communication complexity, and sends them back to the client. The client caches these ads, and utilizes some auction algorithm to rank these ads based upon the user and global modifiers in addition to the bid information. One or several top ranking ads are then displayed in the ad space. Reznichenko et al. [19] has studied the problem of client-side ad auction in details. We can easily adapt their proposal to our system and thus do not discuss this topic in this paper.

If the user clicks on an ad a belonging to Advertiser A, the client sends the click reports $r = (a_{id}, E_{k_B}(P_{id}))$ and $r' = (P_{id}, E_{k_B}(a_{id}))$ to A and P, respectively. Here, we denote by $E_{k_B}()$ probabilistic encryption with the public key of the broker. Note that probabilistic encryption introduces randomness in the encryption so that when encrypting the same message several times, it yields different ciphertexts. This property can not only prevent A (P) from getting the real value of P_{id} (a_{id}) but also prevent him from linking multiple reports sharing the same P_{id} (a_{id}).

After receiving this report from the client, the advertiser first verifies its correctness by checking whether the client really has clicked on the ad of a_{id}. If so, he stores this report in a buffer of fixed size n, otherwise it simply drops it. Once this buffer is full, the advertiser mixes all the reports in this buffer, and sends a message $m_A = (r_1, r_2, \cdots, r_n, Sig_A(r_1||r_2||\cdots||r_n||t_{now}||t_{last}))$ to the broker and flushes the buffer. Here, we denote by r_i, $i = 1, 2, \cdots, n$, the n click reports after being mixed, t_{now} the current time and t_{last} the time this advertiser sends the last similar message to the broker. The broker then charges the advertiser based on the click reports in this message.

When the publisher receives the report from the client, he directly stores it in a buffer of fixed size n. Once this buffer is full, the publisher also mixes the reports, and then sends a similar message $m_P = (r'_1, r'_2, \cdots, r'_n, Sig_P(r'_1||r'_2||\cdots||r'_n||t_{now}||t_{last}))$ to the broker and flushes the buffer. All the symbols in this message have the same meaning with those in the message sent by the advertiser. The broker then pays the publishers based on the click reports in this message. The two time-guards determine a unique time interval for the ad clicks reported in this message and the broker does not accept messages of overlapped time intervals from the same publisher. Namely, each publisher can

†This figure is drawn based on the data in Appendix A of [20]

only report ad clicks once for the same time period.

Compared with today's systems, the biggest difference of the above system is that it lets advertisers (publishers) themselves instead of the broker control how much they should pay (get from) the broker. In other words, we fully reverse the priority between the broker and the advertisers and publishers. Such a priority reversal brings us at least the following advantages:

(1) Malicious advertisers cannot deny the bills generated based upon the click reports signed by themselves because the broker can easily justify them with the attached signatures.

(2) After the broker honestly pays a publisher for the ad clicks reported by himself within a specific time interval, the publisher cannot claim more ad clicks within this interval. This is because we include two time-guards within each publisher message, and one individual publisher is only allowed to report ad clicks once for the same time interval.

(3) A malicious broker now cannot arbitrarily improve bills to advertisers or depress revenue shares with publishers by simply lying about his counts of ad clicks. It is now advertisers and publishers but not the broker that dominate the billing period.

(4) Users can well preserve their privacy from the broker as they do not have to contact directly with the broker for reporting ad clicks. Advertisers and publishers in our system aggregate click reports from a number of clients, and forward them to the broker after anonymizing and mixing them. This design prevents the broker from linking ad clicks with specific users. Furthermore, advertisers and publishers in this process cannot get any extra information on users except what they can naturally learn even without those click reports.

Unfortunately, besides these advantages, this special design also brings some challenges that we have to address before the system can be applied to the practice:

(1) How can the broker prevent advertisers from underreporting ad clicks, while prevent publishers from overreporing ad clicks?

(2) How can the broker defend against fraud clicks from malicious users? Most of existing countermeasures against fraud clicks work in the condition that clients report ad clicks of users directly to the broker.

(3) How can the broker gather various global statistics (e.g., CTR) on ads in a privacy preserving way? The broker in our system needs some global statistics to help picking a set of ads that are expected to maximize the broker's revenue.

V. Detailed Design

In this section, we describe our detailed solutions to the challenges that we present at the end of the last section. We first present the countermeasures against underreporting advertisers and overreporting publishers, respectively. We then extend several existing detection approaches of click fraud to the proposed system. Finally, we use different kinds of CTRs as examples to elaborate how to gather global statistics in a privacy-preserving way.

A. Defending Against the Underreporting of Advertisers

In the proposed system, the broker charges advertisers based on the click reports sent from advertisers themselves. As a result, it is possible that some malicious advertisers who underreport ad clicks (by dropping part of click reports from clients) for the purpose of reducing their costs. We have to design effective countermeasures against such underreporting behaviors, otherwise the broker suffers from the revenue loss.

1) Sampling-based Countermeasure: Recall that in each click report from advertisers to the broker, there is a field, i.e., $E_{k_B}(P_{id})$, whose value is received from the client after being encrypted with the public key of the broker. It is computationally infeasible for advertisers to decrypt them and learn their original values. In addition, as the encryption uses some probabilistic encryption algorithm such as ElGamal, where a given plaintext is encrypted to one of a large set of ciphertexts, it is also infeasible for advertisers to filter out reports sharing the same plaintext in this field. We leverage this field to design the first sampling-based countermeasure against advertisers' underreporting attacks. Specially, when the broker suspects that some advertiser is cheating, he randomly picks k online clients and commands them to actively click ads belonging to this advertiser. Additionally, the broker sends a special nonce to these clients, who will forward $E_{k_B}(nonce)$ to the advertiser instead of $E_{k_B}(P_{id})$. By doing so, the broker is expected to receive k reports that contain the ciphertexts of the special nonce in the second field from the advertiser. Otherwise, the advertiser is considered to be dishonest in reporting ad clicks.

We now analyze the real effect of this countermeasure. We assume that a malicious advertiser independently drops clients' click reports at a probability of p. Then, the expected probability for the broker to capture this underreporting advertiser, i.e., the probability that at least one of the k verifying reports is dropped, is $1-(1-p)^k$. As we will show in Sec. VI-A, this detection rate can be extremely high even with a small k provided the value of p is not small. For instance, if we assume that $p = 10\%$, which is reasonable since the malicious advertisers have the incentive to use a big p to reduce his cost markedly, the detection rate can exceed 90% by just triggering 25 auditors.

Nevertheless, the working of the above sampling-based countermeasure needs a premise that the detection process is carried out just when the advertiser is performing the underreporting attack. Otherwise, the broker has no ability to capture the malicious behaviors of the advertiser. For instance, if a malicious advertiser drops user clicks only between 10 and 10:30 am, the broker cannot detect this attack if he triggers the detecting process during other time. The only way to solve this problem is making the broker constantly choose clients to audit advertisers. Unfortunately, malicious brokers may exploit this measure to bring additional high costs to the advertisers since they have no ability to distinguish between normal and auditing clicks.

We now present a solution to this problem under the assumption that the broker does not collude with clients. When the broker collude with client, this problem actually turns out to be a typical distributed click-fraud attack. We will delay our discussion on click-fraud attack to Sec. V-C. Our solution makes each client add additional information ($k_1 \cdot x + k_2$,k_1, $E_{k_B}(k_2)$) to his report to the advertiser. Here, the integer $x = 2$ if this report is for auditing; otherwise $x = 1$. The integers k_1 and k_2 are randomly picked from $\mathbb{Z}^+$ such that $k_2 > k_1$. When the advertiser receives this information, it is made to forward $E_{k_B(k_2)}$ to the broker together with the signed click report. The broker then decrypts this message and returns k_2 to the advertiser, who can then derive the original value of x and know whether the current client click is just a verification. If so, he will not pay for this click.

In the above scheme, the advertiser does not know k_2 before sending the new click report to the broker. In this case, it is difficult for him to predict x just based on the value of $k_1 \cdot x + k_2$ and k_1. This is because whenever $x = 1$ or $x = 2$ the advertiser can find appropriate k_2 provided that $k_2 > k_1$. Therefore, the advertiser still cannot distinguish between the normal and auditing reports, which means the proposed sampling-based countermeasure can work all the same. We now analyze whether this scheme can well defend against malicious brokers trying to overcharge advertisers. As we learned earlier, the true value of x is equal to 2 for an auditing report. To falsely charge an auditing click as a normal one, a malicious broker should have the ability to make the advertiser derive $x = 1$ by proving him an incorrect value of k_2. Let k_2' denote the incorrect value, i.e., $k_1 + k_2' = 2 \cdot k_1 + k_2$. It is easy to obtain that $k_2' = k_1 + k_2$. As the broker does not know the random integer k_1 under the assumption that he does not collude with the client, it is infeasible for the broker to predict the correct value of k_2'. Therefore, this scheme can well defend against the malicious broker.

2) Secure-Sum based Countermeasure: The basic idea is to enable the broker to estimate the numbers of ad clicks by himself, and thus he could compare his own results with those reported by advertisers to capture underreporting advertisers. As we have introduced earlier, in our proposal each client receives a set of candidate ads from the broker in the procedure of ad delivery. We denote them by $A = [a_1, a_2, \cdots, a_k]$. Once the user clicks on one of them, the broker creates a binary vector $B_A = [b_{a_1}, b_{a_2}, \cdots, b_{a_k}]$, where $b_{a_i} = 1$ indicates the user has clicked on ad a_i. We make the client send a perturbed report $C_A = [c_{a_1}, c_{a_2}, \cdots, c_{a_k}]$ to the broker:

$$c_{a_i} = \begin{cases} b_{a_i} + \delta & \text{with probability } \gamma \\ 1 - b_{a_i} + \delta & \text{with probability } 1-\gamma \end{cases}, \tag{1}$$

where $\delta \sim N(0, \sigma^2)$. Assume that during some period the broker receives n reports that include the element c_a defined for ad a, and also assume the real click number of this ad is x. It is easy to find that the expected sum of the values of c_a from the n reports is $\overline{\sum_{c_a}} = x\gamma + (n-x)(1-\gamma) + \sum_\delta$. If n is large enough, the sum of δ approaches zero according to the law of large numbers (LLN). Then, based on the expected sum $\overline{\sum_{c_a}}$ and its real value $\sum_{c_a}$, the broker can estimate the actual number of clicks during this period:

$$x = \frac{\sum_{c_a} - n(1-\gamma)}{2\gamma - 1}. \tag{2}$$

Since the clients have perturbed their click reports with two random variables (γ and δ), it is difficult for the broker to predict which ads the users indeed have clicked on, and thus the privacy leak is limited in this phase. The broker then compares this estimation with the value counted from the reports of the advertiser, if the latter is much smaller than the difference exceeds some prescribed threshold, he can infer that this advertiser has underreported ad clicks.

Once the broker detects underreporting advertisers, he can punish them by freezing their ads. These advertisers then suffer financial loss as they lose the chance to advertise their products through the broker's advertising network.

B. Defending Against Overreporting of Publishers

In contrast with advertisers, publishers may overreport ad clicks through their web-sites to claim more revenue shares from the broker than those they deserve. We present countermeasures against this attack in this subsection.

Ad-click reports from publishers to the broker are composed of two fields: (1) the encrypted ID of the ad that a user has clicked on through the publisher, and (2) the ID of the publisher. Overreporting publishers use some ad IDs they know in advance to forge a number of fake reports for fooling the broker. However, in our design every time a user clicks on an ad, the client also sends the publisher ID to the advertiser (after encrypting it with the public key of the broker), who will then forward this to the broker. Thus, the broker could derive two copies of ad-click statistics on the same publisher based on the reports of the publisher himself and the reports of the advertisers, and the lying of the publisher may lead to the conflicts between these two copies. We make use of this observation to detect overreporting behaviors of publishers. Note that dishonest advertisers may also tamper with the publisher IDs included in their reports. Nevertheless, we assume that most of the publishers and advertisers are honest. We present our algorithm to detect overreporting publishers in Algorithm 1. Specifically, for each publisher being verified, the broker first collects the set S_1 of ad-clicks related to this publisher from the reports of all the advertisers, and compares it with the similar set S_2 reported by the publisher himself. We analyze the result under three cases:

Case 1: $S_2 \succ S_1$, i.e., S_2 contains more elements than S_1 and the gap (namely, the number of different elements) exceeds a predefined threshold ε . This indicates that either the publisher or some of the advertisers have cheated in their reports to the broker. More specifically, the publisher may have created a number of fake reports, or some dishonest advertisers may have replaced their reports related to this publisher with those related to other publishers. However, since the clients encrypt the publisher IDs with the public key of the broker before sending them to the advertisers, the

```
Input: $S_1$: the set of ad clicks through webpages of publisher
       $P$ collected from the reports of all the advertisers; $S_2$:
       the same set collected from the reports of $P$ himself
1  if $S_2 \succ S_1$ then
2      $P$ is overreporting;
3  end
4  else if $S_2 \approx S_2$ then
5      Collect $ADS_{S_2}$: the set of ads appearing in the reports of
       $S_2$;
6      Count $n_1$: the total number clicks on ads in $ADS_{S_2}$ based
       on the reports of publishers;
7      Count $n_2$: the total number clicks on ads in $ADS_{S_2}$ based
       on the reports of advertisers;
8      if $n_1 \gg n_2$ then $P$ is overreporting;
9      else $P$ is honest;
10 end
11 else $P$ is honest;
```

Algorithm 1: Algorithm to detect overreporting publishers

advertisers have no ability to tell which of their reports include the ID of the given publisher. Thus, to make $S_2 \succ S_1$ for a specified publisher, a limited number of dishonest advertisers (We assume that most of the advertisers are honest) have to blindly replace a large portion of their reports with fake ones including the IDs different from that of this publisher. Such advertisers can be easily filtered out with the same sampling-based countermeasure that we proposed in the last section against underreporting advertisers. Therefore, we come to the conclusion that the broker can safely classify the publisher as malicious under this case.

Case 2: $S_2 \approx S_1$, i.e., the different number of elements between S_1 and S_2 is within ε. The broker cannot simply infer that the publisher is honest under this case because an overreporting publisher may collude with some dishonest advertisers to achieve this. For instance, a malicious publisher may make a contract with a group of advertisers that he forges m reports with their IDs, while the advertisers replace the same number of their reports with the fakes ones pointing at this publisher, and the publisher will finally share some of his revenue with these advertisers. Moreover, if the number of collusive advertisers is large, each of them may just tamper with a small number of reports, which increases the difficulty of using the sampling approach to filter out them. Denote by ADS_{S_2} the set of ads that appear in S_2. The broker counts the total number of clicks on ads belongs to ADS_{S_2} based on the reports from all the publishers and advertisers, respectively. It is easy to imagine that the number based the reports of publishers will be much greater than that based upon the reports of advertisers if the publisher has colluded with some advertisers in the above way. Because those victim publishers whose IDs were replaced in the reports of collusive advertisers will also send reports to the broker. The broker can therefore leverage this point to detect overreported publisher in this case.

Case 3: $S_2 \prec S_1$, i.e., S_1 contains many more elements than S_2 and the gap exceeds ε. As no rational publishers will underreport ad clicks related to them, this case happens only when some advertisers replace their reports unrelated to this publisher with those related to it. Thereby, the broker can simply classify the publisher as honest under this case happening rarely.

Once the broker detects overreporting publishers, he can punish them by blocking them in his ad networks, which will prevent them from obtaining further revenue from the broker.

C. Defending Against Click-Fraud

Click-fraud refers to that some users or bots click on ads for the purpose of attacking one or more parties in the ad network [9]. For instance, a malicious publisher may hire some persons to click on ads displayed on his websites to drive up his revenue from the broker. The competitor of an advertiser may raise some bots to click on ads of the advertiser in large quantities to quickly use up the budget of this advertiser.

Because clients in our proposal do not directly report ad clicks to the broker for protecting user privacy, defending against click-fraud becomes more challenging. There exist many detection mechanisms in today's ad networks, and they usually operate in parallel to defense against click-fraud. We now discuss how to adapt them to our proposal.

Per-user Thresholds: this approach works by limiting the click rate of each client (usually identified by the IP address) [9]. Clients whose click rates exceed a threshold are identified as malicious. Recall that in our second countermeasure against the underreporting of malicious advertisers in Sec. V-A, clients are made to send a perturbed report to the broker whenever their users click on an ad. Receiving such a report, the broker could learn that the user of the corresponding client has clicked on an ad, although he does not know which exact ad he has clicked on. Thereby, this mechanism can not only help the broker detect the underreporting of malicious advertisers, but also help him learn the click rate of each client and further detect the potential click-fraud. Advertisers naturally have the ability to detect click fraud with this method. When they detect malicious clients, they are required to complain about these clients to the broker, who can determine whether advertisers are lying based on his own detection result.

Blacklist: this approach combats with click fraud via ”blacklisting” certain IP addresses for whatever reason, such as suspicion of participating in a botnet [18]. We can deploy public blacklists on the servers of advertisers, who can then make use of them to filter out reports from these machines. Since such lists can help avoiding paying for fraud clicks, advertisers are expected to be pleased to maintain them. When advertisers receive clicks from hosts in the blacklist, they still have to report these clicks to the broker but with additional marks. The broker then checks the truth of these reports before discarding them.

Historical Statistics: This approach (such as [14]) refers to that the broker maintains a set of per-client, per-publishers and per-advertiser historical statistics such as click volumes and click-through-rates (CTRs), and any sudden change in these statistics indicates the existence of click fraud [9]. While some of these statistics, e.g., the click volume through a specific publisher, are easy to obtain based upon the reports of corresponding advertisers and publishers, some others are hard to obtain due to the lack of some necessary information. For instance, the computation of CTRs requires knowing the number of ad impressions, which cannot be extracted from the reports of advertiser or publishers in our current proposal. We will discuss how to address this challenge in the next subsection.

Bait Ads: This approach (similar to [11]) refers to that the broker intentionally advertises clients some ads that do not match the user profiles. These ads are termed *Bait Ads* [9] and users are expected not to click on them. If attackers employ bots to click on ads in a large quantity, they may unwittingly click on the bait ads, and then be captured by the broker. We can easily extend this method to our system by simply setting up some honey-pot servers to host bait ads and accept click reports related to them.

D. Privacy-Preserving Statistics Gathering

In OBA systems, many tasks of the broker require to make use of some global statistics on ad clicks/views. For instance, the ad delivery strategy adopted by our system needs the CTR data in different profiles to help the broker pick the set of replying ads that are expected to maximize his revenue. In addition, as we introduced in the last section, the broker may also utilize some kinds of historical statistics to capture click fraud. Therefore, we have to design an effective privacy preserving statistics gathering mechanism for the broker. For simplicity, we use the computation of three different kinds of CTRs as examples to elaborate our solutions to this challenge, which can be easily extended to gather other statistics.

We consider the following three kinds of CTRs: (1) the CTR of an ad a, $CTR(a) = \frac{clicks_a}{views_a}$, where $clicks_a$ and $views_a$ denote the total numbers of clicks and views on this ad, respectively; (2) the CTR of an ad a in a publisher p, $CTR(a,p) = \frac{clicks_{a,p}}{views_{a,p}}$, where $clicks_{a,p}$ and $views_{a,p}$ denote the total numbers of clicks and views on a, respectively, through the websites belonging to p; (3) the CTR of an ad a in the profile c, $CTR(a,c) = \frac{clicks_{a,c}}{views_{a,c}}$, where $clicks_{a,p}$ and $views_{a,p}$ denote the total numbers of clicks and views on a, respectively, from the users in profile c or its descendant.

In our second countermeasure against underreporting advertisers (See Sec. V-A), clients are made to send the broker perturbed click reports, based on which the broker can estimate the total number of clicks on each ad. Similarly, we can make the clients send perturbed view reports to the broker to enable him to estimate the total number of views on each ad. So, it becomes straightforward for the broker to compute the first type of CTRs.

The second type of CTRs cannot be obtained in the same way because the users' click/view reports do not include any information on publishers due to the privacy concern. The broker, however, may make use of the reports of publisher p to derive $CTR(a,p)$. As we have introduced earlier, the click reports from a publisher to the broker include both the publisher ID and the ad ID. As a result, it is straightforward for the broker to learn $clicks_{a,p}$ based upon the click reports from p. We can also make publishers relay similar view reports to the broker when ads are displayed in the browsers of users. Then, the broker can get $views_{a,p}$ as well, and further compute the value of $CTR(a,p)$.

The last type of CTRs are the most difficult to obtain since no reports of users, publishers or advertisers contain any information on users' contexts for the purpose of protecting user privacy. It is impossible for the broker to utilize these reports to compute $CTR(a,c)$. We therefore design another solution to this challenge.

First, we make each client maintain a log to record every ad click/view event as well as the user's realtime profile as this event occurs. Then, when the broker wants to estimate the value of $CTR(a,c)$, he creates a query taking a and c as the parameters, and delivers it to a wide range of online clients. Once a client i receives this request, it sends an answer $E_{k_B}(clicks_c||views_c)$, which is encrypted by the public key k_B of the broker, to the advertiser owning ad a. Here, we denote by $clicks_c$ and $views_c$ the numbers that the user of i has clicked and viewed on a, respectively, in profile c according to the log maintained by i. The advertiser accumulates such answers and forwards them to the broker once the count reaches some threshold. Certainly, the advertiser has to remove all the identity information (including the IP addresses) and mix these answers before sending them to the broker. The broker can then compute $CTR(a,c)$ after decrypting and aggregating the received messages. This method can also be used to compute the first two kinds of CTRs.

Note that a malicious advertiser may replace real answers with fake ones to drive up the CTRs of his ads, which will place these ads in a better position in the future ad auctions. We can leverage the similar sampling-based detection method of underreporting advertisers to deal with this threat. Specifically, the broker picks a small proportion of the clients being requested as auditors, and assigns each of them a unique nonce for auditing the advertiser. These auditors do nothing but encrypt the received nonce and send them to the advertiser as normal reports. If the number of nonce reports that the broker receives finally is smaller than the number of auditors, this advertiser is suspected of having replaced the answers of some clients. In addition, if the number of nonce reports is correct but the proportion is lower than that of auditors, the broker suspects the advertiser of adding additional fake reports.

Furthermore, the last aggregation scheme does not provide users with any differential privacy guarantees [5], [6], i.e., the broker can reveal a specific user's answer by comparing the aggregation results of different queries. For instance, if the broker wants to learn the answer of a specific user u_i to a query, he may make this query twice among a group of users containing u_i and the same group without u_i, respectively. The difference between the obtained results is obviously the answer of u_i. Akkus et al. [3] have carefully studied a similar problem and we may leverage their elegant approach to counter this attack by introducing a few noisy answers at the advertiser. We omit the details here due to the space limitation.

VI. Evaluation

In this section, we mainly use simulations to evaluate the effectiveness of the proposed system for achieving the security goal during the billing period.

As we introduced in Sec. III-A, the security goal is composed of four subgoals. Since in our proposal, all the billing decisions are made based up the reports of advertisers and publishers, it can naturally prevent malicious brokers from overcharging advertisers or underpaying publishers by cheating in his counts of ad clicks. For the second subgoal, advertisers cannot deny the bills from the broker because these bills are generated based on the click reports singed by themselves, which means they agree with these reports. However, they may underreport ad clicks related to them to reduce the bills. For the third subgoal, as the click reports from a publisher to the broker are signed together with a time interval, the publisher is not allowed to claim additional clicks within the same interval once he submits the signed message to the broker. However, the reported clicks may include fake ones for driving up the revenue shares of the publisher. Therefore, the realization of these two subgoals relies on a premise that the broker can effectively detect the underreporting and overreporting behaviors of advertisers and publishers, respectively. Otherwise, our proposal is unfair to the broker. We below use simulations to evaluate the effectiveness of the proposed countermeasures against them. For the subgoal of detecting click fraud, since we just adapt traditional countermeasures to the proposed system and do not propose any new ones, it is meaningless to evaluate it.

A. Evaluation of Countermeasures Against Underreporting Advertisers

We first consider the sampling-based countermeasure against the underreporting attacks of advertisers. This countermeasure is also used to defend against malicious advertisers in the statistics gathering, which is described in Sec. V-D. We assume that the ads of an advertiser are clicked on by 10,000 users, and this advertiser drops reports at a fixed rate of p. We vary the number of verifiers from 10 to 145 in a step of 5. Fig. 3 presets the average rates that the broker detects the undereporting behaviors under different p. We can see that if the broker triggers 60 auditors, the detection rate can exceed 90% provided the dropping rate is greater than 4%. Moreover, this performance is not affected by the number of ad clicks.

We then evaluate the performance of the second countermeasure against this attack. We assume that there are 50000 users having clicked on ads related to "Movie". These users use the preference "Movie" instead of the five more concrete preferences shown at the bottom of Fig. 2 to request ads from the broker. We assume that the broker returns them the same set of 100 ads that are expected to maximize his revenue. Each of these users and ads randomly picks one of the five concrete preferences as the profile of interest. The ads are then ranked on each user' machine based on their bids and the results of profile comparing. The first ranking ad is assumed to be clicked by the user, and the client will provide the broker

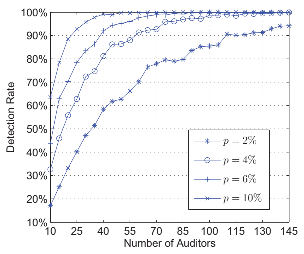
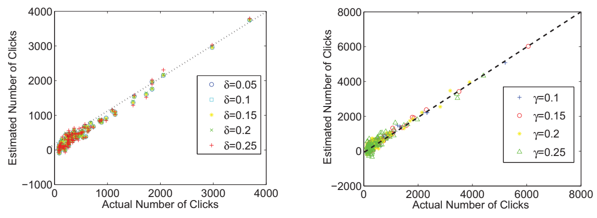


Fig. 3. The detection rate of the sampling-based countermeasure against underreporting advertisers



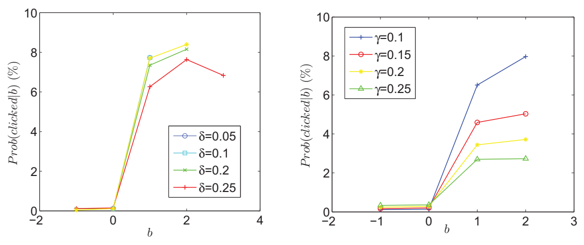
(a) $\gamma = 0.1$ (b) $\delta = 0.25$

Fig. 4. Comparison of Actual and Estimated number of ad clicks

with a perturbed click report as we described in Sec. V-A. The broker finally leverages the received reports to estimate the total number of clicks for each ad based on Equation 2.

We first set $\gamma = 0.1$ and vary the value of δ from 0.1 to 0.3 in the step of 0.05. We round all the random noises δ to integers. Fig. 4(a) shows the actual versus the estimated number of clicks for each ad. Note that the dotted line represents the exact value. We see that the accuracy decreases as the variance δ increases, but when $\delta \leq 0.25$ all points are quite clustered around this line. The estimation error is small enough for detecting underreporting advertisers. We then set $\delta = 0.2$ and vary γ among 0.05, 0.1, and 0.2, the new result is present in Fig. 4(b), we see that the accuracy also decreases as γ increases.

We also evaluate the effects of this countermeasure on users' privacy. For this purpose, we transverse the perturbed reports to compute $Prob(clicked|b)$, the probability that an ad has been really clicked on if the observed value of its corresponding element in a report is b. Obviously, if clients directly report their original binary vectors to the broker without any perturbation, $Prob(clicked|1) = 100\%$, i.e., the broker can learn exact which ad has been clicked on. Fig. 5 presents the results after applying our perturbation strategy. We also vary δ and γ, respectively. We can find that $Prob(clicked|b)$ is now confined to below 9% for any value of b. As a result, it is difficult for the broker to identify which ad has been clicked based on the observed reports. In addition, as δ and γ increase, the privacy is better protected. However, as we mentioned earlier the estimation accuracy decreases. We thereby need to make a trade-off between these two properties when we employ this mechanism to defend against advertiser underreporting.



(a) $\gamma = 0.1$ (b) $\delta = 0.25$

Fig. 5. Effects of the second countermeasure against underreporting advertisers on users' privacy ($Prob(clicked|b)$ is the probability that an ad has been clicked if the observed value of the corresponding element in a click report is b)

B. Evaluation of the Countermeasure Against Overreporting Publishers

We next evaluate the effectiveness of Algorithm 1 to detect overreporting publishers. We use the same setup on ads and users with the last simulation. We also assume there are 500 publishers, and the 50000 clicks are randomly assigned to these publishers. For each ad click, we create legal reports for both the advertiser and the publisher. We then modify these reports to simulate two kinds of overreporting attacks:

(1) We create 500 additional reports for a specific publisher, which simulates the case that this publisher has launched a simple overreporting attack. The ad IDs within these reports are randomly selected from 100 ads.
(2) We randomly select 500 reports of 20 advertisers and replace the publisher IDs within them with the ID of a specific publisher P_x. We also create 500 additional reports for this publisher using the ad IDs of the 500 advertiser reports. This simulates the case that P_x has overreported by colluding with 20 advertisers.

We then run Algorithm 1 to verify each publisher with the original report set and the above modified versions. We consider $a \approx b$ if $|a - b| \leq 300$. The result shows that it successfully captures both the above two overreporting publishers, while produces no false positives.

VII. Conclusion

In this paper, we have proposed a novel OBA system reversing the priority between the broker and the advertisers and publishers. Specifically, when users click on ads, this system makes related advertisers and publishers forward click reports of users to the broker after verifying, anonymizing and signing them. The broker then settles accounts fully based on these reports. To protect the interests of the broker after the priority inversion, we propose effective mechanisms for detecting underreporting and overreporting attacks of malicious advertisers and publishers, respectively. We show that this system can naturally prevent malicious brokers from launching any overcharging or underpaying attacks, and enable legitimate brokers to justify their accounts. Moreover, it can also protect the user privacy in the billing stage since clients now do not directly contact with the broker for reporting ad clicks.

References

[1] Advertiser sues microsoft for click fraud. http://www.medialifemagazine.com/advertiser-sues-microsoft-for-click-fraud/.
[2] Google and yahoo! accused of click fraud collusion. http://www.theregister.co.uk/2005/04/05/google_and_yahoo_accused_of_click_fraud_collusion/.
[3] Istemi Ekin Akkus, Ruichuan Chen, Michaela Hardt, Paul Francis, and Johannes Gehrke. Non-tracking web analytics. In *Proceedings of the 2012 ACM conference on Computer and communications security (CCS'12)*, pages 687–698. ACM, 2012.
[4] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. Obliviad: Provably secure and practical online behavioral advertising. In *Proceedings of the 33rd IEEE symposium on security and privacy (SP'12)*, pages 257–271. IEEE, 2012.
[5] Cynthia Dwork. Differential privacy. In *Automata, languages and programming*, pages 1–12. Springer, 2006.
[6] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Advances in Cryptology-EUROCRYPT 2006*, pages 486–503. Springer, 2006.
[7] Matthew Fredrikson and Benjamin Livshits. Repriv: Re-envisioning in-browser privacy. In *Proceedings of the 32nd IEEE symposium on security and privacy (SP'11)*, pages 131–146. IEEE, 2011.
[8] Eric Goldman. Click fraud. In *Proceedings of the 20th Annual Technology and Computer Law Conference*, pages 24–25, 2007.
[9] Saikat Guha, Bin Cheng, and Paul Francis. Privad: practical privacy in online advertising. In *Proceedings of the 8th USENIX conference on networked systems design and implementation (NSDI'11)*, pages 13–13. USENIX Association, 2011.
[10] Saikat Guha, Alexey Reznichenko, Kevin Tang, Hamed Haddadi, and Paul Francis. Serving ads from localhost for performance, privacy, and profit. In *Proceedings of the 8th Workshop on Hot Topics in Networks (HotNets 09)*, 2009.
[11] Hamed Haddadi. Fighting online click-fraud using bluff ads. *ACM SIGCOMM Computer Communication Review*, 40(2):21–25, 2010.
[12] Hamed Haddadi, Saikat Guha, and Paul Francis. Not all adware is badware: Towards privacy-aware advertising. In *Proceedings of 9th IFIP conference on e-Business, eServices, and e-Society*, pages 161–172. Springer, 2009.
[13] Michaela Hardt and Suman Nath. Privacy-aware personalization for mobile advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security (CCS'12)*, pages 662–673. ACM, 2012.
[14] Nicole Immorlica, Kamal Jain, Mohammad Mahdian, and Kunal Talwar. Click fraud resistant methods for learning click-through rates. In *Internet and Network Economics*, pages 34–45. Springer, 2005.
[15] Ari Juels. Targeted advertising... and privacy too. In *Topics in CryptologyłCT-RSA 2001*, pages 408–424. Springer, 2001.
[16] Murali Kodialam, TV Lakshman, and Sarit Mukherjee. Effective ad targeting with concealed profiles. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12)*, pages 2237–2245. IEEE, 2012.
[17] Balachander Krishnamurthy and Craig E Wills. Cat and mouse: content delivery tradeoffs in web access. In *Proceedings of the 15th international conference on World Wide Web (WWW'06)*, pages 337–346. ACM, 2006.
[18] Juan-Carlos Perez. Google launching tools to fight click fraud. http://www.medialifemagazine.com/advertiser-sues-microsoft-for-click-fraud/.
[19] Alexey Reznichenko, Saikat Guha, and Paul Francis. Auctions in do-not-track compliant internet advertising. In *Proceedings of the 18th ACM conference on computer and communications security (CCS'11)*, pages 667–676. ACM, 2011.
[20] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings of the 17th Annual Network & Distributed System Security Symposium (NDSS'10)*, 2010.