ObliviAd: Provably Secure and Practical Online Behavioral Advertising

Michael Backes*†, Aniket Kate*, Matteo Maffei†, and Kim Pecina†

* *MPI-SWS, Germany* {backes,aniket}@mpi-sws.org

† *Saarland University, Germany* {maffei,pecina}@cs.uni-saarland.de

***Abstract*—Online behavioral advertising (OBA) involves the tracking of web users' online activities in order to deliver tailored advertisements. OBA has become a rapidly increasing source of revenue for a number of web services, and it is typically conducted by third-party data analytics firms such as brokers, which track user behaviors across web-sessions using mechanisms such as persistent cookies. This practice raises significant privacy concerns among users and privacy advocates alike. Therefore, the task of designing OBA systems that do not reveal user profiles to third parties has been receiving growing interest from the research community. Nevertheless, existing solutions are not ideal for privacy preserving OBA: some of them do not provide adequate privacy to users or adequate targeting information to brokers, while others require trusted third parties that are difficult to realize.**

In this paper, we propose ObliviAd,[1] a provably secure architecture for privacy preserving OBA. The distinguishing features of our approach are the usage of secure hardware-based private information retrieval for distributing advertisements and high-latency mixing of electronic tokens for billing advertisers without disclosing any information about client profiles to brokers. ObliviAd does not assume any trusted party and provides brokers an economical alternative that preserves the privacy of users without hampering the precision of ads selection.

We present the first formal security definitions for OBA systems (namely, profile privacy, profile unlinkability, and billing correctness) and conduct a formal security analysis of ObliviAd using ProVerif, an automated cryptographic protocol verifier, establishing the aforementioned security properties against a strong adversarial model. Finally, we demonstrated the practicality of our approach with an experimental evaluation.

I. Introduction

Today, a large majority of online services garner most of their revenues through advertising. Online advertisements, however, are not effective unless they are targeted to the right audience. As a result, online advertisements are no longer shown in a scattershot way but rather targeted to reach a clientele that is selected according to various traits such as demographics, previously visited urls, the current web page, information stored in cookies, and, in general, any kind of observed behavior.

[1]The name is derived from Obliviate! and Ad. Obliviate! is a charm from the famous Harry Potter books that lets an individual (in our case the broker) forget a specific memory (in our case all user-related data).

A. Online Behavioral Advertising

An online behavioral advertising (OBA) system consists of four principal players: the *advertisers*, *brokers*, *publishers*, and *clients*. Advertisers want the ads for their products to reach plausible clients (e.g., Ford wants to inform those people interested in buying a car about a new model) and they are willing to pay for this service. Publishers (e.g., newspapers such as the New York Times) are willing to place ads on their webpages, but they expect to be payed in return. Brokers (e.g., Google and Yahoo) mediate between advertisers, publishers, and clients: the advertisers send their ads and bids to the broker, who then distributes them to the publishers' webpages, which are in turn viewed by the clients. The client may click on an ad if she finds it to be relevant to her needs or interests. For every ad viewed in the *pay-per-view* (PPV) advertising model or clicked in the *pay-per-click* (PPC) advertising model, the advertiser pays the broker who in turn pays the publisher. The PPC, being a pay-for-performance model, receives more attention in the industry. OBA protocols are divided in two phases: the *distribution phase*, in which the brokers distributes the ads, and the *tallying phase*, in which the broker computes the number of viewed (or clicked) ads for billing purposes.

In the existing online behavioral advertising (OBA) systems, publishers embed a link for the broker on their webpages. The sole purpose of these broker links is user tracking. When a user views the webpage, the user's browser contacts the broker's servers, which enables the broker to track the user across all partnering publishers. The broker then runs its own algorithm over the tracked data to decide which ad to present on the publisher's page. This tracking practice poses a significant threat to the privacy of users as research has shown that it is often easy to link tracked information with an individual's personally identifying information (PII) [32], [33].

There have been a number of proposals for addressing these privacy concerns. Consumers groups have suggested that the behavioral advertising industry undergo

regulatory reform, with the goal of enhancing users' privacy and data protection by constraining the role and the activities of third-parties such as advertisers and brokers [14], [30]. However, the effectiveness of such a reform is questionable, since it is hard to prevent third parties from passively gathering PII and later destroying any traces in the case of a future investigation [31]. Initiatives like Do Not Track [35] enable users to opt out of tracking by analytics services, advertising networks, and social platforms. These initiatives do not solve the core problem, however, as they significantly hamper the economic model of free online services that depend solely on advertising revenue. Furthermore, they prevent users from viewing ads that are relevant to their current needs and interests. Although anonymous browsing solutions such as Tor [48] can also enforce the desired privacy goals, they make detection of client-side fraud (e.g., click fraud[2]) infeasible. Moreover, they are not considered scalable enough to support the existing user base [37]. Thus, there is still need for a scalable solution that simultaneously satisfies the financial goals of web services and the privacy requirements of users.

B. Our Contributions

We present ObliviAd, a novel OBA architecture that preserves the privacy of user profiles, is compatible with the existing business model, supports arbitrary ad-selection algorithms, and provides high performance. The distinguishing features of our architecture are the usage of secure hardware-based *private information retrieval* (PIR) for distributing advertisements and high-latency mixing of *electronic tokens* for billing advertisers without disclosing any information about client profiles to brokers. Following the approach proposed by Williams and Sion [51], we implement our PIR protocol using a secure coprocessor (SC), such as the 4765 cryptographic coprocessor by IBM [27], which provides secure storage for sensitive data as well as a trustworthy,[3] programmable execution environment. This solution ensures the privacy of data as well as the integrity of the computation, even if the broker is malicious.

In our protocol, while fetching an ad, the user sends her profile in encrypted form to a secure coprocessor that resides on the broker side. The broker may learn the identity of the user, since we do not assume any anonymous channel, but not the user profile. The SC selects the advertisement that best fits the user profile according to the algorithm specified by the broker. We build on a state-of-the-art *oblivious RAM* (ORAM) protocol [43], to prevent the broker from learning any information about the selected advertisement. We modify this ORAM scheme to handle multiple entries per keyword, i.e., multiple different advertisements can be stored and retrieved for a single keyword, and we prove our modifications secure. The advertisement is finally shipped in encrypted form to the user along with a fresh electronic token, i.e., a signed piece of data comprising a sequential timestamp and the symmetrically encrypted advertisement id. The creation of such tokens causes only a minimal computational overhead. As soon as the ad is clicked or viewed, depending on the business model, the token is sent back to the broker. After gathering a sufficiently large number of tokens, the broker passes the tokens to the SC, which decrypts, mixes, and finally publishes them in order to charge the advertisers.

Our cryptographic construction preserves the *privacy of the user profile*, which implies that the broker does not learn which advertisement is shown to which user, and *unlinkability* of subsequently observed units of information. In addition, our protocol fulfills a number of broker and advertiser desiderata, such as *client-side fraud prevention*, *click success measures*, and *high performance*. In fact, the motivation behind ObliviAd is not to replace the existing OBA infrastructure. It rather aims at complementing the existing infrastructure so that PII of users is secured without advertisers or brokers being forced to change their business model. Our system does not introduce significant computational overhead or financial costs to either the users or the broker. We avoid fancy cryptography at the user ends as we want our solution to work even for web browsers on handheld devices. ObliviAd design is scalable, since brokers can utilize multiple secure coprocessors, thus improving the performance without altering the privacy properties of our system. We believe that the privacy guarantees that the broker may demonstrate to the privacy-conscious user base make the investment into ObliviAd a worthwhile venture.

We formalize two fundamental privacy properties (namely, privacy of the user profile and profile unlinkability) as observational equivalence relations and the correctness of the billing process as a trace property. We conduct a formal security analysis of ObliviAd using ProVerif, an automated cryptographic protocol verifier, establishing the aforementioned security properties against a strong adversarial model. We believe both the definitions and the security analysis are of independent interest, since they are the first for an OBA system.

We also demonstrate the practicality of ObliviAd by performing an experimental evaluation of our proof-of-concept implementation.

[2]Click-fraud consist of users or bots clicking on ads in order to drive up a given advertiser's costs, a publisher's revenue, the click-through rate of an advertisement, and so on.

[3]SCs offer a remote code attestation procedure that allows clients to verify what code is being executed [46].

C. Organization

The rest of the paper is organized as follows. In the next section, we define the problem of non-tracking behavioral advertising. Section III presents an overview of the ObliviAd architecture. In Section IV, we describe our ORAM construction in details. We analyze the performance of our system in Section V. In Section VI, we conduct a formal security analysis. Section VII discusses the related work and Section VIII concludes.

II. Privacy-Preserving OBA

In this section, we describe the privacy goals and system properties that are desirable in a privacy-preserving OBA system (Section II-A) and we specify the threat model considered in this paper (Section II-B).

A. Privacy and System Goals

A privacy-preserving OBA system should achieve the following privacy goals [41]:

Profile Privacy. The broker cannot associate any unit of learned information (e.g., clicked ads) with any user PII (including network address).[4]

Profile Unlinkability. The broker cannot associate separate units of learned information with a single client.

The former property is analogous to vote privacy in the setting of electronic voting, i.e., the impossibility of associating a vote in the final tally with a specific voter. The latter property prevents a broker from building up a user profile and then associating it with a known user using externally gathered information. For instance, even if the broker knows that two users have seen two ads each and the four ads comprise two car ads and two football ads, the broker does not know whether the two car-ads were seen by the same user or not, i.e., whether or not the two users have disjoint interests.

In principle, both privacy goals can be trivially satisfied by any anonymous browsing solution [2], [48]. Existing anonymity networks, however, have two drawbacks: they do not provide adequate performance (ads should be displayed almost instantaneously) and they make the users unaccountable [7], whose implications are not acceptable to the ad industry. In general, an OBA solution needs to satisfy the following system properties.

Client-side Fraud Detection. The likelihood of detection of clients' malicious behaviors should not decrease as compared to existing systems.

Click Success Measures. Computations of success measures such as click-through rate [19] or click-probability [41] should be possible on the broker or client's side.

Performance. Privacy-preserving mechanisms should not hamper the system performance and the auction mechanism should achieve close-to-ideal ranking of ads.

B. Adversary Model

We assume an active adversary with read and write capabilities on the public network, on the *SC*-to-database bus, and on the database itself. The adversary exercises full control over the broker[5] and the publisher can issue arbitrary requests to the secure coprocessor (*SC*), obtain the respective response, and observe the resulting operations on the database. The trusted computing base is limited to the management of key material within the *SC*; i.e., we assume that secret keys are not leaked. The integrity of the code executed by the *SC* can be enforced, since modern secure coprocessors offer a remote code attestation procedure [46] that gives clients the ability to verify that the *SC* is executing a certain code. In our architecture, this server-side code is made public for peer scrutiny.

Unlike other privacy-preserving advertising systems [25] where brokers are assumed to be honest-but-curious, we do not make any assumptions about them. We also allow the attacker to arbitrarily corrupt or create client principals and act on their behalf.

In the case of the user clicking on the retrieved ad, we have to assume that the advertiser and the brokers are not colluding; profile privacy is otherwise impossible without using an external anonymity solution such as Tor [48] or Anonymizer [2], since the client reveals her identity to the broker when retrieving an advertisement and clicking on it reveals her identity along with the retrieved advertisement, i.e., her profile, to the advertiser.

III. Protocol Overview

We first introduce the cryptographic concepts that are required for the understanding of our system (Section III-A). We then describe the cryptographic assumptions and requirements of our system (Section III-B). Finally, we present a high-level protocol description (Section III-C) and discuss the most important properties of our architecture (Section III-D).

A. Preliminaries

Digital Signatures and Encryption Schemes. Our construction requires an existentially unforgeable digital signature scheme [22] and an authenticated encryption scheme (i.e., INT-CTXT and IND-CPA secure [5]).

We do not rely on any particular digital signature or encryption scheme. In fact, our construction is fully

[4] In [41], this property is termed anonymity, which we find to be slightly inaccurate as users need not be anonymous here.

[5] Assuming the broker is a malicious (as opposed to honest but curious) party is reasonable, since the user cannot select the broker and, in fact, the identity of the broker is not known to the user.

parametric in these two cryptographic primitives, inasmuch as they satisfy the respective security definitions.

Oblivious RAM (ORAM). Oblivious RAM was originally devised to protect the access pattern of software on the local memory and thus to prevent the reverse engineering of that software [21]. The observation is that encryption by itself prevents an attacker from learning the content of any memory cell but monitoring how memory is accessed and modified may still leak a great amount of sensitive information.

In the ORAM model, the processor executing a program is considered a black box, i.e., it is impossible to observe the processor's internal state, internal storage, and internal operations. The external storage and the bus connecting that storage with the processor are observable and ORAM schemes use sophisticated combinations of data structure and cryptographic operations to mask their access pattern on the external storage. In our construction, the ORAM scheme is running on a secure coprocessor (SC), which enforces the black-box characteristics. Clients contact the black box via a secure channel (e.g., TLS) to prevent an attacker from obtaining any information on the requested operation. The ORAM storage is organized as a data structure such that every entry contains a keyword kw and a payload (i.e., an ad and possibly additional information in our case). ORAM schemes export two methods, namely $\mathsf{Read}(kw)$ and $\mathsf{Write}(kw, ad)$. The former returns the list of ads associated with kw and, for access privacy reasons, removes the corresponding entries from the data structure; the latter adds the entry (kw, ad) to the data structure.

Private Information Retrieval (PIR). Private information retrieval schemes allow a client to access a database stored on a server, while hiding the query and the resulting answer from the database [12]. In this work, we use a PIR scheme to allow the client to download relevant ads from the broker, without the broker learning any information about such ads or the user profile. PIR can be achieved in either an information-theoretic or a computational setting. Information-theoretic PIR requires multiple non-colluding database servers, and the non-collusion assumption is certainly inappropriate for broker servers. Computational PIR, instead, relies on cryptographic assumptions, is suitable also for a single database server, and thus fits our setting better. According to recent analyses, however, none of the existing computational PIR schemes significantly outperforms the trivial solution of downloading the whole database (i.e., all advertisements in our case) [38], [44]. For this reason, we follow the approach proposed by Wang et al. [17], [50] and Williams et al. [51], which consists of implementing a computational PIR scheme using ORAM over a secure coprocessor (SC). This solution turns out to be orders of magnitude faster than the other computational PIR solutions. However, as we discuss in Section IV, the previously mentioned ORAM constructions are not useful for OBA systems. Our construction instead builds on the ORAM protocol recently developed by Shi et al. [43], which we modify to fit our needs and, in particular, to run on a SC and to associate an ad with multiple keywords.

This solution is also well-suited for clients with only little computational power such as cell phones and netbooks, because the main work (i.e., the cryptographic operations) is performed by the SC, which has dedicated cryptographic hardware and resides on the broker's side; the client has merely to establish a secure connection to the SC, e.g., via TLS, send the query, and receive the result.

Electronic Tokens. Electronic tokens are the digital equivalent of real-world money, i.e., it is impossible or, at least, computationally infeasible to fake them; a token by itself reveals neither its spender nor what it was spent on; and double-spending a token is detectable. In our construction, electronic tokens enforce the correct billing of the advertiser, while preventing brokers from tracking the respective user.

Electronic tokens may resemble electronic coins [10], [11]. Our tokens, however, are purely based on highly efficient symmetric encryption and digital signature schemes.

Mixing. The concept of mixing was introduced by Chaum [9]. Here, we use it to prevent the attacker from learning the correlation between the content of electronic tokens and the respective users. Specifically, the broker provides the SC with a set of (symmetrically) encrypted tokens containing the ad identifiers. The SC decrypts those tokens on behalf of the broker. It also randomly permutes (or mixes) the resulting ad identifiers in order to maintain profile privacy and profile unlinkability.

B. Cryptographic Assumptions and Requirements

We assume a publicly verifiable binding between the SC and its public key. Such a binding is easily possible with a chain of certificates with the root public key stored in the user's browser. Using this binding, the client software can establish authenticated and encrypted TLS connections with the SC.

Our broker-side code must be executed in a trusted environment, and it requires a rich set operations, e.g., file I/O, data structure management, TLS connections, digital signatures and authenticated encryptions; thus, we need a programmable SC [47]. Furthermore, to guarantee that a SC is executing a correct program, we also expect a remote attestation capability from the SC [46].

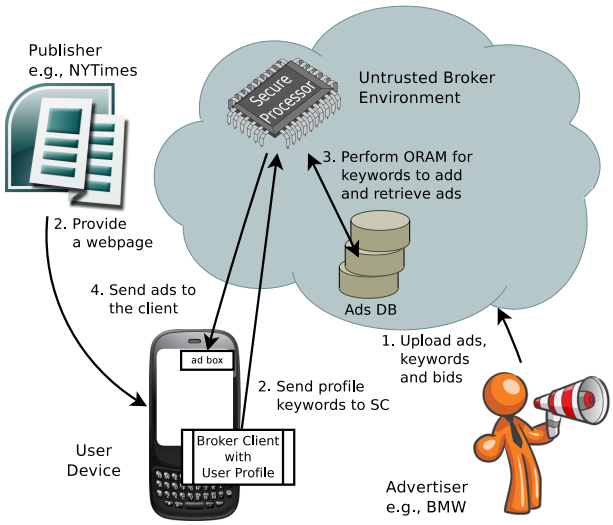

Figure 1. Distribution phase

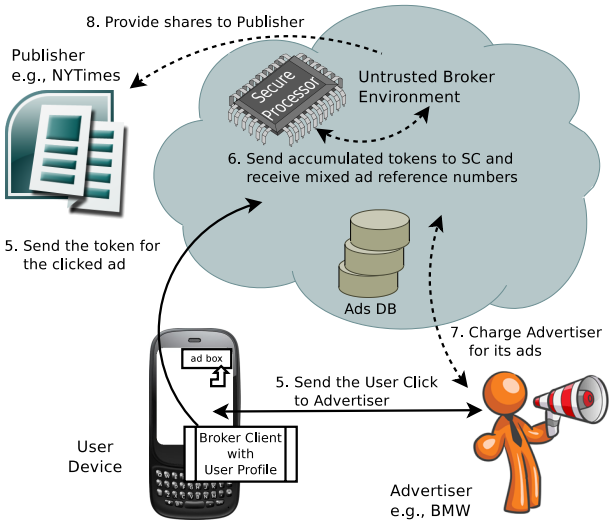

Figure 2. Tallying phase

C. Protocol Overview

We now overview the cryptographic protocol underlying ObliviAd.

1) The advertisers initiate the protocol by uploading each of their ads ad, the corresponding keywords kws, and other information (e.g., bids) to the broker server. The broker forwards these tuples to the SC along with unique ad identifiers. The SC includes these tuples in the ORAM structure, i.e., it stores tuples containing the advertisement ad, the corresponding identifier id_{ad}, and one keyword kw_{ad} in encrypted form on the broker's server. Notice that the number of triples referring to the same ad is the same as the number of keywords associated with that ad. Publishers that are interested in showing ads on their webpages must also register with the broker as they are interested in showing ads on their webpages.
2) When a user visits a publisher's webpage containing an ad box, the broker's client program on the user machine is invoked. This program maintains the user's profile in the form of keywords kws_U based on the user's online behavioral history and sends those keywords to the SC over a secure and (server-side) authenticated channel.
3) The SC then searches the ORAM structure for kws_U, collects the resulting ads, and selects a subset according to the ranking algorithm specified by the broker, which usually takes into account a number of factors, such as bids, click-probabilities, and so on. Finally, the SC attaches an electronic token to each selected ad for the future accounting. The electronic token for the advertisement ad is of the form $\mathsf{sig}(\mathsf{enc}(id_{ad}, kw_{ad})_k, t)_{sk_{SC}}$, i.e., it consists of a digital signature produced by SC on (i) the ciphertext obtained by encrypting the ad's identifier id_{ad} and the corresponding keyword kw_{ad} with a symmetric key k, which is chosen by the SC and kept secret, and on (ii) a timestamp t (or, alternatively, on an increasing number).
4) Once the SC has finished its processing, it sends ads and associated tokens to the client software over the secure and authenticated channel. The client software then presents a selection of these ads to the user.
5) When the presented ad is viewed by the user in the PPV model or is clicked in the PPC model, the client software sends back the token to the broker server.
6) Following the mixing methodology, the broker server accumulates the tokens over a predefined billing period and sends the set of accumulated tokens to the SC. The SC removes duplicates (i.e., tokens with the same timestamp), removes the tokens with timestamps outside of the current billing period, decrypts the ads in the remaining tokens, and publishes a random permutation thereof.
7) The broker then distributes these identifiers to the corresponding advertisers and charges them accordingly. The ad keywords retrieved from the tokens may be used for improving future auctions, e.g., for further click-through analysis.
8) Finally, the broker provides revenue shares to publishers.

D. Discussion

Role of the Client. Similarly to other privacy preserving OBA architectures (e.g., Adnostic and Privad), we

assume that user profiles, encoded as sets of keywords, are created and managed in the broker's client software, which is envisioned as a browser extension on the user's device. The client monitors user behavior (i.e. the user's browsing, ads viewed and clicked and so on) in order to create and maintain the user profile. We assume that the client is not compromised and specifically does not leak the keywords or the ads to the broker. Although our system is flexible in the choice and implementation of the client, privacy preserving browser-based mining of core interests can be performed using, for instance, the recently introduced RePriv platform [20].

Role of the SC. The secure coprocessor establishes a secure program execution environment on the server. A certificate signed by a certificate authority ensures the user that she is communicating with the *SC* and a remote code attestation procedure [46] ensures the user that the correct program is running.

Privacy of the User Profile. User profiles are transferred to the *SC* in encrypted form over secured channels, and are therefore not visible to any third party. The ORAM architecture on the broker's side ensures that not even the selection of ads leaks any information about user profiles to the broker. Although the broker may learn which electronic token was processed by which user, since we do not assume anonymous channels, the broker cannot learn which electronic token corresponds to which ad, thanks to the mixing performed by the *SC*. It is interesting to observe that the degree of privacy of the user profile is determined by the number of electronic tokens that are provided by non-compromised clients in the respective mixing procedure, given that the tally of the ads must be made public for billing purposes. If all other electronic tokens are provided by the attacker, the privacy of the user profile cannot be guaranteed. This is reminiscent of electronic elections, where the privacy of the user vote cannot be guaranteed if all other voters are under the control of the attacker, given that the final tally is public [16].

Notice that a malicious broker could in principle derive a particular user profile by allowing only the tokens of that honest client to reach the *SC*. The resulting bill would reveal the user profile, thus breaking the desired privacy property. For protecting client profiles unconditionally, the usage of anonymous channels is indispensable, a solution we do not advocate due to its computational cost and network delay. Typical brokers, however, behave rationally, i.e., their primary goal is to excel in commerce rather than to identify users at all costs. Our scheme protects the privacy of the user profile against rationally-behaving brokers: excluding user tokens from the tally leads to a significant monetary loss, since the timestamp mechanism prevents those tokens from being counted in the next tallying periods. It is interesting to observe that privacy-aware users can get perfect privacy (i.e., even against non-rational brokers) in the pay-per-click model, which is the most common one, by simply avoiding clicking on ads.

Profile Unlinkability. Not only is the broker not able to learn which user saw which ad, but it cannot even learn whether or not two or more ads were seen by the same user. This property is enforced by (*i*) the structure of the electronic tokens, which are unlinkable and do not reveal any information about the user profile, and (*ii*) the mixing, which breaks the correlation between the list of tallied ads and the list of received tokens. Breaking this correlation is crucial since the attacker may learn the correlation between tokens and clients by looking at the traffic on the non-anonymous communication channel between clients and *SC*.

Billing Correctness. The timestamp mechanism also serves the purpose of ensuring the correctness of the billing process, since each ad cannot be counted more than once in the final tally and it is counted only if the client has forwarded the electronic token to the broker, i.e., the user has viewed (or clicked) the ad.

Click-related Information. Without using an anonymous browsing solution, which we do not want to adopt for efficiency reasons and for the sake of click-fraud detection (see below), it is impossible to prevent the advertiser from learning which user clicked which ad. In practice, the broker may try to collude with the advertiser to obtain this information. If OBA is the only goal of the broker, however, there is no motivation for the broker to determine which user clicked a particular ad. The information required to hold auctions, such as the click-through rate or the click-probability of an ad, is derivable by the broker from the tally produced by the *SC*.

Click-Fraud Detection. In our design the interaction pattern between the clients and the broker remains almost unchanged and the broker is still notified when the client clicks on an ad, although it does not know which ad was clicked. Thus, real-time click-fraud detection mechanisms, which typically monitor the click-ratio of each user, are expected to continue to work. Offline detection mechanisms are expected to continue to work as well, since they are typically enforced on the advertiser side and, in our architecture, advertisers know who clicked their ads.

IV. ORAM Construction

We adopt the ORAM scheme by Shi et at. [43], which we modify to fit the OBA setting. We first review their basic construction (Section IV-A) and later discuss our modifications (Section IV-B).

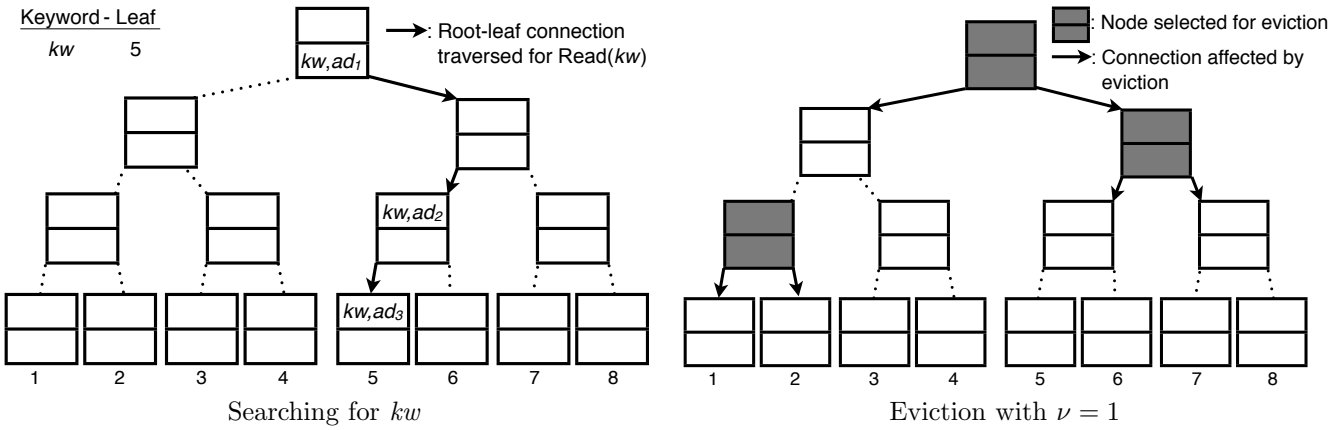

Figure 3. ORAM Operations for a bucket size of 2 and a tree depth of 4.

A. ORAM Scheme by Shi et al.

Shi et al. propose three different constructions. In this work, we adopt their "basic construction with trivial buckets". The complete database is stored as a binary tree. Every node in the binary tree is a bucket ORAM, i.e., an array of entries. Each entry consists of a keyword kw and a payload in encrypted form. An authenticated encryption scheme is used to prevent an active attacker from learning and modifying entries in an unobservable way.

Initially, the bucket is filled with dummy entries. The ORAM scheme manipulates buckets via the $\mathsf{ReadAndRemove}(kw, b)$ and $\mathsf{Add}(b, e)$ operations. Both traverse the complete bucket b, reading and decrypting every entry e_{old}, and replacing e_{old} with a new entry e_{new}. In case of $\mathsf{ReadAndRemove}(kw, b)$, this new entry is either a re-randomization of e_{old} if the keyword does not match e_{old}, or an encryption of a dummy entry otherwise. In case of $\mathsf{Add}(b, e)$, this new entry is either e if e_{old} is a dummy entry and e has not been stored yet, or a re-randomization of e_{old} otherwise. The IND-CPA indistinguishability property of the encryption scheme ensures that the adversary cannot differentiate between a dummy entry and an entry comprising a kw and a payload.

The ORAM construction maintains the invariant that the entries for every keyword kw are located on a unique path from the root node to a leaf ℓ assigned to that keyword. This keyword-leaf (kw, ℓ) assignment is securely maintained in the ORAM program (i.e., in the SC in our case).

To read the entry for a keyword kw, the Read operation (cf. Section III-A) traverses the complete path from the root node to the leaf assigned to kw, searching for a bucket containing kw (see Figure 3). The entry for kw is removed from that bucket using $\mathsf{ReadAndRemove}$ bucket operations and stored in the ORAM program. Finally, the program assigns a new randomly chosen leaf ℓ' to the keyword kw, and moves the retrieved entry from its internal memory to the root node using the Add bucket operation. Any subsequent query for the same keyword kw is indistinguishable from queries to other keywords as the search paths are randomly distributed in the tree.

To write a new entry for the keyword kw, the Write operation first retrieves the entry stored for that keyword and drops it. This enforces that at most one entry per keyword is present in the database. The new entry is added to the root node.

If we keep on adding entries to the root bucket, it will eventually overflow. To prevent that, a background *eviction* process continuously moves entries from the root towards their designated leaves. After every query and on every tree level (starting from the root towards the leaves), a constant number ν of buckets is randomly chosen and evicted (a bucket can be chosen multiple times during one eviction phase). One entry $(kw, payload)$ in a chosen bucket $\mathcal{N}$ is removed and written to the next bucket on the path towards the leaf ℓ assigned to kw; the other child bucket of $\mathcal{N}$ is re-randomized; if $\mathcal{N}$ contains only dummy elements, both of its child buckets are re-randomized. To make this operation oblivious, the order is fixed and the left child is always processed before the right child. Figure 3 provides an example of the eviction process for $\nu = 1$. Notice that it is not necessary for the security of the scheme to perform an eviction after every ORAM operation and, in fact, eviction is only necessary to prevent bucket overflows and to guarantee the performance of the scheme, although the eviction process must, of course, be performed in an oblivious way.

We now state the security property for ORAM schemes.

Definition 1 (ORAM security [43]): A data request sequence $\vec{x}$ is an operation-argument tuple sequence $\vec{x} = ((\mathsf{op}_1, \mathsf{arg}_1), \ldots, (\mathsf{op}_\ell, \mathsf{arg}_\ell))$ where $\mathsf{arg}_i = kw$ if $\mathsf{op}_i = \mathsf{Read}$ and $\mathsf{arg}_j = (kw, ad)$ if $\mathsf{op}_j = \mathsf{Write}$ for keyword kw and data ad. We let $\mathsf{ops}(\vec{x}) := (\mathsf{op}_1, \ldots, \mathsf{op}_\ell)$ and $\mathcal{A}(\vec{x})$ denote the access pattern resulting for the execution of the data request sequence $\vec{x}$.

An ORAM construction is secure if and only if for every two arbitrary data request sequences $\vec{x}$ and $\vec{y}$ such that $\mathsf{ops}(\vec{x}) = \mathsf{ops}(\vec{y})$, the access patterns $\mathcal{A}(\vec{x})$ and $\mathcal{A}(\vec{y})$ are computationally indistinguishable.

The ORAM scheme by Shi et al. is secure and, thanks to its background eviction process, it achieves the best worst-case time complexity among the existing ORAM schemes.

Theorem 1 (ORAM properties [43]): The data structure by Shi et al. is a secure oblivious RAM with $\mathcal{O}(\log^2 N)$ worst-case and average-case time complexity and program storage of $\mathcal{O}(K \log N)$ size, where N is the number of entries in the database and K is the number of keywords.

B. Our Construction

In OBA, there can be multiple ads for every kw. It is therefore natural to have multiple entries for a keyword kw in our database, and we need to obtain all of them while searching (the Read operation) for a kw. Note that the ability to store and retrieve multiple elements per keyword is not definitional to ORAM. In fact, ORAM designs based on the so-called square-root solution [23], [24] cannot retrieve more than one entry per query from their ORAM architecture. In contrast, the ORAM scheme by Shi et al. can retrieve all possible entries associated with a kw, which, together with its best worst-case complexity, is the reason we chose it as a building block for ObliviAd. We now explain how the ORAM data structure described above can handle multiple entries per keyword without hampering the access privacy.

In the ORAM construction, it is possible to retrieve (Read) all the entries for a specific keyword while the SC goes through the complete path (from the root to the leaf) assigned to the keyword. This is achieved by modifying the ReadAndRemove operation to store in the ORAM memory and removing from the traversed bucket these entries. The subsequent Write operation stores the retrieved entries back into the root bucket. We stress that these modifications do not affect the cryptographic operations performed in the query processing, but only the amount of retrieved data.

Like the original ReadAndRemove (resp. Add) operation, the adapted ReadAndRemove (resp. Add) operation also modifies the complete bucket; thus, this operation remains secure as the IND-CPA property of the encryption scheme prevents the attacker from learning the number of entries which have been replaced by (resp. have replaced) dummy entries. As a result, there is no difference in the adversarial view of ORAM from a privacy perspective between the original ORAM construction and our variant.

One further important difference is that the ORAM client is replaced by a SC in our setting, as we use ORAM to efficiently perform PIR. As a result, during an ORAM Read operation, we cache the retrieved entries inside the SC internal memory before sending the response to the client and putting (via the Add operation) the retrieved entries into the root node bucket.

Let K be the number of keywords in the system. Let N be the number of keyword-advertisement entries (kw, id_{ad}, ad) to be stored in the database, where every entry is of size B bits. Note that N is generally greater than the number of ads as there can be multiple keywords kws attached to an ad. Therefore, the database size D is equal to $N \cdot B$, which we expect to be in the order of several GB or even a couple of TB for some brokers. Further, we expect the server storage n to be larger than D due to the overhead imposed by the ORAM construction. We also expect the SC to have an internal storage of size $m = \Omega(K \log N)$, which we use to store a mapping between keywords and leaves.

The tree contains N nodes and we let the ORAM buckets be of size $\mathcal{O}(\log N)$, implicitly bounding the maximum number of entries per keyword to $\mathcal{O}(\log N)$ (otherwise, it is impossible for a leaf to contain all the entries assigned to it). It can be shown that, in our construction, the Read and Evict operations take $\mathcal{O}(\log^2 N)$ time, while the write operation takes $\mathcal{O}(\log N)$ time. Intuitively, Read and Evict operate on each level of the tree (from the root to leaves) a constant number of times, while Write operates on the root only (and, therefore, are faster by a $\log N$ factor than in the original scheme).

Theorem 2 (Properties of our ORAM scheme): The data structure by Shi et al. with the modifications detailed above is a secure oblivious RAM with $\mathcal{O}(\log^2 N)$ worst-case and average-case time complexity and $\mathcal{O}(K \log(N))$ SC storage requirement.

Proof (Sketch): The ORAM property follows from the ORAM property of the original scheme [43]. The required decryption and encryption operations are performed also in the original version and our modifications are not distinguishable for the attacker thanks to the IND-CPA property of the encryption scheme. ■

V. Performance Analysis

In this section, we describe the implementation and evaluate the practicality of our ORAM construction,

which dominates the computational cost of our solution. We also suggest some optimizations based on our analysis and discuss other important system factors of our solution.

A. Implementation

We have developed a prototype to demonstrate the feasibility of our construction. Our implementation is a single-threaded application, comprising approximately 1200 lines of Java code, which performs the operations that in a concrete implementation of our system would be performed by the *SC*.

In our implementation, we assume that the ranking and selection of ads is conducted inside the *SC* [39]; note that it would, in principle, be possible to shift such auctions to the client side, if required [41]. We implemented buckets as arrays. The array size is determined at compile-time. Every array slot holds an advertisement and the corresponding keyword. Typically, buckets are only a few MB in size and fit easily into the *SC* memory. Therefore, we encrypt at the bucket level and not at the slot level. Consequently, if the advertisements differ greatly in size, we must apply a padding before encrypting a bucket.

B. Experiments

All experiments were conducted on a commodity PC with an Intel i5 quad-core processor with 3.3 GHz and 8 GB RAM. The hard drive has a speed of 7200 RPM and a cache of 16 MB. We implement the authenticated encryption scheme with AES encryption and HMAC. The cryptographic implementation was provided by the standard SunJCE Provider. To get consistent and comparable results, we set the advertisement size to 20KB and fix the tree depth to 24,[6] and report the average of 100 repetitions of the following experiments:

- We measure the impact of the bucket size (in terms of array slots) on the overall performance of our system. Figure 4*a*) displays the time required to read an advertisement from the database and the time required for the eviction process for a bucket size varying between 10 and 50. The tree depth remains unchanged at 25.
  For a reasonable bucket size of 30, we are able to read advertisements for a keyword in 750 ms. Even for a large bucket of size 50, we only require 750 ms where more than 85% of the time was spent on the cryptographic operations.
  The eviction process takes longer (in between 1 and 4.1 seconds, depending on the bucket size) but is performed after the reply is sent and, therefore, not experienced by the user.
- We measure how the number of ads retrieved in a single query influences the system performance. We fix the bucket size to 50 (the tree depth remains unmodified) and we store various amounts of advertisements for a single keyword inside the ORAM. Figure 4*b*) depicts the time required to read all the advertisements for the given keyword and the time spent by the eviction process. The results show that the number of ads does not affect the retrieval time.
- We show the scalability of our approach and fix the bucket size to 30 and let the depth of the tree vary from 10 to 30; i.e., we let the number of advertisements stored in the tree vary from one million to over one billion. Figure 4*c*) depicts the obtained timings; the experienced delay increases linearly from 280 ms for one million advertisements to 780 ms for one billion advertisements.

[6] A tree with depth 24 stores more than 16 million advertisements. Given an average advertisement size of 20 KB, the database stores over 300 GB of advertisement data.

C. Discussion

Impact on the User-experienced Delay. The experiments show that the Read operation requires up to 750 ms and the Evict operation requires up to 4.2 seconds. As the eviction is not necessary for achieving security, a client does not have to wait for the eviction process to finish. We can deliver the retrieved ad as soon as the Read process has terminated, increasing the overall delay of our system only by the amount of time required by a Read operation.

Our experimental results show that on a commodity PC with the cryptographic operations performed in software, our implementation requires, depending on the bucket size, between 150 ms and 728 ms. More than 85% of that time is spent in the various cryptographic routines. A dedicated hardware implementation as available in an *SC* will further decrease the time required to retrieve an advertisement and increase the performance of our system.

We use a secure and authenticated link between user devices and the *SC* to privately download the ads. Comprehensive studies [13] show that this delay is negligible compared to the ORAM-induced delay and, as *SC* CPU speeds increases, this delay will drop even further. The generation of an electronic token takes only 1.4 ms using RSA signatures and also constitutes a negligible overhead in comparison with the ORAM computations. Since the final tallying among the broker, publishers, and advertisers remains virtually identical to the existing system, the overall experienced user delay is dominated by the delay induced by the ORAM scheme.

Other System Delays. We determined that individual token verification operations takes only 0.08 ms for

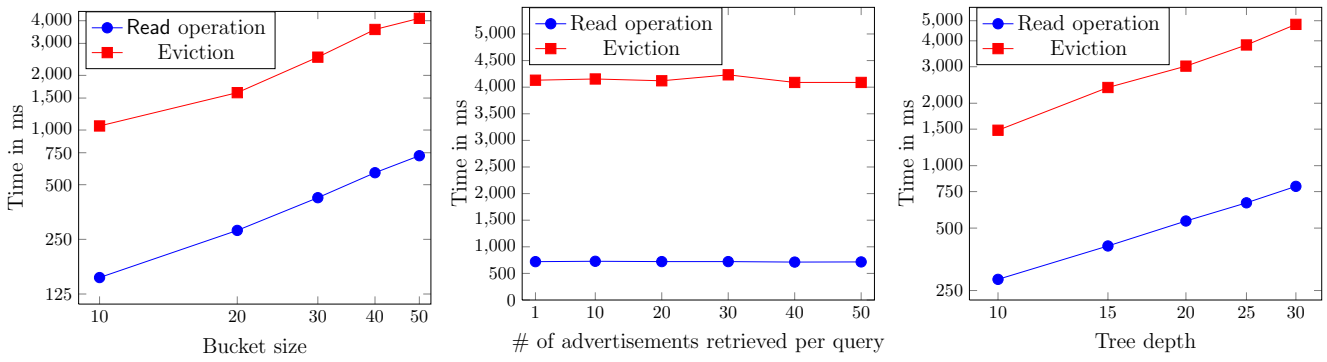

Figure 4. Results of our microbenchmark. For experiments *a*) and *b*), the tree depth is fixed to 24. For experiment *c*), the bucket size is fixed to 30.

RSA signatures. In addition, batched verification techniques [4] can further improve the overall verification performance during the tallying phase. Thus, our billing system can quickly process large amounts of electronic tokens.

Replication and Concurrency. The bottleneck of our construction is the *SC* fetching an advertisement from the database. Replications of the database and resulting concurrent computations can significantly improve the performance. It is possible for the broker to employ multiple *SC* units so that each of them maintains its own replicated copy of the database and caters to a different set of users in a completely parallel fashion. This replication does not affect the profile privacy of the users as their network addresses are known to the broker anyway. Realizing such a replication is harder in anonymity-based solutions to privacy preserving OBA (e.g., Privad).

It is also possible to let multiple *SC* units operate on the same ORAM storage since all ORAM operations consist of bucket-level operations, which are independent from each other. We just have to let all *SC* units share the same key material and enforce that at most one *SC* unit operates on a single bucket. Consequently, the only concurrency issue that needs to be taken care of is that all collaborating *SC* chips maintain the same copy of the keyword-leaf assignment. In that respect, the only blocking operation is the Add operation at the root node (as it changes the keyword-leaf assignment). At the cost of using memory in the *SC* chip, it is possible to postpone these Add operations, thus improving the performance. Finally, we mention that the eviction procedure is also highly parallelizable since it operates on distinct buckets.

VI. Formal Verification

We conduct a formal security analysis of our system. Although the deployed cryptographic primitives are secure by themselves, we must ensure the absence of flaws in the protocol design, e.g., unintended, attacker-driven interleavings of concurrently executed protocol sessions that break the security properties. To exclude such flaws and to establish a security proof, we model our protocol in the applied-pi calculus [1], a process calculus for the specification and analysis of cryptographic protocols. We formalize privacy properties as observational equivalence relations between processes and correctness properties as trace properties. The verification is automatically conducted using ProVerif [6], a state-of-the-art automated theorem prover based on Horn-clause resolution, which provides security proofs for an unbounded number of concurrent protocol sessions. The ProVerif scripts used in the analysis are publicly available [3].

A. Profile Privacy

We verify that an attacker cannot obtain any information about client profiles, even when in full control of the advertiser, the publisher, and the broker. We model this property as an indistinguishability game (technically, an observational equivalence relation $\approx$ [1]) between two processes, as depicted in Figure 5. Here and throughout the rest of this paper, we let E_{ad} denote the symmetric encryption $\mathsf{enc}(id_{ad}, kw_{ad})_k$ of the ad identifier id_{ad} along with the ad keyword kw_{ad}. In the first process P (left-hand side), A and B's profiles consist of the keywords kw_A and kw_B, respectively. In the second process Q (right-hand side), the two profiles are swapped. If the processes P and Q are observationally equivalent, written $\mathsf{P} \approx \mathsf{Q}$, then the attacker cannot learn which profile belongs to which client. We assume a very pessimistic setting where the attacker has the control over arbitrarily many clients and knows the two profiles kw_A and kw_B. More precisely, our game works as follows:

0. The attacker chooses two advertisements ad_A and ad_B and stores them in the *SC* (this corresponds to the broker filling her ORAM database via the *SC*).

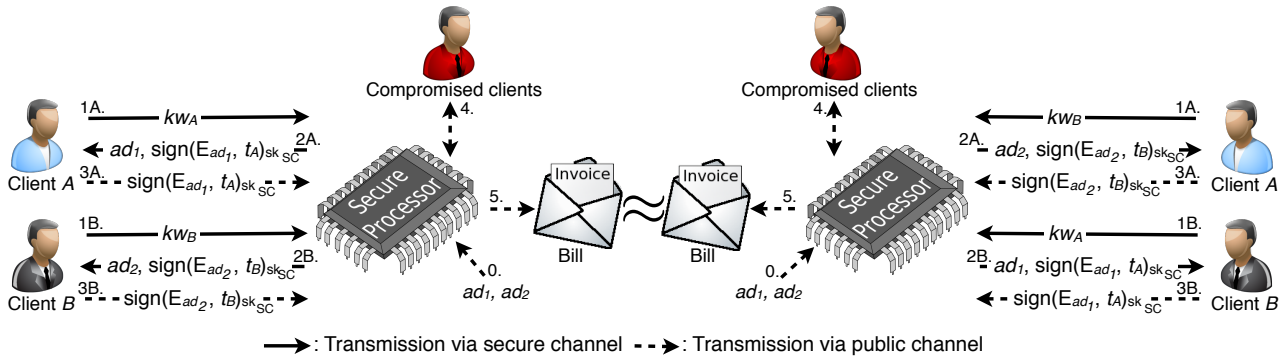

Figure 5. Overview of the observational equivalence relation. The left side of the picture corresponds to P and the right side to Q.

These two advertisements match the two profiles kw_A and kw_B, respectively.

1A./1B. Client A and Client B send their profile to the SC via a secure channel.

2A./2B. The SC sends back the response, comprising the advertisement that best matches the received profile along with the accompanying token.

3A./3B. The two clients publish their token on a public channel.

4. Corrupted clients, i.e., clients acting exclusively on behalf of the attacker, can arbitrarily interact with the SC.

5. After collecting the two honest clients' tokens, and possibly other tokens from compromised clients, the SC initiates the accounting process. The SC verifies the signatures in the tokens, verifies the timestamps, decrypts the ad identifiers, and publishes a permutation thereof, which constitutes the bill.

The attacker has full control over scheduling decisions, e.g., the actions of client A, client B, and compromised clients can be interleaved in any order, with the natural constraint that client A and client B follow the protocol, i.e., their respective actions are executed in the right order.

Theorem 3 (Profile Privacy): The observational equivalence relation $\mathsf{P} \approx \mathsf{Q}$ holds true.

Proof: Automatically proven using ProVerif. ∎

B. Profile Unlinkability

When verifying the privacy of the client profiles, we assume the worst case scenario, i.e., the attacker knows the client profiles. We now analyze a different property, namely, profile unlinkability. In this scenario, the attacker does not know the client profiles. We verify that it is impossible for an attacker to deduce any information about client profiles by observing the tokens sent by that client and the final tally, even when in full control of the advertiser, the publisher, and the broker. We model this property as an indistinguishability game between two processes P (left-hand side) and Q (right-hand side), as depicted in Figure 6. In both processes, A and B's profiles consist of the keyword kw_A and kw_B, respectively. The game obeys the following steps:

0. The attacker chooses four advertisements ad_1, ad_2, ad_3, and ad_4, and stores them in the SC (this corresponds to the broker filling her ORAM database via the SC). In process P, advertisements ad_1 and ad_3 are the best-matching advertisements for kw_A and kw_B, and ad_2 and ad_3 are the second-best-matching advertisements for kw_A and kw_B, respectively.[7] Notice that we assume the worst-case scenario, i.e., the two client profiles kw_A and kw_B are disjoint and, thus, easier to be distinguished. In process Q, ad_2 and ad_3 are swapped, i.e., client profiles are different in P and Q.

1A./1B to 3A./3B. Client A and B both send their keyword to the SC via a secured channel and receive back the best-matching advertisements (i.e., ad_1 and ad_3 in P and ad_1 and ad_2 in Q), and the corresponding tokens. Following the protocol, the tokens are immediately sent to the SC.

4A./4B to 6A./6B. Both clients perform the same steps as above. The returned advertisements, however, are the second-best-matching ones (i.e., ad_2 and ad_4 in P and ad_3 and ad_4 in Q).

7. Compromised clients can arbitrarily interact with the SC.

8. After collecting the four honest clients' tokens, and possibly other tokens from compromised clients, the SC initiates the accounting process. The SC verifies the signatures on the tokens, verifies the time stamps, decrypts the ad identifiers, and publishes a permutation thereof, which constitutes the bill.

As in the game for profile privacy, the attacker has full control over scheduling decisions and the above described game steps can be interleaved in any order,

[7] We recall that the SC ranks the advertisements according to each user profile.

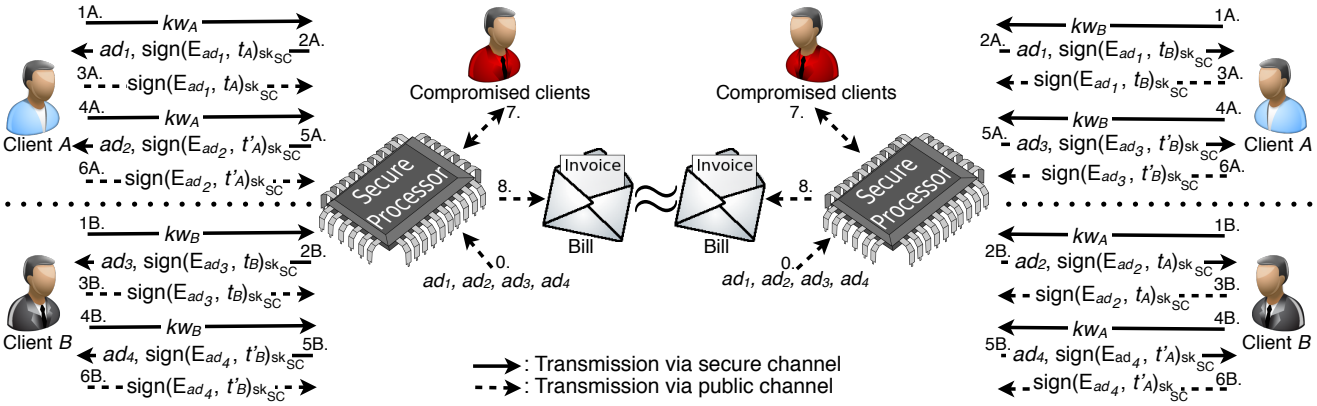

Figure 6. Overview of the observational equivalence relation. The left side of the picture corresponds to P and the right side to Q.

as long as the actions of client A and client B follow the protocol. If the processes P and Q are observationally equivalent, then the profiles are unlinkable, i.e., the adversary cannot determine which entries in the final tally were caused by which profile. For instance, if the final tally contains two entries for cars and two for sports, it is impossible to say if each client is interested only in one of the two topics, or if the two clients are both interested in sports and in cars.

Theorem 4 (Profile Unlinkability): The observational equivalence relation $\mathsf{P} \approx \mathsf{Q}$ holds true.

Proof: Automatically proven using ProVerif. ∎

C. Billing Correctness

A fundamental goal of our system is the correctness of the billing process. Brokers expect to be reimbursed for their services and the advertiser is only willing to pay for advertisements that have been seen or clicked on. A first property we expect is *non-reusability of tokens*, i.e., each tokens is counted at most once. A second property is *billing fairness*, i.e., whenever a token is counted, then the corresponding ad was really watched by the user. We formalize and verify these two properties in a strong adversarial model, in which the attacker has the control over arbitrarily many corrupted clients. Therefore, we distinguish whether a token $\mathsf{sig}(\mathsf{E}_{ad}, t)_{sk_{SC}}$ is issued for a honest client, annotated with the predicate $\mathsf{IssueHonToken}(id_{ad}, t)$, or for a compromised client, marked with the predicate $\mathsf{IssueCompToken}(id_{ad}, t)$. Additionally, we decorate the point in the protocol where a honest client watches the ad and sends her token with $\mathsf{SendToken}(id_{ad}, t)$ and the point where the SC counts a token with $\mathsf{CountToken}(id_{ad}, t)$. Notice that we cannot decorate compromised clients, since they run arbitrary code under the control of the attacker. Figure 7 depicts the process P annotated with these predicates. We want to verify that in all protocol executions each CountToken predicate is preceded either by a distinct SendToken predicate, which is in turn preceded by a distinct IssueHonToken predicate (token for honest clients), or by a distinct IssueCompToken predicate (tokens for compromised clients). In the literature, this kind of properties are known as injective agreement [34]. The billing correctness property can be formalized in ProVerif notation as follows:

$$\begin{aligned}&\mathsf{CountToken}(id_{ad}, t) ==>_1 \\ &\quad (\mathsf{SendToken}(id_{ad}, t) ==>_1 \mathsf{IssueHonToken}(id_{ad}, t)) \\ &\qquad \vee\ \mathsf{IssueCompToken}(id_{ad}, t)\end{aligned} \tag{1}$$

where $P_1 ==>_1 P_2$ denotes the requirement that each predicate P_1 must be preceded by a distinct predicate P_2 in all protocol executions. The above property says that tokens are never counted more than once. For honest clients, we also know that whenever a token is counted, then the corresponding ad has been watched. Compromised clients cannot be decorated with events, reflecting the fact that a compromised client might forward the token to the broker behind the scenes, i.e., without the user really watching the ad.

Theorem 5: The trace property stated in equation (1) holds true in all possible execution traces of the process P.

Proof: Automatically proven using ProVerif. ∎

VII. Related Work

Given the significance of the privacy-preserving OBA problem to the masses, the privacy enhancing technologies (PETs) research community is showing a growing interest in this problem [18], [20], [25], [26], [28], [29], [49]. Instead of blocking OBA, as it is done by tools such as [15], [40], these PETs are meant to be practical privacy preserving alternatives that the advertising industry should also find attractive. The solutions proposed so far, however, fall short of providing an adequate degree of privacy and a satisfactory performance at the same time.

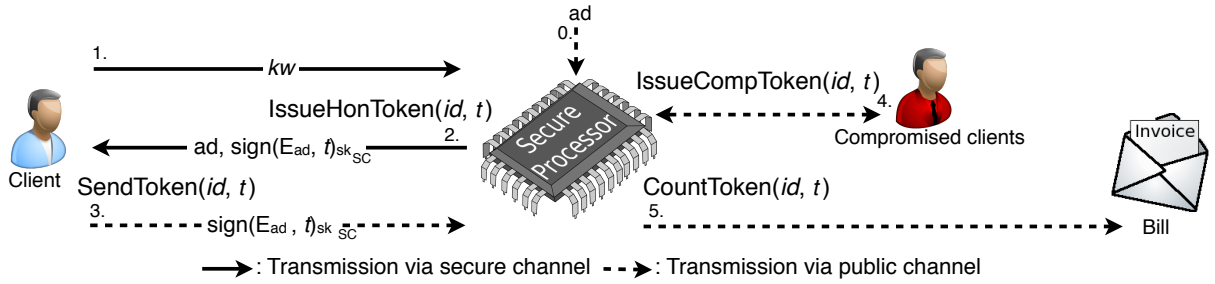

Figure 7. The process P, annotated with logical predicates, used in the verification of the trace property in equation (1).

Juels [28] was the first to explore the notion of targeted yet privacy preserving advertising and the first to suggest the usage of client-side proxies to manage user profiles, a concept used by almost all privacy preserving advertising systems today. Like us, Juels utilizes a PIR scheme for privacy preserving distribution of ads. That PIR scheme, however, is impractical for a real time use and, in particular, makes it impossible to retrieve ads on-the-fly. Furthermore, that work does not secure the tallying phase, in which the broker computes the number of ads viewed by clients for billing purposes. Finally, provable anonymity is achieved against a threshold adversary, which is not a good model of real life brokers. In contrast, we use a highly efficient PIR solution based on a secure coprocessor, which enables us to retrieve ads on-the-fly. We also secure the tallying phase using electronic tokens that are mixed by the *SC* in order to preserve the privacy of the client profiles.

Privad [25], [26] presents a complete system for privacy preserving targeted advertising. Along with the usual client software and broker entities, Privad introduces a *reference monitor* that watches the client software to ensure that no information is sent by the client to the broker through a covert channel, and a *dealer* that works as an anonymizing proxy between the user and the broker. Similar to our system, the Privad client builds a user profile and requests relevant ads from the broker by sending the profile information to the broker system, with the fundamental difference that every communication in Privad is proxied by the dealer. Further, in Privad, any communication between the client and the broker is encrypted with the broker's public key or an agreed session key so that the dealer cannot see the profile information or the downloaded, viewed or clicked ads.

Privad achieves profile privacy through the anonymity of the client. In order to facilitate client-side fraud detection, Privad does not employ a reliable anonymity solution like Tor [48]; it instead asks for a privacy preserving proxy (the dealer) that is assumed to be *honest-but-curious* and, specifically, to not collude with the broker. For detecting client-side fraud, however, the dealer needs to satisfy the advertiser's and the broker's demands. Realizing such a party looks to be a difficult problem to solve from the administrative, legal, and technological perspective. Even if realized, the dealer is highly susceptible to traffic analysis attacks and may also introduce a single-point-of-failure in the advertising system. In our solution, we avoid such an entity and completely eliminate the corresponding trust requirement by introducing a broker-side trusted hardware. As a result, we achieve profile privacy without enforcing user anonymity.[8]

Adnostic [49] provides a completely different approach to privacy preserving OBA. Here, a privacy-protecting client software obtains a small set of ads, randomly chosen by the broker, when the user visits a webpage containing slots for ads. The software then selects the most appropriate ad based on the user profile and shows it to the user. For these viewed ads, Adnostic performs homomorphic encryptions and creates zero-knowledge proofs at the user devices to allow the broker to reliably settle the accounts with the publishers and the advertisers without knowing who viewed which ad. A fundamental assumption in Adnostic is that the advertisers and the broker will always collaborate; thus, ad-clicks are treated the same as in current ad networks, where the client reports clicks directly to the broker.

Unlike ObliviAd and Privad, Adnostic does not hide users' web browsing or clicking behavior from the broker, which makes their privacy goals considerably weaker. Further, as ads are sent without any behavioral targeting and only in small sets, it seems that this approach would be more comparable to the old scattershot approach than today's modern OBA methods. Further, although their accounting solution is novel and interesting cryptographically, it does not provide any information about the viewed ads. This information is needed for conducing future ad-auctions and click-through analysis; thus, the quality of OBA will not improve, even over a longer period. Finally, our solution is significantly more efficient on both the client and the broker sides: Adnostic uses rel-

[8]If the user wants to anonymously access web services, anonymity networks like Tor are necessary anyway.

atively expensive public-key homomorphic encryptions and zero knowledge proofs, while we instead relay on inexpensive symmetric-key encryptions.

An interesting work in the field of OBA is RePriv [20], an architecture that provides an ideal solution for realizing the client-side software required by virtually all privacy preserving OBA systems, including ours. Reznichenko, Guha and Francis [41] have recently proposed a solution for running advertising auctions that leverage user profiles for ad ranking without compromising their privacy. We observe that their auction design can easily be incorporated in our system.

VIII. Conclusion and Future Work

Online behavioral advertising is receiving increasing attention by the financial and research community. The essence of the current business model is to target advertisements according to user profiles. This poses a significant threat to the privacy of users. In this paper, we show that it is possible to achieve strong privacy properties while retaining the current business and system model. Our solution utilizes PIR technology, which we implement using a secure coprocessor and an efficient ORAM construction, in order to allow the clients to retrieve advertisements that best match their behavioral profile without the broker learning any personal information. At the same time, our architecture allows brokers to learn any non-personal information that they may find useful for improving their business model (e.g., click-through rate, statistics for click-fraud detection, etc.).

We formalize two fundamental privacy goals (i.e., profile privacy and profile unlinkability) as observational equivalence relations and the billing correctness requirement as a trace property. These are the first formal security definitions for OBA. We used ProVerif, an automated cryptographic protocol verifier, to formally establish them on ObliviAd. An experimental evaluation demonstrates the feasibility of our approach.

We are developing a complete implementation of our system, comprising a browser plugin and open-source software for brokers. We also intend to devise an ORAM construction that is tailored to our low-latency system, for instance, by natively supporting multiple entries per keyword and by increasing the fanout of the tree. Finally, we plan to identify and formalize further security properties for OBA systems (e.g., click-fraud detection) and develop tools for their verification.

A trusted platform module (TPM) is a downscaled and inexpensive version of the secure coprocessors that provide only a restricted set of operations such as trusted boot, remote attestation, and sealed storage. Recent efforts [8], [36], [42], [45] suggest that more involved secure operations can also be performed from these commodity chips. It would be interesting to explore the usage of TPM architectures for implementing our ORAM construction, in order to provide an inexpensive alternative to cryptographic coprocessors.

Acknowledgments. This work was partially supported by the initiative for excellence and the Emmy Noether program of the German federal government, and the Center for IT-Security, Privacy and Accountability (CISPA).

References

[1] M. Abadi and C. Fournet, "Mobile values, new names, and secure communication," in *Proc. Symposium on Principles of Programming Languages (POPL'01)*. ACM Press, 2001, pp. 104–115.
[2] http://www.anonymizer.com.
[3] M. Backes, A. Kate, M. Maffei, and K. Pecina, "ObliviAd ProVerif Scripts," http://lbs.cs.uni-saarland.de/obliviad.
[4] M. Bellare, J. A. Garay, and T. Rabin, "Fast Batch Verification for Modular Exponentiation and Digital Signatures," in *Advances in Cryptology (EUROCRYPT'98)*, 1998, pp. 236–250.
[5] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," *Journal of Cryptology*, vol. 21, no. 4, pp. 469–491, 2008.
[6] B. Blanchet, "An Efficient Cryptographic Protocol Verifier Based on Prolog Rules." in *Proc. IEEE Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society Press, 2001, pp. 82–96.
[7] A. W. Branscomb, "Anonymity, Autonomy, and Accountability: Challenges to the First Amendment in Cyberspaces," *The Yale Law Journal*, vol. 104, no. 7, 1995.
[8] A. Brown and J. Chase, "Trusted Platform-as-a-Service: A Foundation for Trustworthy Cloud-Hosted Applications," in *The ACM Cloud Computing Security Workshop (CCSW'11)*, 2011.
[9] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Journal of the ACM*, vol. 24, no. 2, pp. 84–88, 1981.
[10] ——, "Blind Signatures for Untraceable Payments," in *Proc. Advances in Cryptology (CRYPTO'87)*, 1982, pp. 199–203.
[11] D. Chaum, A. Fiat, and M. Naor, "Untraceable Electronic Cash," in *Proc. Advances in Cryptology (CRYPTO'88)*, 1990, pp. 319–327.
[12] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," in *Proc. IEEE Symposium on Foundations of Computer Science (FOCS'95)*, 1995, pp. 41–50.
[13] C. Coarfa, P. Druschel, and D. S. Wallach, "Performance analysis of TLS Web servers," *ACM Trans. Comput. Syst.*, vol. 24, no. 1, pp. 39–69, 2006.
[14] F. T. Commission, "FTC Staff Report: Self-Regulatory Principles For Online Behavioral Advertising," http://www.ftc.gov/opa/2009/02/behavad.shtm, Feb 2009, accessed October 2011.
[15] http://www.aboutads.info/choices/.
[16] S. Delaune, S. Kremer, and M. Ryan, "Verifying privacy-type properties of electronic voting protocols," *JCS*, vol. 17, pp. 435–487, 2009.

[17] X. Ding, Y. Yang, R. H. Deng, and S. Wang, "A new hardware-assisted PIR with O(n) shuffle cost," *Inter. Journal of Information Security*, vol. 9, no. 4, pp. 237–252, 2010.

[18] D. S. Evans, "The Online Advertising Industry: Economics, Evolution, and Privacy," *Journal of Economic Perspectives*, vol. 23, no. 3, pp. 37–60, 2009.

[19] J. Feng, H. K. Bhargava, and D. M. Pennock, "Implementing Sponsored Search in Web Search Engines: Computational Evaluation of Alternative Mechanisms," *INFORMS J. on Computing*, vol. 19, pp. 137–148, January 2007.

[20] M. Fredrikson and B. Livshits, "RePriv: Re-imagining Content Personalization and In-browser Privacy," in *Proc. IEEE Symposium on Security & Privacy (S&P'11)*, 2011, pp. 131–146.

[21] O. Goldreich, "Towards a theory of software protection and simulation by oblivious RAMs," in *Proc. ACM Symposium on Theory of Computing (STOC'87)*, 1987, pp. 182–194.

[22] ——, *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[23] O. Goldreich and R. Ostrovsky, "Software Protection and Simulation on Oblivious RAMs," *Journal of the ACM*, vol. 43, pp. 431–473, May 1996.

[24] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia, "Oblivious RAM Simulation with Efficient Worst-Case Access Overhead," in *Proc. ACM Workshop on Cloud Computing Security (CCSW'11)*, 2011, pp. 95–100.

[25] S. Guha, B. Cheng, and P. Francis, "Privad: Practical Privacy in Online Advertising," in *Proc. Symposium on Networked Systems Design and Implementation (NSDI'11)*, Mar 2011.

[26] S. Guha, A. Reznichenko, K. Tang, H. Haddadi, and P. Francis, "Serving Ads from localhost for Performance, Privacy, and Profit," in *Proc. Workshop on Hot Topics in Networks (HotNets)*, 2009.

[27] http://www-03.ibm.com/security/cryptocards/.

[28] A. Juels, "Targeted Advertising ... And Privacy Too," in *CT-RSA'01*, 2001, pp. 408–424.

[29] P. Kazienko and M. Adamski, "AdROSA - Adaptive personalization of web advertising," *Inf. Sci.*, vol. 177, no. 11, pp. 2269–2295, 2007.

[30] N. J. King, "Why privacy discussions about pervasive online customer profiling should focus on the expanding roles of third-parties," *International Journal of Private Law*, vol. 4, no. 2, pp. 193–229, 2011.

[31] S. Komanduri, R. Shay, B. U. Greg Norcie, and L. F. Cranor, "AdChoices? Compliance with Online Behavioral Advertising Notice and Choice Requirements," Carnegie Mellon University, Tech. Rep. CMU-CyLab-11-005, October 2011.

[32] B. Krishnamurthy and C. E. Wills, "Cat and mouse: content delivery tradeoffs in web access," in *Proc. International World WideWeb Conference (WWW'06)*, 2006, pp. 337–346.

[33] ——, "Privacy diffusion on the web: a longitudinal perspective," in *Proc. International World WideWeb Conference (WWW'09)*, 2009, pp. 541–550.

[34] G. Lowe, "A Hierarchy of Authentication Specifications," in *Proc. IEEE Computer Security Foundations Workshop (CSFW'97)*. IEEE Computer Society Press, 1997, pp. 31–44.

[35] J. Mayer and A. Narayanan, "Do Not Track: Universal Web Tracking Opt-Out," http://donottrack.us, Stanford University, accessed October 2011.

[36] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and H. Isozaki, "Flicker: an execution infrastructure for tcb minimization," in *Proc. European Professional Society on Computer Systems (EUROSYS'08)*. ACM Press, 2008, pp. 315–328.

[37] J. McLachlan, A. Tran, N. Hopper, and Y. Kim, "Scalable onion routing with torsk," in *Proc. ACM Conference on Computer and Communications Security (CCS'09)*, 2009, pp. 590–599.

[38] F. Olumofin and I. Goldberg, "Revisiting the Computational Practicality of Private Information Retrieval," in *Proc. Inter. Conference on Financial Cryptography and Data Security (FC'11)*, 2011.

[39] A. Perrig, S. W. Smith, D. X. Song, and J. D. Tygar, "SAM: A Flexible and Secure Auction Architecture Using Trusted Hardware," in *Proc. International Parallel & Distributed Processing Symposium (IPDPS'01)*, 2001, p. 170, http://sparrow.ece.cmu.edu/˜adrian/projects/SAM/.

[40] http://www.privacychoice.org/privacymark.

[41] A. Reznichenko, S. Guha, and P. Francis, "Auctions in Do-Not-Track Compliant Internet Advertising," in *Proc. ACM Conference on Computer and Communications Security (CCS'11)*, 2011, pp. 667–676.

[42] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *Proc. Conference on Hot topics in Cloud Computing (HotCloud'09)*, 2009.

[43] E. Shi, T.-H. H. Chan, E. Stefanov, and M. Li, "Oblivious RAM with $O((logN)^3)$ Worst-Case Cost," in *Advances in Cryptology (ASIACRYPT'11)*, 2011, pp. 197–214.

[44] R. Sion and B. Carbunar, "On the Practicality of Private Information Retrieval," in *Proc. ISOC Network and Distributed System Security Symposium (NDSS'07)*. Internet Society, 2007.

[45] E. G. Sirer, W. de Bruijn, P. Reynolds, A. Shieh, K. Walsh, D. Williams, and F. B. Schneider, "Logical attestation: an authorization architecture for trustworthy computing," in *Proc. Symposium on Operating Principles (SOSP'11)*, 2011, pp. 249–264, http://www.cs.cornell.edu/people/egs/nexus/.

[46] S. W. Smith, "Outbound Authentication for Programmable Secure Coprocessors," *Inter. Journal of Information Security*, vol. 3, no. 1, pp. 28–41, 2004.

[47] S. W. Smith and S. Weingart, "Building a high-performance, programmable secure coprocessor," *Computer Networks*, vol. 31, no. 8, pp. 831–860, 1999.

[48] "The Tor Project," https://www.torproject.org/, 2003, accessed October 2011.

[49] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy Preserving Targeted Advertising," in *Proc. Network and Distributed System Security Symposium (NDSS'10)*, 2010.

[50] S. Wang, X. Ding, R. H. Deng, and F. Bao, "Private Information Retrieval Using Trusted Hardware," in *Proc. European Symposium on Research in Computer Security (ESORICS'06)*, 2006, pp. 49–64.

[51] P. Williams and R. Sion, "Usable PIR," in *Proc. ISOC Network and Distributed System Security Symposium (NDSS'08)*, 2008.