AdVeil: A Private Targeted-Advertising Ecosystem

Sacha Servan-Schreiber
MIT CSAIL
3s@mit.edu

Kyle Hogan
MIT CSAIL
klhogan@mit.edu

Srinivas Devadas
MIT CSAIL
devadas@mit.edu

ABSTRACT

This paper presents AdVeil, a privacy-preserving advertising ecosystem with formal guarantees for end users.

AdVeil is built around an untrusted advertising network which is responsible for brokering the display of advertisement to users. This ad network targets relevant ads to users *without* learning any of the users' personal information in the process. Our targeting protocol combines private information retrieval with standard, locality-sensitive hashing based techniques for nearest neighbor search. By running ad targeting in this way, users of AdVeil have full control over and transparency into the contents of their targeting profile.

AdVeil additionally supports private metrics for ad interactions, allowing the ad network to correctly charge advertisers and pay websites for publishing ads. This is done *without* the ad network learning which user interacted with an ad, only that some honest user did. AdVeil achieves this using an anonymizing proxy (e.g., Tor) to transit batched user reports along with unlinkable anonymous tokens to certify the authenticity of each report.

We build a prototype implementation of AdVeil which we evaluate on a range of parameters to demonstrate the applicability of AdVeil to a real-world deployment. Our evaluation shows that AdVeil scales to ad networks with millions of ads, using state-of-the-art single-server private information retrieval. A selection of ads from a database of 1 million ads can be targeted to a user in approximately 4.5 seconds with a single 32-core server, and can be parallelized further with more servers. Targeting is performed out-of-band (e.g., on a weekly basis) while ad delivery happens in real time as users browse the web. Verifying report validity (for fraud prevention) requires less than 300 microseconds of server computation per report.

1 INTRODUCTION

Internet advertising is a $124 billion industry that relies on pervasive tracking of internet users for the purpose of serving them relevant advertisements. This process is known as *targeted* advertising and has been the focus of recent controversy due to the often highly personal nature of the data being used and the invasiveness of the collection practices [41, 73, 82]. Proposals for moving away from targeted advertising often tout *contextual* advertising as a privacy focused alternative [52, 62, 80]. In contextual advertising, ads are chosen based *only* on the website they will be displayed on, and *not* on personal data, eliminating the need for web tracking.

However, relevancy of advertisements for end users is believed to increase user engagement which, consequently, increases profit [56, 68]. Google and Facebook, the most prominent ad-tech companies today, respectively earned 83% and 99% of their 2019 revenue from advertising alone [37, 48, 90]. Because of this, ad networks have proven unwilling to move away from targeted advertising and the associated web tracking. Recent proposals from Google for a privacy preserving alternative [99, 100] fall short, with privacy advocates such as the Electronic Frontier Foundation (EFF) referring to them as being "the opposite of privacy-preserving technology" [29].

Other solutions for privacy-preserving targeted advertising have practical performance limitations or fail to achieve targeting accuracy comparable to non-private advertising [9, 14, 43, 45–47, 53, 75, 92, 93, 99, 100]. AdVeil addresses these problems.

AdVeil is a low-latency, scalable solution for targeted advertising with clear-cut privacy guarantees. The main goal of AdVeil is to provide *unlinkability* between users and their personal data. We define *personal data* to mean any information about the user, including which ads they interact with. The interests and demographics contained in users' targeting profiles are not sent in the clear—even anonymized—to the ad network as part of the targeting process. However, due to the nature of targeting, knowledge of the ads a user sees can be used to infer the, clearly personal, data that makes up the user's profile. As such, AdVeil ensures that the ad network learns **only** which ads are displayed to (and clicked by) users, but not which user saw any given ad.

AdVeil is fully compliant with data privacy legislation such as General Data Protection Regulation (GDPR) [2] and California Consumer Privacy Act (CCPA) [3]. Users' profiles are held locally, with the users themselves having full control over and transparency into *which* of their features are used for targeting. They may even opt out of targeted advertising entirely, in which case websites can display contextual ads that are related only to the page itself, not to the users viewing them.

Giving users control over their own data is important for reasons that stem beyond privacy. Companies have been found to use protected demographics like gender, race, or religion to target advertisements in a discriminatory manner [5, 22, 49, 101]. This practice can limit the visibility of needed services, such as home loans or continuing education, to people who might have benefited from them [33, 76].

Finally, addressing ad fraud is a crucial requirement for online advertising [85, 108]. Malicious parties may attempt to generate false ad interactions for the purposes of artificially running down an advertiser's budget or increasing revenue for a website for displaying ads. AdVeil provides an integrated fraud-prevention mechanism for detecting bots and ensuring accuracy of reports.

Components. AdVeil is realized using well established building blocks which we combine to provide a scalable, accurate, and privacy-preserving advertising ecosystem. We use private information retrieval in conjunction with standard nearest neighbor search data structures to achieve private ad targeting. Our fraud-prevention mechanism is built using unlinkable tokens with metadata [57, 81, 95]. We use any common anonymizing proxy (e.g., Tor [35]) to hide the identities of users from the ad network during ad delivery and reporting; (the proxy is not required during targeting).

To the best of our knowledge, we are the first to apply private-information-retrieval to privately *target* ads to users. We believe that this technique may be of independent interest for related problems such as privacy-preserving recommendation systems.

Contributions. AdVeil makes the following contributions:

(1) A novel and efficient protocol for ad targeting that reveals no information about the user profile.
(2) An integrated fraud prevention mechanism based on anonymous tokens. Our solution remains fully compatible with private browsing and existing ad-fraud counter-measures.
(3) Out-of-the-box compliance with recent privacy regulations such as GDPR and CCPA, in addition to full compatibility with existing anti-tracking and privacy-enhancing technologies.
(4) A prototype implementation which we empirically evaluate to demonstrate applicability to a real-world deployment. Our code is open-source and available at http://adveil.com/code.

Limitations. We highlight several drawbacks of AdVeil compared to the status-quo in (non-private) targeted online advertising.

(1) AdVeil imposes a computational overhead on the ad network, which translates to higher operational costs. However, in our evaluation (see Section 7) we provide concrete cost estimates and show that AdVeil remains profitable, even if deployed on off-the-shelf cloud infrastructure.
(2) AdVeil is intended to *disincentivize* web tracking by ad networks and is fully compatible with the use of any existing web tracking defenses. However, it is not itself a defense against browser fingerprinting or other tracking mechanisms.
(3) AdVeil requires cooperation of browsers. We note, however, that major browser vendors are already providing local support for several aspects of privacy-preserving advertising [75, 99, 102].

2 BACKGROUND

In this section we start by introducing the existing, non-private advertising ecosystem. This ecosystem consists of a set of parties who engage in different stages of an *advertising pipeline*.

This pipeline supports many different styles of targeted advertising, each with tradeoffs between their data requirements and the relevancy of ads delivered to users. We provide details on a selection of prevalent advertising styles in Section 2.3.

2.1 Participants

We use standard terminology for parties that comprise the advertising pipeline [9, 43, 45, 53, 92]:

- **Users**: people browsing the web and viewing ads;
- **Clients**: web browsers controlled by users (e.g., Firefox);
- **Advertisers**: companies with products and services to advertise to a targeted demographic (e.g., Squarespace, inc.);
- **Publishers**: websites that display ads to users viewing the webpage (e.g., `wired.com` or mobile applications);
- **Broker**: an ad-tech company (e.g., Google AdSense) responsible for matching users to ads, billing advertisers, and compensating publishers for user interactions.

The Broker, or ad network, can be thought of as the governing body of the advertising ecosystem. The Broker's primary goal is matching ads from an Advertiser to the users most likely to engage with them.

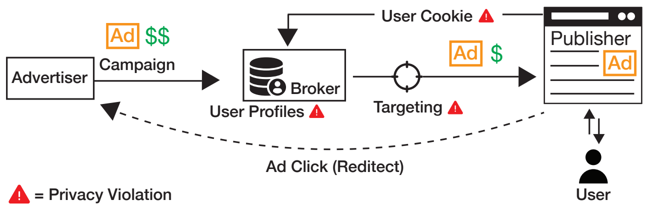


Figure 1: (non-private) targeted advertising pipeline

This is done via Publishers, companies that generate revenue by displaying ads to users visiting their webpage or app.

2.2 Advertising pipeline

The advertising pipeline, shown in Figure 1, begins when an Advertiser creates a new ad campaign with the Broker, specifying a target audience and allocating a campaign budget. The target audience is specified using a *feature vector* which contains information about the contents of the ad, e.g., tags such as `mechanical keyboards` or `outdoor gear`. In addition to the feature vectors provided by Advertisers, the Broker possesses *user profiles* containing demographics and interests, e.g., `[woman, computers, high-income]`, about users. These profiles are constructed by tracking users on the web [101]. The Broker is responsible for *targeting* ads to users based on this profile through a similarity search algorithm [78, 99].

When users are shown an ad on a publisher's webpage, a browser script generates an *impression* report for the Broker. This is by far the most common type of user engagement, followed by *clicks*, which are redirected through the Broker to measure engagement. With impression reports and click redirects, the Broker obtains necessary metrics for billing Advertisers, compensating Publishers, and updating targeting algorithms. Crucially, the Broker eliminates all fraudulent views and clicks that it deems to be generated by bots or malicious Publishers and clients to artificially skew perceived user interactions with ads [85].

2.3 Styles of online advertising

Online advertising consists of two main targeting strategies for matching ads to users: *contextual* and *behavioral* targeting.

Contextual advertising is done *independently of the user's profile*. Publishers display ads that are relevant to their own content, which is assumed to be relevant to visiting users based on their choice to access the Publisher's site or app.

Behavioral ad targeting matches a user to a set of ads based on the *user's profile*. Each profile consists of the user's browsing behavior or collection of apps installed on the user's device, obtained by the Broker through user tracking. Retargeting is a common type of behavioral advertising in which the Broker may preferentially display an Advertiser's ads to users who have previously interacted with that Advertiser, e.g., by visiting their website.

Remark 1. ***Real-Time Bidding*** *is a mechanism associated with both behavioral and contextual advertising. It occurs on-demand, at the time a user requests an ad, and auctions the available ad slot to the highest bidding Advertiser.*

Each style of online advertising has tradeoffs between the relevance of ads and the invasiveness of the data required. Serving increasingly relevant ads, as in behavioral or retaregted ads, currently requires corresponding increases in user tracking granularity.

In the next section we describe how AdVeil supports the targeting of relevant ads *without* sacrificing user privacy.

3 THE ADVEIL SYSTEM

AdVeil is a privacy-preserving instantiation of the advertising pipeline. AdVeil is designed to be modular and extensible with each stage of the advertising pipeline implemented independently.

This allows AdVeil to provide a general-purpose solution for private advertising that maintains unlinkability guarantees for users. The Broker retains comparable targeting accuracy, performance, metrics, and fraud prevention to existing advertising pipelines *without* the ability to learn which ads a user sees. That is, the user's Personally Identifiable Information (PII) is kept locally on the user's client (i.e., the browser or device) and is never revealed in the clear.

3.1 Overview

AdVeil is designed around *targeting*, *delivery*, and *reporting*. These are the three stages of the advertising pipeline where personal user information is involved (see Figure 1). Our overview of AdVeil follows the steps described in Figure 2, which transforms the advertising pipeline of Figure 1 into a privacy-preserving system.

Targeting (steps 1a to 1c in Figure 2) assigns relevant ads to a user. The client holds a user profile (step 1a) described as a *feature vector*. Ads are likewise associated with a feature vector describing their targeting attributes. The targeting protocol reveals nothing to the Broker and outputs a selection of ad IDs to the user's client. In addition, for each targeted ID, the Broker provides a blind signature on a special anonymous token with an embedded private metadata bit [57] indicating whether the client is identified as a "human" or a "bot". The client later returns the unblinded token (step 1c) to the Broker in reporting, which the Broker uses to discard "bot" reports.

Delivery (steps 2a to 2c in Figure 2) retrieves a targeted ad to display to the user corresponding to an ad ID (step 2a). The choice of which targeted ID to request (from the set of IDs received in targeting) is performed by the client with the help of a local *selection function* (see Sections 5.2 and 9). The client then sends the ID to the Broker via the anonymizing proxy to hide its IP address. The output of the delivery protocol is a targeted ad selected by the Broker in real-time (step 2b) for the targeted ID. This mechanism provides support for real-time bidding and other on-demand delivery logic (see Section 5.3). As in Targeting, the Broker provides a blind signature on a fresh anonymous token with embedded *public* metadata bits [81], binding the token signature to the returned ad. Later, in reporting, the unblinded token (step 2c) ensures that the report is uniquely associated to a delivered ad by examining the metadata.

Reporting (steps 3a to 3c in Figure 2) provides interaction reports (e.g., impressions and clicks; step 3a) to the Broker. To do so privately, the client batches and sends all reports to the Broker at fixed time intervals through the anonymizing proxy. The client also attaches the signed tokens obtained in the targeting and delivery protocols. All reports that the Broker deems fraudulent based on these tokens are discarded by the Broker. All remaining (i.e., valid) reports are used by the Broker for metrics and billing (step 3c).

3.2 Threat model and requirements

AdVeil has two independent security concerns: user privacy, which we define as *unlinkability* between users and their personal data, and *fraud prevention*, which we define as billing and reporting statistics that are resilient to botnets and false claims.

3.2.1 *Personal Data.*

Concretely, we define **personal data** to be any information that is related, either directly or indirectly, to a user's targeting profile. This includes elements of the user's feature vector used for targeting, which may contain demographics and interests, and are thus *directly* related to the user's profile. It also includes any ads the user sees or interacts with. This is because, given the nature of ad targeting, ad features are *indirectly* related to the user's profile through the targeting algorithm. AdVeil never reveals any personal data directly, even anonymized or unlinked from the client's identity. The Broker may indirectly infer profile features from the billing statistics on ads, however, these remain unlinked from the identity of the corresponding user (see Section 6).

3.2.2 *Adversarial Model.*

AdVeil assumes a rational [39, 44, 104] Broker that may try to link the identity of users to any of their personal data. A **rational** Broker is incentivized to provide correct functionality for its advertising ecosystem. The two assumptions we make about the Broker's behavior in order to provide user unlinkability in AdVeil are:

(1) the Broker does not deny service to honest users during ad targeting or delivery,
(2) the Broker includes all valid user reports when computing metrics on ad interactions.

The Broker has a direct financial incentive not to violate these two assumptions. If the Broker refuses to serve ads, then it cannot collect revenue from advertisers. Similarly, if some ad interactions from honest users are excluded from metrics, then the Broker violates the billing contract it has with Publishers and Advertisers.

3.2.3 *User Unlinkability.*

User privacy in AdVeil is defined in the context of *unlinkability* between users and their personal data. AdVeil guarantees that any data, e.g., ad interactions, that users report on cannot be tied back to their identity. Users in AdVeil are only required to report ad interactions and, as such, their reports do **not** need to contain any information about their profile by default.

Unlinkability for users in AdVeil is defined on a per-epoch basis where each epoch is the time period from the start of the targeting protocol through the end of reporting. Epochs hide timing correlations between AdVeil's protocols and ensure that all users participating in a given epoch are part of an *anonymity set*. The Broker cannot determine which user within an anonymity set was responsible for generating a set of reports.[1]

We require that users make delivery and reporting requests via an anonymizing proxy such as Tor[2] [35]; see Section 4. This ensures

[1]We discuss potential cross-epoch leakage resulting from intersection attacks [30, 31, 64, 66, 103] between the anonymity sets of different epochs in Section 6.1.3.

[2]Any anonymizing proxy hiding the sender identity is sufficient; see Section 4.

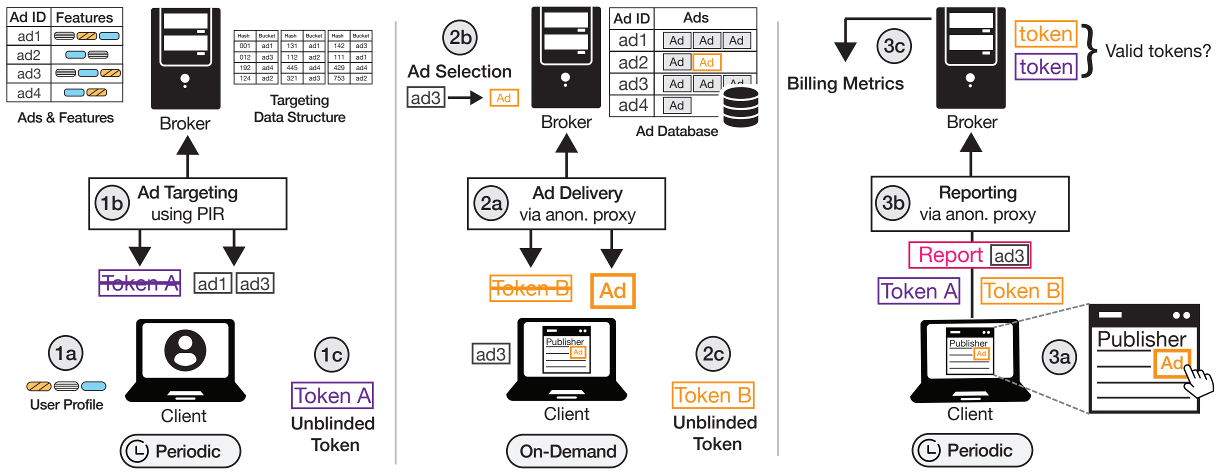


Figure 2: Overview of AdVeil. Description of the components is provided in Section 5.

that, while the Broker learns which ads are delivered and reported on, it does not learn the identity of the user who saw the ad.

3.2.4 Fraud Prevention. AdVeil expects users of the system to arbitrarily misbehave or to be impersonated by large scale bot networks. Users may try to avoid being shown ads at all or incur billing for ads that were never targeted or delivered.

AdVeil integrates a fraud-prevention mechanism using a combination of anonymous tokens [57, 81] that cleanly compose with existing bot detection mechanisms for fraud-prevention. Importantly, this is a *silent* method of fraud prevention. That is, while the Broker can recognize reports with invalid tokens, no other user can. This prevents bots from using AdVeil's tokens to learn which of their evasion strategies were successful.[3] In addition, AdVeil ensures that users only obtain tokens for the exact ads they were delivered and, as such, cannot generate valid reports on arbitrary ads.

In AdVeil, fraud-prevention guarantees that:

(1) the Broker can distinguish reports generated by a fraudulent request from those generated by an honest request,
(2) each valid report is included at most once in billing metrics,
(3) users can only generate valid reports for delivered ads.

We note that AdVeil does not make it easier for users to block ads. If users opt out of targeted advertising by not participating in AdVeil, then Publishers can fall back to *contextual* ads which do not require any user data or participation.

3.2.5 Non-goals. While AdVeil supports the return of arbitrary data during the reporting phase, it only requires reporting on the ads themselves for correct functionality. Determining what, if any, user data can be reported on privately is an orthogonal problem. AdVeil is compatible with any choice of data privacy mechanism such as differential privacy [36] or k-anonymity [89]. Similarly, while AdVeil guarantees that the ad identified in reports matches the ad delivered, we consider handling the integrity of any optional data submitted in reports to be out of scope.

[3]See https://github.com/WICG/trust-token-api#extension-metadata.

Finally, we cannot prevent users from intentionally disclosing PII to Advertisers and Publishers e.g., inputting their address to order an advertised product or logging into an account on a Publisher's website. AdVeil is neither intended to, nor capable of, preventing such behavior. As such, we see solutions for addressing this threat (e.g., through UI warnings and anonymous authentication) to be orthogonal research problems.

4 BUILDING BLOCKS

We design AdVeil around several standard cryptographic and data structure building blocks. Table 3 summarizes the building blocks and their use in AdVeil. Overall, we aim to make AdVeil general and modular, so as to fit a wide array of potential deployment scenarios.

For private targeting, we use single-server PIR [7, 59] in conjunction with a standard similarity search data structure based on locality-sensitive hashing (LSH) [6, 40]. We emphasize that using a LSH-based data structure ensures that AdVeil is compatible with targeting techniques used in practice [78, 99].

For private delivery and reporting, we use an anonymizing proxy (e.g., Tor [35] or I2P [106]), which reveals which ad was retrieved or reported on but not which client downloaded the ad.

For data integrity in both targeting and delivery, the Broker commits to all data structures using a vector commitment (e.g., Merkle tree) which can be efficiently verified by the client.

For fraud prevention, we use anonymous one-time-use tokens with public metadata. We use these tokens to enforce a "one-delivery; one report" policy for all client reports and ensure that each submission is uniquely associated with a prior delivery request.

4.1 Private information retrieval

Private information retrieval (PIR) is a standard cryptographic technique for retrieving items from a remote database or data structure without revealing *which* item was retrieved [15, 25, 26]. PIR in the single-server setting [59] with a database of N items requires $O(N)$ work on the server and sublinear communication (in practice $O(\sqrt{N})$

Building block	Purpose in AdVeil
LSH [6, 40]	Ad targeting based on user interests.
PIR [25, 59]	Private ad targeting with low bandwidth.
Anonymous tokens [57, 81, 95]	Fraud prevention in interaction reporting.
Vector commitment [71]	Integrity of targeting and delivery.
Anonymizing proxy [35, 97, 106]	Ad delivery and private user interactions.

Table 3: Summary of building blocks and their use in AdVeil.

communication [7]) between the client and the database. In AdVeil, PIR is used by the client to privately query the targeting data structure and retrieve a set of targeted ads.

Definition 1 (Private Information Retrieval [59] (informal))**.** *Let M be a set of instantiation-specific public parameters. For a fixed database $\mathcal{D}$ (or a key-value store [26]), PIR consists of the following functionality:*

$\underline{\mathsf{Query}}(M, \mathsf{idx}) \rightarrow (s, Q)$. Query *takes as input public parameters M and index* idx*. Outputs secret state s and query Q.*

$\underline{\mathsf{Answer}}(\mathcal{D}, Q) \rightarrow A$. Answer *takes as input a database $\mathcal{D}$ and query Q. Outputs answer A.*

$\underline{\mathsf{Recover}}(s, A) \rightarrow a$. Recover *takes as input secret state s and query answer A. Outputs recovered database item a.*

The functionality must satisfy correctness *and Informally, correctness holds if the computed answer produces the item at index* idx *in the database $\mathcal{D}$ when fed through* Recover*. Privacy for the client holds if the (potentially malicious) server learns no information on the query. We refer to Kushilevitz and Ostrovsky [59] for a formal definition.*

4.2 Anonymous one-time-use tokens

We make use of anonymous one-time-use tokens [57, 81, 95] for fraud prevention in AdVeil. These tokens are instantiated between a prover and a verifier. The verifier (e.g., the Broker) signs a blinded token generated by the prover (e.g., the client). During signing, the verifier can embed public metadata [81, 95] into the signature. Later, the unblinded token and the signature is redeemed by the verifier. The redeemed token is unlinkable to the originally signed token. When redeeming a token, the verifier can:

(1) check if the signed token is valid,

(2) check if the token was previously redeemed and,

(3) read the public metadata embedded in the token signature.

We use the randomized anonymous tokens of Kreuter et al. [57] based on Okomoto-Schnorr blind signatures [74] to achieve fraud prevention and rate limiting in AdVeil. We will use the public metadata extension of Tyagi et al. [95] when required. Specifically, AdVeil uses two distinct token schemes: one that supports valid/invalid tokens *and* and one that supports embedded public metadata. For simplicity, we present a single interface that captures both functionalities.

Definition 2 (Anonymous one-time-use token [34, 57, 81, 95] (informal))**.** *An anonymous one-time-use token scheme consists of efficient algorithms* Setup, KeyGen, TokenGen, Sign*,* Unblind *and* Redeem *with the following functionality:*

$\underline{\mathsf{Setup}}(1^{\lambda}) \rightarrow \mathsf{crs}$. Setup *outputs a common reference string* crs.

$\underline{\mathsf{KeyGen}}(\mathsf{crs}) \rightarrow (\mathsf{pk}, \mathsf{sk})$. KeyGen *outputs a new public key* pk *and secret token signing key* sk.

$\underline{\mathsf{TokenGen}}(\mathsf{pk}) \rightarrow (\tau, \bar{\tau}, r)$. TokenGen *outputs a new token τ, blinded token $\bar{\tau}$, and blinding randomness r using the public key* pk.

$\underline{\mathsf{Sign}}(\mathsf{sk}, \bar{\tau}, \mathsf{md}) \rightarrow \bar{\sigma}$. Sign *takes as input the secret signing key, blinded token $\bar{\tau}$, and (optional) public metadata* md*. Outputs signature $\bar{\sigma}$ on the blinded token $\bar{\tau}$ with public metadata* md.

$\underline{\mathsf{Verify}}(\mathsf{pk}, \tau, \sigma, \mathsf{md}) \rightarrow \mathsf{yes/no}$. Verify *takes as input the token, signature, and public metadata. Outputs* yes *if and only if σ contains embedded public metadata* md.

$\underline{\mathsf{Unblind}}(\mathsf{pk}, \bar{\tau}, \bar{\sigma}, r) \rightarrow (\tau, \sigma)$. Unblind *takes as input the public key, blinded token $\bar{\tau}$, blind signature $\bar{\sigma}$, and blinding randomness r. Outputs unblinded token τ and signature σ.*

$\underline{\mathsf{Redeem}}(\mathsf{sk}, \ell, \tau, \sigma) \rightarrow (\mathsf{md}, \ell)$. Redeem *takes as input a secret key* sk*, spent token list ℓ, token τ, and signature σ. Outputs whether the token is valid with respect to the spent list and signature, public metadata* md*, and updated spent list ℓ'. For notational convenience, we let* md = invalid *if the token τ or signature σ is invalid.*

The functionality must satisfy completeness, unlinkability, and unforgeability. A token may have no embedded metadata, in which case the metadata is denoted by the special symbol $\perp$.

Properties of one-time-use tokens. A one-time-use anonymous token must satisfy completeness, unforgeability, and unlinkability. Informally, completeness states that a verifier always accepts valid tokens using Redeem. Unforgeability states that a prover cannot forge a valid token or change the embedded metadata. Unlinkability requires that the verifier learns nothing beyond the metadata and token validity.

Constructions. We make a new observation on the Okomoto-Schnorr Privacy Pass (OSPP) token construction described by Kreuter et al. [57]. Our insight is that the OSPP-without-NIZK construction [57] is ideally suited to fraud-prevention. Using OSPP-without-NIZK, the Broker can prevent fraud by giving *invalid* token signatures to bots and *valid* signatures to honest clients during targeting. By the randomization of the token, clients cannot tell whether they received a valid or invalid token (thus not alerting bots to detection). Such tokens satisfy 2-unlinkability [57], meaning that all valid and invalid tokens are unlinkable within their respective anonymity sets. We provide further details in Section 6 and Appendix B.

4.3 Other tools

Locality-sensitive hashing (LSH) [6, 40, 50] is a technique used for efficient neighbor search in high-dimensional space, with applications to ad targeting. We use LSH as the building block in our targeting mechanism, described in Section 5.1.

Vector commitments [4, 13, 23, 71, 91] are a common technique for proving correctness of a value at a given index in a vector relative to a short commitmesnt. In AdVeil, we use a vector commitment (e.g., a Merkle tree) to ensure lookup consistency across client requests. The two properties that are important for this work are that each proof (1) is of logarithmic (or constant [4, 13, 23, 91]) size relative to the size of the vector and (2) each proof can be efficiently verified given only the commitment, value, and proof.

Anonymizing proxies [35, 97, 106] serve as an intermediary, or *proxy*, for communications between a client and server to hide the relationship between sender and recipient of messages. AdVeil only requires *client anonymity* [10] from the proxy. That is, the identity (IP address) of the client should be hidden by the proxy. The Onion Router (Tor) [35], Invisible Internet Project (I2P) [106], and VPN0 [97] all provide this property.[4]

5 SYSTEM ARCHITECTURE

This section introduces our construction of AdVeil. We compose AdVeil from protocols that instantiate the stages of the advertising pipeline, using the tools of Section 4.

First, in Section 5.1, we detail *non-private* nearest neighbor search which forms the foundation of our private ad targeting protocol between the client and the Broker through PIR.

In Section 5.2, we describe the protocol constructions used for targeting, delivery, and reporting. In Section 5.3, we explain how the protocols combine to form a complete ecosystem for private targeted advertising.

5.1 Targeting and nearest neighbor search

At the core of any targeting system is a nearest neighbor search data structure for matching ads to users [78, 107]. Hashing-based data structures solve *approximate* nearest neighbor (ANN) search. We note that ANN search is a general problem and can be applied to many definitions of similarity [6]. Formally, ANN search is defined over a set of high-dimensional feature vectors and a query vector $\boldsymbol{q}$. For a fixed distance metric an ANN search data structure returns the approximate nearest neighbor(s) to $\boldsymbol{q}$ from the set.

Ad targeting using ANN search is realized as follows. Let $\mathcal{D}$ be a database of N tuples of the form $(\mathsf{id}_i, \boldsymbol{v_i})$. Each tuple consists of an ad identifier id_i and a corresponding feature vector $\boldsymbol{v_i}$ describing the targeting attributes for the ad. To find the most relevant ad to a query $\boldsymbol{q}$ (where $\boldsymbol{q}$ is the user's profile vector), it suffices to find the nearest neighbor of $\boldsymbol{q}$, which we denote by $\boldsymbol{v_j}$, and output the corresponding ad ID id_j. This problem forms the basis for recommendation systems, including ad targeting [78, 99].

The data structure used for solving ANN search efficiently is based on locality-sensitive hashing (e.g., MinHash [19] or SimHash [24]). The data structure is defined by two algorithms: Build and Query. Because our targeting protocol requires *privately* querying the ANN data structure, we present a concrete instantiation of Build and Query as described in the seminal work of Gionis et al. [40]. In Section 5.2, we transform the Query function into a privacy-preserving protocol between the client and the Broker using PIR. We point to the survey of Andoni et al. [6] for further details and discussion on parameter selection.

```
ANN.Build($\mathcal{D}, \mathcal{H}, L$) → $\mathcal{S}$.
1: Define $L$ hash tables $T_1, \dots, T_L$ using LSH functions $\mathsf{h}_1, \dots, \mathsf{h}_L$ sampled i.i.d. from LSH family $\mathcal{H}$; // e.g., $\mathcal{H}$ = MinHash or SimHash
2: For each $\boldsymbol{v_i} \in \mathcal{D}$, compute $k_j \leftarrow \mathsf{h}_j(\boldsymbol{v_i})$, and append $(\mathsf{id}_i, \boldsymbol{v_i})$ to the bucket in hash table $T_j$ with key $k_j$;
3: Output $\mathcal{S} = \{T_1, \dots, T_L, \mathsf{h}_1, \dots, \mathsf{h}_L\}$.
```

```
ANN.Query($\mathcal{S}, \boldsymbol{q}$) → id.
1: Compute $k_j \leftarrow h_j(\boldsymbol{q})$ for $j \in [L]$;
2: Set $\mathcal{C} := B_1 \cup \cdots \cup B_L$ where each $B_j$ is the bucket in hash table $T_j$ with key $k_j$;
3: Output the id of the nearest neighbor to $\boldsymbol{q}$ in $\mathcal{C}$.
```

The key observation is that the client can individually query each hash table using PIR and locally recover the nearest neighbor(s).

5.2 AdVeil protocols

AdVeil realizes each stage of the advertising pipeline via separate and modular protocols. We first describe the context in which the protocols are instantiated.

5.2.1 The setting. With the exception of Protocol 1 (Setup), the protocols describe the steady state of AdVeil.[7] In this section, we begin by describing the background setting and starting assumptions.

Epochs. AdVeil divides time into discretized epochs. Aside from Protocol 1, which runs only once, all protocols run within the context of these epochs. The division of epochs is a deployment-specific parameter and only impacts the volume of users participating in each epoch. That is, epochs affect the frequency of ad targeting and reporting, but not *ad delivery* which remains on-demand.

Ad feature vectors. As a starting point, we assume that the Broker has a database of ads with associated feature vectors that describe the targeting attributes of each ad. How the feature vectors are obtained by the Broker is orthogonal to AdVeil. In practice, the feature vectors are provided by Advertisers directly or derived by the Broker through machine learning models. AdVeil is designed to be agnostic to the specifics of the targeting features in order to be compatible with a wide array of Broker strategies.

User profile. The user's profile feature vector is constructed locally by the client using a "profile building" function $\mathcal{F}_{\text{profile}}$, provided as part of the public parameters generated in setup (Protocol 1). $\mathcal{F}_{\text{profile}}$ maps user browsing history to a profile feature vector and can include information about websites they visit, searches performed, etc. [54, 58, 79, 99]. Ultimately, AdVeil leaves the final output of $\mathcal{F}_{\text{profile}}$ up to the users themselves as we require they retain full control over the use of their data. Unlike in existing targeted advertising, this profile is not sent to, nor collected by, the Broker.[8]

5.2.2 Setup (Protocol 1). The setup for the Broker involves publicly committing (e.g., through a public bulletin board [11, 27] or gossip network [60, 70, 72]) to all the parameters required for targeting, delivery, and reporting such that it cannot equivocate.

It begins with the Broker constructing the ANN search data structure for targeting. The Broker commits to the ANN search data structure hash tables and ad database (e.g., using a Merkle tree [71] or a vector commitments [23, 42, 61]). The Broker then generates a new token keypair (Definition 2) and makes the public key available

[4]Tor is the most widely deployed anonymizing proxy and is bundled by default in the Brave browser [17] and made available through extensions for Firefox[5] and Chrome[6]

[7]Protocol 1 can be re-run periodically to update targeting and ad database if needed.
[8]This mirrors other privacy-preserving targeting approaches [45, 75, 92, 99].

Protocol 1: AdVeil Setup

```
Step 1 (Broker)
// $\mathcal{L}$: feature vectors describing each ad
1: $\mathcal{S} \leftarrow \mathsf{ANN.Build}(\mathcal{L}, \mathsf{params})$
2: $C_\mathcal{S} \leftarrow \mathsf{VC.Commit}(\mathcal{S})$ // Vector commitment to hash tables
3: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{AT.KeyGen}(\mathsf{crs})$ // Anonymous token keys
4: $\mathsf{sk}^* \xleftarrow{R} \mathbb{Z}_p^*$ // invalid signing key; see Appendix B.
5: Publish: common reference string crs, public key pk, params, commitment $C_\mathcal{S}$, ad selection function $\mathcal{F}_{\mathrm{ad}}$, profile building function $\mathcal{F}_{\mathrm{profile}}$, and LSH functions $\mathsf{h}_1, \ldots, \mathsf{h}_L$ on a public bulletin board.
```

publicly. For simplicity, we assume that all public parameters are consistent and accessible to all clients via a public bulletin board.

5.2.3 Targeting (Protocol 2). Targeting occurs at the beginning of each epoch (e.g., once a day) and runs between the client and the Broker. The client obtains the ID(s) of relevant ads while the Broker does not receive any output. We present retrieval of a single targeted ad. We note that Protocol 2 can be trivially extended to output the top-k [40] ad IDs without significant overhead using batch-PIR [7, 51]. The client uses the ID(s) of the targeted ads to fetch ad *content* in Protocol 3 through an anonymizing proxy.

In addition to the ad ID(s), the client also receives the Broker's blind signature on the provided anonymous token(s). Each token signature is either valid or invalid, indicating to the Broker whether the client is believed to be a "bot" or a "human". The validity of each token signature is later used in Protocol 4 to reject bot reports. We note that the clients cannot determine the validity of the received token signature (see Appendix B).

To ensure integrity of targeting (i.e., that the Broker returns the correct set of targeted ads), the client checks the ID(s) it receives against the vector commitment $C_\mathcal{S}$. To facilitate this, we assume the Broker concatenates the proof string for each hash table bucket to the bucket contents. In this way, the client can retrieve the proof simultaneously with the bucket contents when privately querying the hash table through PIR.

5.2.4 Delivery (Protocol 3). Clients use Protocol 3 on-demand within epochs to retrieve ads corresponding to the ID(s) obtained from Protocol 2. The client fetches the corresponding ad from the Broker, via the anonymizing proxy. Importantly, the Broker has the ability to decide *which* ad to deliver based on the provided ID. We intentionally ensure that multiple ads can be associated with an ID so as to provide support for Broker-side selection logic (e.g., real-time bidding). Thus each ID represents a *class* of targeted ads. We discuss PIR as an alternate delivery mechanism in Section 7, but note that it does not provide the required concrete performance for on-demand delivery nor does it provide the ability to support ad selection logic.

5.2.5 Reporting (Protocol 4). Reporting occurs at the end of an epoch. When the user interacts with a retrieved ad through a Publisher's webpage (e.g., by viewing the ad or clicking on it), an impression report is generated and stored by the client. At the end of the epoch, the client sends each report to the Broker, via the anonymizing proxy, as specified in Protocol 4. The client includes the signed tokens obtained in Protocols 2 & 3. Upon receiving a

Protocol 2: Targeting

```
Step 1 (Client)
// $\boldsymbol{p}$: user profile feature vector held by the client
// $\mathsf{h}_i$: LSH functions in the public parameters
1: $k_i \leftarrow \mathsf{h}_i(\boldsymbol{p})$ for $i \in [L]$ // Compute profile hash keys
2: $Q_i \leftarrow \mathsf{PIR.Query}(M, k_i)$ for $i \in [L]$ // Query for bucket
3: $(\tau_t, \cancel{\tau}_T, r) \leftarrow \mathsf{AT.GenToken}(\mathsf{pk})$ // New reporting token
4: Send queries $Q_1 \ldots Q_L$ and $\cancel{\tau}_T$ to the Broker.

Step 2 (Broker)
// $\mathcal{S}$: ANN data structure with (committed to) hash tables $\{T_1, \ldots, T_L\}$
1: $A_i \leftarrow \mathsf{PIR.Answer}(T_i, Q_i)$ for $i \in [L]$ // Compute query answer
2: $\widehat{\mathsf{sk}} \leftarrow \begin{cases} \mathsf{sk}^* & \text{if client identified as bot} \\ \mathsf{sk} & \text{if client identified as human} \end{cases}$
       // Make invalid signature (bot) / Make valid signature (human)
3: $\cancel{\sigma}_T \leftarrow \mathsf{AT.Sign}(\widehat{\mathsf{sk}}, \cancel{\tau}_T, \perp)$ // Sign using the valid or invalid key
4: Send $A_1, \ldots, A_L$ and $\cancel{\sigma}_T$ to the client.

Step 3 (Client)
1: $(B_1 \| \pi_1, \ldots, B_L \| \pi_L) \leftarrow \mathsf{PIR.Recover}(A_1, \ldots, A_L)$
2: If there exists $i$ such that $\mathsf{VC.Verify}(C_\mathcal{S}, B_i, \pi_i) = \mathsf{no}$, abort.
3: id $\leftarrow$ nearest neighbor to $\boldsymbol{p}$ in $\mathcal{C}$ where $\mathcal{C} := B_1 \cup \cdots \cup B_L$.
4: $(\tau_t, \sigma_t) \leftarrow \mathsf{AT.Unblind}(\mathsf{pk}, \cancel{\tau}_T, \cancel{\sigma}_T, r)$ // Unblind signature
5: Output $(\mathsf{id}, \tau_t, \sigma_t)$
```

Protocol 3: Delivery

```
Step 1 (Client)
// id : targeted ID obtained from Protocol 2
1: $(\tau_d, \cancel{\tau}_d, r) \leftarrow \mathsf{AT.GenToken}(\mathsf{pk})$ // New delivery token
2: Send $(\mathsf{id}, \cancel{\tau}_d)$ to the Broker, via the proxy.

Step 2 (Broker)
// $\mathcal{D}$: database of ads
1: ad $\leftarrow \mathsf{SelectAd}(\mathcal{D}, \mathsf{id})$. // Broker's ad selection logic for ad ID
2: $\cancel{\sigma}_d \leftarrow \mathsf{AT.Sign}(\mathsf{sk}, \cancel{\tau}_d, \mathsf{id})$ // Sign with public metadata id
3: Send $(\mathsf{ad}, \cancel{\sigma}_d)$ to the client, via the proxy.

Step 3 (Client)
1: $(\tau_d, \sigma_d) \leftarrow \mathsf{AT.Unblind}(\mathsf{pk}, \cancel{\tau}_d, \cancel{\sigma}_d, r)$ // Unblind signature
2: If $\mathsf{AT.Verify}(\mathsf{pk}, \sigma_d, \mathsf{id}) = \mathsf{no}$, abort. // Invalid public metadata
3: Output $(\mathsf{ad}, \tau_d, \sigma_d)$.
```

report, the Broker verifies the tokens, rejecting all reports with invalid (i.e., "bot") tokens. All reports with valid and a yet-unspent token are accepted.

5.3 The AdVeil ecosystem

AdVeil focuses on *general* targeted advertising, with support for multiple targeting and metrics strategies.

Targeting strategies. By giving the Broker control over the choice of ANN search data structure and real-time ad selection logic, AdVeil does not restrict the Broker to a specific targeting algorithm while still

Protocol 4: Reporting

Step 1 (Client)

```
// report: report payload
// (τ_t, σ_t): token and signature obtained from Protocol 2
// (τ_d, σ_d): token and signature obtained from Protocol 3
1: Send $(\text{report}, \tau_t, \sigma_t, \tau_d, \sigma_d)$ to the Broker, via the proxy.
```

Step 2 (Broker)

```
// (ℓ_t, ℓ_d): lists of all redeemed tokens
1: $(\text{md}_t, \ell'_t) \leftarrow \text{AT.Redeem}(\text{sk}, \ell_t, \tau_t, \sigma_t)$ // $\text{md}_t \in \{\text{valid}, \text{invalid}\}$
2: $(\text{md}_d, \ell'_d) \leftarrow \text{AT.Redeem}(\text{sk}, \ell_d, \tau_d, \sigma_d)$ // Public metadata
3: If $\text{md}_d = \text{valid}$ or $\text{md}_d = \text{invalid}$: reject. // Invalid token
4: If $\text{id}' = \perp$: reject. // No metadata in delivery token
5: Else, set $\ell_t \leftarrow \ell'_t$ and $\ell_d \leftarrow \ell'_d$ and accept report.
```

ensuring privacy for users. Types of targeting that can be supported in AdVeil include:

- **Contextual** targeting in AdVeil can bypass users entirely by having the Publisher fill their role in the protocols. Users will still see and be able to interact with ads, but will have no involvement in any other aspect of the pipeline.
- **Behavioral** targeting in AdVeil is achieved through the profile building function $\mathcal{F}_{\text{profile}}$ which is able to locally observe and record information about users' browsing habits forming a profile feature vector for the user. Protocol 2 is then run periodically to retrieve a new set of targeted ads based on the user's profile. Retargeting, or preferentially displaying ads to users who have had prior interactions with the Advertiser, is one example of behavioral targeting that AdVeil can support in this manner.

As mentioned in Section 5.2, the only requirement for targeting strategies is that they must be fully local. Users must not make any requests other than those specified in Protocol 2, Protocol 3, and Protocol 4 as a consequence of any targeting strategy.

Real-time bidding in AdVeil is supported automatically by giving the Broker full control over which ad to deliver to the client based on the ad ID requested. The Broker can, in real time, select from a set of ads to respond with in Protocol 3 for a given ad ID. That is, while the targeted ID is fixed, the associated ad content need not be. Because the delivery and reporting requests are unlinkable from the targeting request, delivering different ads for the same ad ID does not compromise unlinkability, see Section 6.

Measurement strategies. Reports in AdVeil are *individual* without being *linkable*. The Broker learns precisely which ads were seen and how often, without ever learning who saw them. This allows the Broker to support a variety of measurement strategies including:

- **Impression** reports generated when users *see* a displayed ad.
- **Click** reports generated when users *click* on a displayed ad.
- **Conversion** reports generated when a user engages with the Advertiser after clicking on an ad. The user's local client is capable of observing when an ad click generates a conversion event and creates the resulting report.[9]

[9]This is similar to how conversions are tracked in the Safari browser today [102].

All reports in AdVeil are assumed to contain, at minimum, the ad delivered and interaction type (e.g., Impression or Click). There is a large body of *other* information the Broker may wish to receive as part of reporting. AdVeil supports reporting on arbitrary information but, as discussed in Section 3.2, is not intended to provide *data* privacy guarantees. While it is always possible to report on user data privately using differential privacy [36] or k-anonymity [89], such techniques are deployment specific and orthogonal to AdVeil's primary goal of providing unlinkability between users and reports.

6 SECURITY ARGUMENTS

In this section we analyze the security of AdVeil when instantiated using the protocols of Section 5.2. We frame our analysis in terms of *user unlinkability* and *fraud prevention*, the two requirements outlined in the threat model of Section 3.2.

6.1 User unlinkability

We recall that the security guarantee of AdVeil, as established in Section 3.2, is *unlinkability* between users and their personal data, including which ads they see and interact with.

We show in Claims 1 and 2 that, individually, none of the protocols are linkable to personal user information. Therefore, the crux of the unlinkability argument lies in analyzing the combination of targeting (where the Broker learns the client's identity; required for effective fraud prevention) and delivery/reporting (where the client's identity is hidden by the anonymizing proxy). Concretely, we must ensure that the Broker cannot deviate in targeting to link the client in delivery or reporting, without also compromising its own goals (and violating the *rationality* assumption; see Section 3.2).

Theorem 1 (Unlinkability). *Fix a reporting epoch. The set of recovered reports through Protocol 4 in the epoch is unlinkable from the set of clients that submitted them as well as prior executions of Protocol 2 and Protocol 3, conditioned on:*

(1) the user and client do not explicitly or implicitly leak personally identifying information to any party,

(2) the Broker is rational in the sense that it does not deny service to honest users or self-sabotage the fraud prevention mechanism,

(3) the privacy of PIR [59], soundness of the vector commitment [23, 61, 71], unlinkability of one-time-use tokens [57, 81, 95], and client anonymity property of the anonymizing proxy [10], all hold under their respective assumptions.

6.1.1 *Privacy of individual protocols.*

We first show that the targeting, delivery, and reporting protocols, *individually* provide unlinkability from user data.

Claim 1. *Protocol 2 (targeting) reveals the identity of the client but no other information to the Broker, conditioned on the privacy requirement of the PIR scheme.*

Proof. The claim follows almost immediately from protocol inspection. Specifically, in Protocol 2 (targeting), the client only interacts with the Broker through a series of PIR queries which, by definition, hide the user's query (i.e., profile) from the Broker. The anonynous token issued in Protocol 2 is generated by the client independently of all user data and hence reveals no information on its own. ■

Claim 2. *Protocol 3 (delivery) and Protocol 4 (reporting) reveal the ad delivered and report contents to the Broker, but not the identity of the client, conditioned on the client anonymity property of the anonymizing proxy [10].*

Proof. In Protocols 3 & 4, the client interacts with the Broker through the anonymizing proxy, which reveals the targeted ad ID and which ad was served (resp. the contents of the report) but not which client it was served to (resp. which client submitted the report). This follows directly from the client anonymity property of the anonymizing proxy [10]. The anonymous tokens generated and sent by the client in both the delivery and the reporting protocols contain no identifiable information as they are computed independently of the user data (and are uniformly random). ■

6.1.2 Proof of Theorem 1. The unlinkability argument hinges on showing that a malicious (but rational) Broker cannot deviate in the targeting (Protocol 2) to link the client in delivery (Protocol 3), or reporting (Protocol 4). This is shown in Lemma 1.

Lemma 1. *Fix a reporting epoch. The Broker cannot rationally deviate from Protocol 2 to link the client to a delivery request (Protocol 3) or report submitted through Protocol 4, assuming the soundness of the vector commitment [23, 61, 71] and the unlinkability property of anonymous tokens [57, 81, 95].*

Proof. At a high level, Lemma 1 follows from the Broker committing to the ANN hash tables in Protocol 1 (setup) and the total number of ads, coupled with the unlinkability property of anonymous tokens. More formally, we must individually examine interactions taken by the client and the Broker in an epoch.

First, because the ANN hash tables are committed to in Protocol 1, a malicious Broker cannot change the ANN search data structure between client requests. As such, the contents of the targeting data structures must be *consistent* across all clients and, consequently, the contents of all clients' reports must also be consistent with the fixed targeting data structures. This is guaranteed by the vector commitment proof returned to the clients with their PIR queries. If the Broker is capable of answering the PIR query with a valid proof (w.r.t. the commitment $C_{\mathcal{S}}$) for a bucket value that is *not* in the hash table, with better than negligible probability in a security parameter, then the Broker is also capable of breaking the soundness property of the vector commitment with non-negligible probability [23, 61, 71].

Second, we examine the anonymous token (and the embedded public metadata) as a vector for linking a client to a report. We show that exploiting anonymous tokens for linking clients to reports, while possible, is monetarily disincentivized and hence falls under irrational behavior. Specifically, because each token issued in Protocol 2 is set up to reveal *one* bit of information (valid or invalid), the Broker can only partition the anonymity set into two groups: clients with valid tokens and clients with invalid tokens (see Appendix B for additional details). This division is necessary for any scheme that allows the Broker to silently tag fraudulent requests for later identification. However, all tokens within these two sets are unlinkable from other tokens in their respective set by the properties of anonymous tokens [57] (also Appendix B); hence the Broker cannot link a valid (resp. invalid) token to a prior targeting request.

A malicious Broker can still split a single client into their own group (by issuing only one valid token), allowing it to directly link their identity to the contents of their report. However, this sabotages fraud prevention, as the Broker must group all other users **and** all bots into the other set (invalid tokens), losing its ability to distinguish fraudulent reports and causing it to violate its billing arrangement with Advertisers and Publishers. This strategy would result in significant monetary losses for the Broker, as it sabotages correctness for its advertising ecosystem. As such, a *rational* Broker will only use the anonymous to tag fraudulent requests. ■

As a consequence of Lemma 1, the Broker cannot deviate from targeting without either compromising fraud prevention or denying service to a client (i.e., causing the client to abort). Such behavior, however, is inherently irrational as it either 1) prevents users from being shown ads or 2) sabotages fraud prevention resulting in potentially inaccurate billing metrics. As a result, a rational (monetarily-driven) Broker is incentivized to follow protocol ensuring the unlinkability property is satisfied between clients and generated reports.

6.1.3 Cross-epoch unlinkability. If only a subset of all users participate in each epoch, then AdVeil cannot provide full unlinkability across epochs due to intersection attacks [30, 31, 64, 66, 103]. In this case, intersection attacks become possible because users are correlated with the ads they see. Over time, the Broker can infer the relationship between a user and the ads they are shown by observing the *intersection* of epochs in which the user participated, *even though the unlinkability property is satisfied within each epoch.* That is, ads that appear most frequently alongside a certain user are likely to be the ads that user reported on. This leakage is small, but not resolvable without holding the set of either users or ads constant across all epochs—neither of which is reasonable for an internet-scale system.

6.1.4 Unlinkability of features. AdVeil guarantees that the Broker cannot link any user to any of their personal data, but does not make guarantees about any associations between the data elements themselves. That is, the Broker may learn relationships between different *features*, e.g. that "`mechanical keyboards`" and "`programming`" are two distinct features that commonly occur together in user profiles.

More generally, the Broker can determine whether there exists *some* user who has a specific set of n features by generating a "tagged" ad ID that is targeted to exactly that set of features and no others. The Broker can then observe whether a report corresponding to the tagged ad ID appears during the reporting phase. If so, there exists at least one user that has the tagged set of n features in their profile. This does not reveal information about *which* user or users are involved, but allows the Broker to learn feature clusters, or sets of features that commonly appear together in user profiles.

6.2 Fraud prevention

Recall that the requirements for fraud prevention (detailed in Section 3.2), are that 1) the Broker is capable of flagging all targeting requests detected as coming from bot clients and 2) no duplicate or bot reports are recovered through Protocol 4.

Theorem 2 (Fraud Prevention). *Assuming the unforgeability property of anonymous tokens [57, 81, 95], each report recovered by the Broker through Protocol 4 is:*

(1) submitted by a client given a valid token signature in Protocol 2,
(2) unique in the set of all reports submitted across all epochs,
(3) associated with an ad that was delivered in Protocol 3,

Proof. The association of each report to a unique prior execution of Protocol 3 is guaranteed by the unforgeability property of the anonymous tokens; no client can forge a valid token signature without knowledge of the secret signing key held by the Broker [57]. Report uniqueness is likewise guaranteed by the anonymous token unforgeability property and the list ℓ of all redeemed tokens maintained in Protocol 4 (see Privacy Pass [34] for more details and optimizations). Finally, the anonymous token unforgeability property guarantees that no client can forge a valid signature on a token. To expand on this final point, only clients marked as "human" in Protocol 2 are given valid signatures on the token. All other clients are given "invalid" signatures (which are indistinguishable from valid signatures to the client, see Appendix B); in essence, valid/invalid signatures partition all clients into two anonymity sets. In Protocol 3, the Broker issues a signature with *public* metadata containing the delivered ad ID. Due to the unforgeability property [81, 95], this fixes the ad ID, which in turn means that the client cannot report on a different ad from the one requested through Protocol 3. ■

6.3 Correctness and efficiency

In this section, we briefly argue correctness and asymptotic efficiency of the protocols composing AdVeil.

6.3.1 Correctness. Correctness of AdVeil follows immediately from the correctness of the underlying building blocks. Accuracy of targeting is inherited from the correctness of the similarity-search data structure of Section 5.1 and PIR [25]. Correctness of delivery follows from correctness of the anonymizing proxy routing traffic between the client and the Broker. For reporting, we note that the client submits the report which, at minimum contains the ad ID embedded in the token signed in Protocol 3 (delivery). By the correctness property of the anonymous tokens (signed in targeting and delivery), only valid reports are accepted by the Broker. The correctness of the recovered metrics thus follows from the correctness of the targeting and delivery protocols.

6.3.2 Asymptotic Efficiency. We provide a brief analysis of *asymptotic* efficiency of AdVeil, in terms of the efficiency of the:

(1) ANN search data structure used in targeting,
(2) PIR scheme used to query the data structure,
(3) anonymizing proxy used in delivery and reporting,
(4) reporting token redemption and storage overheads.

Only the first three factors impact the efficiency for the end user. Asymptotically, the guarantees of the similarity search data structure [40], and the underlying PIR scheme [7, 59], result in communication of $O(N^{\epsilon})$ for $\epsilon > 0$. In practice, $\epsilon \approx \frac{1}{2}$ [6, 7]. If the anonymizing proxy is instantiated using Tor [35], then the overhead on the client is $O(1)$ in delivery and reporting. The Broker's work for targeting ads is $O(N)$ due to lower-bounds in PIR [25] and $O(1)$ for delivery and reporting. The work and space imposed on the Broker in reporting is $O(R)$ for an epoch containing R submitted reports in total (note: in practice, key rotation can be used to prevent storing reporting tokens across different epochs for de-duplication purposes; see [95]).

7 EVALUATION

We implement a prototype of AdVeil and measure its end-to-end performance in a networked deployment. We evaluate the computational overhead of the Broker when serving client requests and the end-to-end latency of targeting and delivery on the client. We note that we do not compare AdVeil *quantitatively* with existing work in privacy-preserving advertising. Prior proposals are either of a theoretical/qualitative nature [53, 100], only solve one aspect of the pipeline under different assumptions [43, 47, 93], or have incomparable approaches [45, 47, 92]. We instead *qualitatively* compare all these systems with AdVeil in Section 8.

7.1 Setup

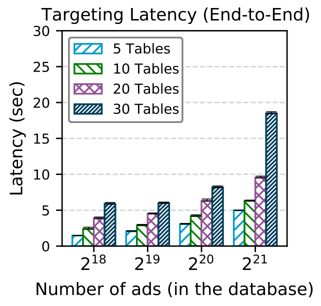
Implementation. We implement AdVeil in Go (v1.13) and C++17. Our implementation is open source and available at http://adveil.com/code. We use the open-source Microsoft SealPIR library [7] to instantiate single-server PIR. Our implementation of anonymous one-time-use tokens is partially based on Cloudflare's Privacy Pass code [34]. We assume the Broker commits to the targeting data structure hash table using the vector commitment of Libert and Yung [61], which require a one-time trusted setup but have very succinct opening proofs of 48 B. Concretely, short proofs improve the performance of targeting given that the user retrieves the proof in conjunction with the value from the hash table via PIR.

Environment. We deploy AdVeil on Amazon Elastic Cloud Compute (EC2) server instance for the Broker and a MacBook Pro for the client. The Broker's server runs on `c5a.8xlarge` VM (32 vCPUs @ 3.6GHz; 64 GB RAM) with an hourly cost of $0.525 as of June 2021.[10] We run the client on a MacBook Pro (8 CPUs; 32 GB RAM). We measure a throughput of 200 Mbit/s download and 5 Mbit/s upload using the `speedtest-cli` tool.[11]

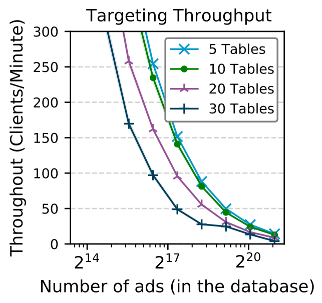
Parameters. The tunable parameters in AdVeil include the number of ads, the size (in KiloBytes) of each ad, and the number of hash tables used in the ANN search data structure (see Section 5.1). The total number of ads and ad size has a direct impact on network bandwidth usage and server processing time due to PIR (see Section 4.1). Likewise, for the ANN search data structure, more hash tables improve accuracy, but also increase the total number of parallel PIR queries required (see Protocol 2). In our runtime experiments, we vary the number of hash tables from $L = 5$ to $L = 30$. We report the accuracy of the ANN search protocol while varying the parameter L in Figure 6. We make each hash table in the ANN search data structure have N keys (i.e., total number of ad IDs) and cap the size of each hash table bucket to one ID.

Datasets. We evaluate AdVeil using synthetic data. Real-world ad features are proprietary [99] and we were not able to find suitable data for our evaluation. We stress, however, that the performance of PIR (which is used to query the ANN search hash tables) is not impacted by the underlying data distribution. Moreover, AdVeil does

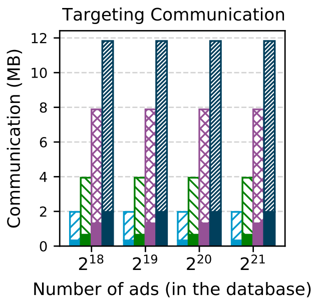
[10]https://aws.amazon.com/ec2/spot/pricing/
[11]https://www.speedtest.net/apps/cli



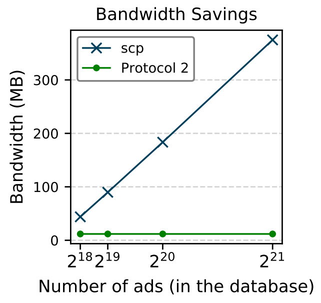
(a) Client end-to-end latency (seconds) for a targeting query. Latency is primarily impacted by Broker computation time (computing query answers), but also affected by ping time and client network bandwidth.



(b) Broker targeting throughput (clients per minute) with 32 cores. Throughput is only impacted by PIR query answer computation which is linear in the number of ads in the database.



(c) Client-Broker communication for targeting (PIR query and response size). Bottom solid areas of each bar indicate fraction of upload communication.



(d) Communication comparison between "naive" PIR (retrieving entire ANN search data structure (scp)) and Protocol 2, which uses SealPIR queries. We fix $L = 30$.

Figure 4: Evaluation of ad targeting in AdVeil. We report end-to-end latency as measured on the client machine. Communication is measured as the network overhead between the client and Broker. Throughput measures the raw computational processing throughput of the Broker's server (parallelized across 32 cores) to answer client targeting requests. Shaded regions and error bars represent a 95% confidence (occasionally invisible).

not impact *accuracy* of targeting and hence is also agnostic to the underlying features.

In contrast, the number of hash tables in the ANN search *does* impact accuracy (which directly impacts AdVeil's performance by requiring more PIR queries). To gain a sense of how many hash tables are required in a deployed setting, we follow the synthetic data generation and evaluation methodology performed by Datar *et al.* [32] for evaluating ANN search queries over Euclidean distance in a *worst-case* setting. Each ad feature vector is randomly generated by sampling a 100-dimensional vector with random coordinates in the range $[-255, 255]$ (each coordinate is one byte). A client profile feature vector is then randomly "planted" within a fixed distance radius from a randomly chosen ad feature vector. Because LSH performs best on *clustered* data (as is often the case with non-synthetic data), generating the dataset in this way results in *worst-case* recall [6]. This setup is used to evaluate *accuracy* of approximate nearest neighbor search on specific LSH parameters[12] as a function of the number of hash tables to gain a crude estimate for the number of tables required in practice.

Ad sizes. Common online banner ad sizes[13] (which account for 89% of ads on some platforms [65, 69]) are typically under 150 kB in size. Video ads are typically under 4 MB in size [69]. Ad size only impacts delivery as all other protocols operate on fixed-size ad IDs.

Methodology. Unless otherwise stated, we run each experiment 10 times and report mean and 95% confidence interval over the trials. We parallelize computation on the servers when possible. However, we note that our implementation leaves room for additional parallelization and optimizations.

7.2 Results

In this section we describe our evaluation results for targeting, delivery, and metric recovery in terms of processing time, latency, and communication. We also report the impact of changing the number of LSH hash tables has on targeting accuracy.

7.2.1 Client latency. We first evaluate client end-to-end latency in wall-clock time for ad targeting and delivery.

Targeting (Protocol 2) latency. Figure 4a shows the impact that the number of ads in the dataset and number of ANNS hash tables has on client latency. Latency ranges from several seconds on smaller sets of ads ($\approx 130,000$ ads) to 18 seconds on larger sets ($\approx 2,000,000$ ads). Network throughput accounted for 1 to 8 seconds of end-to-end latency on the client (recall that the client has throughput comparable to a home WiFi network).

Delivery (Protocol 3) latency. Figure 5 shows the relative performance of Tor vs. single and two server PIR for delivering ads. We take the mean latency for downloading a 50 kB and 1 MB file over the Tor network from Tor metrics [63] data between February and May 2021. Delivery latency for a 50 kB ad (e.g., a small image) through Tor is approximately one second while latency for a 1 MB ad (e.g., 5 second 480p video[14]) is approximately three seconds.

To illustrate the impracticality of using PIR for delivering ads, we evaluate SealPIR over 500 B to 1 kB ads while varying the total number of ads in the database. Additionally, we consider two-server PIR, which is more concretely efficient in practice [15, 98]. Our results show that both single-server and two-server[15] PIR impose a high overhead for delivery, even when only considering small (up to 1 kB) ads. As such, we choose to use Tor for ad delivery in AdVeil. We stress that *targeting* requires use of PIR as replacing it with Tor (or any other anonymizing proxy) would require user data to be sent to the Broker during targeting, which would directly reveal PII.

7.2.2 Communication overhead. We report the total communication (in MB) exchanged between the Broker and the client when targeting ads in Figure 4c. Communication depends on database size and

[12]Many factors are at play when determining optimal LSH parameters; see [83] for details on parameter optimization for ANN search in practice.

[13]https://support.google.com/adsense/answer/6002621

[14]See https://www.animotica.com/blog/full-guide-what-is-video-bitrate-and-why-does-it-matter/ for video bitrates.

[15]The two servers are required to be non-colluding in this setting.

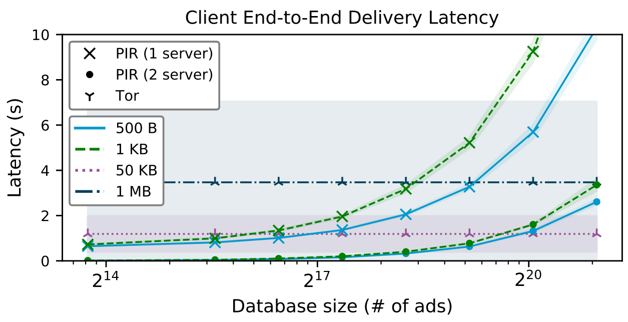


Figure 5: Client end-to-end ad delivery latency (seconds) using PIR, two-server PIR, and Tor. Latency for PIR queries is proportional to the size of the ad database (number of total ads and their size). Latency for Tor is only dependent on ad size (independent of the total number of ads). Shaded region represents a 95% confidence interval (note that Tor has high variance in latency).

number of tables. The total communication remained under 12 MB. We find that the majority of communication is from the PIR query response since the PIR queries themselves are of constant size with respect to the database size in our evaluation (due to SealPIR query compression [7]). We believe this communication to be reasonable for the average internet client, especially considering targeting happens once per epoch (e.g., once a day) and is "download-heavy", which aligns well with real-world networks [84]. In Figure 4d we contrast the bandwidth usage of SealPIR with "naive PIR" where the entire ANN search data structure is sent to the client.

7.2.3 Targeting throughput. We report the targeting throughput (in terms of clients per minute) of the Broker in Figure 4b. Throughput is limited by the computational overhead of processing the PIR queries over the ANN search hash tables. With 30 hash tables and 2^{20} ads, targeting throughput was approximately 12 clients per minute. We note that the throughput is massively parallelizable; increasing linearly with the Broker's computational capacity. If we assume that targeting is required once per week and 100 million active users in the ecosystem (with anonymity sets of 2^{20} unique ad IDs), the Broker would need $\approx$ 800 servers to target ads on a weekly basis.

7.2.4 Targeting accuracy. We compare targeting accuracy in AdVeil to that of contextual advertising as users' features become increasingly distant from those of the website. ANN search accuracy is typically measured through *recall*: the fraction of neighbors found over the total number of queries performed. While AdVeil is agnostic to the ANN search parameters (such as the LSH family used) we nonetheless measure the accuracy of nearest neighbor queries as a function of the number of hash tables used in the ANN search data structure. We first fine-tune the ANN search parameters to achieve recall accuracy of over 90% with $L = 30$ hash tables and report the drop in recall as we decrease the number of hash tables. We report recall accuracy across different values of L in Figure 6, evaluated over 10,000 independent queries while varying the number of hash tables in the ANN search data structure. We find diminishing returns in recall improvement when the number of hash tables increases over 20; suggesting that on real-world data (not worst-case as evaluated here), 20 hash tables in the ANN search may be sufficient for accurate targeting.

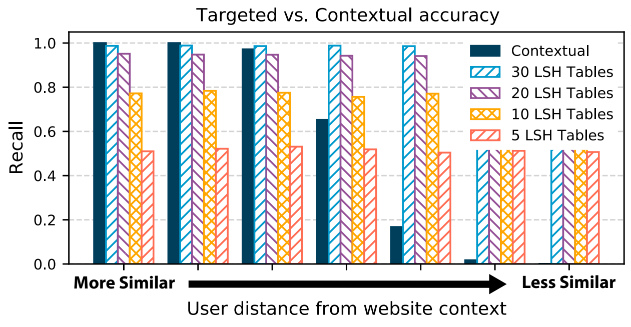


Figure 6: Recall accuracy comparison between targeted and contextual ads as a function of the user's profile distance from the website's profile. Increasing the number of hash tables has diminishing returns on recall accuracy for targeted ads. As user profiles grows less similar (i.e., more distant) from the website profile, contextual advertisement become less relevant. Targeted advertisement relevancy is independent of the user's similarity to the website context.

7.2.5 Reporting. The computation on the Broker per submitted reports is light and consists of redeeming the two tokens attached to each report. In total, redeeming the tokens for a single report requires under 300 μs of processing time on one core (see Table 7). Thus, a single 32 core server can process upwards of 90,000 reports per second when parallelized.

7.2.6 Microbenchmarks. We execute a series of microbenchmarks to analyze the impact of the building-blocks used in AdVeil. Specifically, we measure the client processing time for PIR queries and the anonymous tokens. Generating the PIR queries requires under one millisecond. Decryption of the PIR query response takes approximately two milliseconds. All client-side token processing (generating and unblinding) is under 300 μs. Server-side processing of tokens requires under 300 μs for all operations.

Microbenchmarks			
Client			
879 μs PIR.Query	2007 μs PIR.Recover	252 μs AT.GenToken	328 μs AT.Unblind
Broker			
278 μs AT.Sign (no metadata)	167 μs AT.Redeem (no metadata)	178 μs AT.Sign (public metadata)	151 μs AT.Redeem (public metadata)

Table 7: Microbenchmarks (in microseconds) for SealPIR (query and answer recovery) and anonymous one-time-use tokens (with and without public metadata).

7.2.7 Storage overhead. We report the storage overhead in Table 8. The client stores the user's profile feature vector locally which requires d bytes (assuming one byte per profile vector coordinate). Additionally, the client stores the PIR key pair and public parameters published by the Broker in Protocol 1. The Broker stores all redeemed tokens in a spent list to prevent duplicate reports (the storage cost can be reduced with a bloom filter [34]). Additionally, we assume the Broker stores the PIR public keys of each client to avoid having the client to repeatedly send them with each PIR query.

Storage Overhead				
Client			Broker	
d (B)	4 (MB)	200 (KB)	$32R$ (B)	$4C$ (MB)
Profile vector	SealPIR key	Public params	Token spent list.	Client keys

Table 8: Storage overhead. C is the total number of clients in the system. R is the total number of reports submitted in an epoch. $d = 100$ in our evaluation.

7.2.8 Operational costs. We estimate the operational costs of running AdVeil. Our estimate focuses on an ad database and reports-per-epoch size of 1M ads. We assume the Broker uses 30 LSH tables and use the AWS costs from our own evaluation – \$0.525/hr for the Broker's machine on AWS. Given this setup, the processing time of the Broker is approximately 4.5 seconds per execution of the targeting protocol (Protocol 2).[16] The total costs to target, and recover reports for, $k = 1$ ads is computed as in Equation (1) and equals 0.07¢.

$$\underbrace{(\$0.525/\text{HOUR}) \cdot 4.5 \text{ s}}_{\text{cost of targeting } k \text{ ads}} + \underbrace{k \cdot (\$0.525/\text{HOUR} \cdot 0.5 \text{ ms})}_{\text{cost of token signing and redemption}} \tag{1}$$

Compared to average revenue generated by an ad impression of approximately 0.20¢ [86], we get that the expected revenue is roughly 2.85× the cost of serving a single ad (on a non-enterprise server). Amortized over $k = 10$ ads targeted simultaneously (see Gionis et al. [40] for how ANN search trivially extends to k-nearest neighbor search), the expected revenue is over 20× the cost of targeting. Hence, AdVeil can be deployed (with off-the-shelf cloud infrastructure) and *still* result in net profit gains for the Broker.

8 RELATED WORK

There exists a large body of works on privacy-preserving advertising [9, 14, 43, 45–47, 53, 75, 92, 93, 99, 100]. In this section we focus on providing a detailed comparison to other systems that cover the complete advertising pipeline and leave discussion of the less similar works to Appendix A.

Jules [53] is an early, theoretical proposal for supporting privacy-preserving targeted advertising on the internet. It is primarily of historical interest, but shares some similarities to AdVeil. Specifically with respect to the use of PIR (which is applied to *delivery* of ads). We show in Section 7, that PIR is not practical for ad delivery.

Privad [45] provides a targeting model based on broad interest categories that are narrowed locally by the client. Privad relies on a *centralized* anonymizing proxy, referred to as the Dealer, to provide user privacy and enforce fraud prevention. The Dealer is assumed not to collude with the Broker and provides user privacy by mediating *all user communication*. Thus, the Broker learns the contents, but not the origin, of each request. To do this, the Dealer must sustain the load of the entire ad network as a non-colluding third party. Privad cannot trivally solve this issue by replacing the Dealer with an existing, distributed anonymizing proxy as this would leave their system without fraud prevention.

[16] Note that Figure 4a includes network latency in addition to processing time.

OblivAd [9] relies heavily on a Trusted Execution Environment (TEE) to provide user privacy. The TEE is single-handedly responsible for *all stages of the advertising pipeline*, from ad targeting to unlinkability of reports and fraud prevention. To prevent the Broker from learning which ads are delivered to which clients, the TEE uses ORAM [87] to retrieve ads from the ad database. For reporting, users again encrypt their responses to the TEE which batches and shuffles them to hide timing information.

ObliviAd provides strong privacy guarantees only because of its reliance on the TEE and ORAM, both of which have major practical drawbacks. ORAM can only serve a single client request at a time; parallel requests require an equivalent number of separate instances of the ORAM scheme causing linear storage blowup in the number of concurrent client requests [8, 87]. TEEs have seen a series of powerful attacks since ObliviAd was published [16, 21, 28, 96]. As a result, we do not believe that either issue is surmountable with today's instantiations of TEEs or ORAM [8, 28, 87].

Adnostic [92] primarily focuses on the targeting and reporting stages. Similarly to Privad, Adnostic performs targeting locally on the client. However, Agnostic only uses *contextual* features during targeting and does not make an effort to hide which ads are delivered to a user. Adnostic attempts to prevent the Broker from linking the delivery and reporting phases by aggregating reports using homomorphic encryption and zero-knowledge proofs rather than revealing them individually. Decryption of aggregate reports is handled by a TTP that checks the size of reports prior to decryption to ensure that only large groups of users are reported on. However, Adnostic makes no effort to hide users' browsing or click behavior from the Broker, leading us to agree with Privad's description of their Broker model as "honest-and-*not*-curious" [45].

THEMIS [75] is the Brave browser's contribution to private targeted advertising. It attempts to replace the role of the Broker with a permissioned blockchain, run by Publishers or foundations such as the EFF. THEMIS additionally supports payment to users for their interaction with ads. Privacy for payments, metrics, and auditing of the blockchain is based on a Proof of Authority protocol [1].

The attempted removal of the Broker, auditability, and user compensation are all interesting directions for the future of private advertising. Unfortunately, THEMIS achieves them by entirely offloading both targeting and delivery to clients. Users must download both the targeting model and the *entire* database of ads and ad features to their local device. Fetching fewer ads is not trivial for THEMIS due to the timing information revealed on its blockchain. We do not believe this to be insurmountable, but it would require careful analysis of its security implications which we have not yet seen addressed in Brave's continued development efforts [18].

TURTLEDOVE [100] takes a similar approach to Privad where users subscribe to interest groups and can participate in local auctions to receive an ad based on these interest groups. However, unlike Privad, TURTLEDOVE does not make any assumptions about the specificity of these groups or what metadata may be included in the auction.

Table 9: Comparison of AdVeil to related work on targeted advertising.

	Privacy		**Correctness**			**Scalability**		
	Trusted Third Party (TTP)	Targeting Data Sent to Broker	Targeting Accuracy	Fraud Prevention	Report Granularity	TTP Workload	TTP Availability	Simultaneous Requests
Adnostic [92]	Decryption Oracle	Contextual	Contextual+	Limited [i]	Aggregate	$\ll$ Broker	$\ll$ Broker	✓
ObliviAd [9]	TEE	None[ii]	Full Targeted	Full	Individual	N/A [iii]	N/A	✗
Privad [45]	Dealer	Broad Categories	Limited Targeted	Full	Individual	$\approx$ Broker	= Broker [iv]	✓
THEMIS [75]	PoA Blockchain	None	Full Targeted	Full	Individual	$\approx$ Broker [v]	$\approx$ Broker [v]	✓
TURTLEDOVE [100]	Traditional TTP	None [vi]	Limited Targeted+	N/A	Individual+	< Broker	$\approx$ Broker [iv]	✓
AdVeil	None	None	Full Targeted	Full	Individual	N/A	N/A	✓

[i] Adnostic does not perform fraud detection beyond guaranteeing that an individual report is for a single ad.
[ii] Complete targeting data is sent to the TEE held by the Broker.
[iii] Oblivad's TTP is a TEE held by the Broker. It does not correspond to a physically or administratively separate entity.
[iv] The TTP participates in processing every targeting request, on demand. In Privad it also participates in reporting.
[v] The blockchain authorities are required to validate every report on demand and selected users are required to participate in MPC to compute metrics.
[vi] User data *is* sent to the TTP.

It only requires that group size meets an unspecified k-anonymity threshold. Additionally, TURTLEDOVE requires a trusted server to support the inclusion of real time external data into the local auction. User requests to this server are *not* required to conform to any k-anonymity bound. Between the specificity of interest groups, the blind trust in this server, and the lax restrictions on reporting, it is not clear that TURTLEDOVE provides its users with any more privacy than traditional advertising.

9 DISCUSSION AND CONCLUSIONS

On today's internet, there is a fundamental conflict between privacy and advertising. What we show with AdVeil is that, while the pervasive user tracking performed by ad networks is incompatible with privacy, targeted advertisements are not. AdVeil provides full compliance with technological and legislative best practices for user privacy *without* limiting the relevance of advertisements shown on the internet. Ultimately, we believe that AdVeil is a viable alternative to existing, non-private targeted advertising schemes that meets the needs of both users and ad network brokers.

Targeted Advertising can have serious negative side effects, from web tracking to discriminatory or manipulative practices [37]. In light of this, it seems challenging to advocate for even privacy-preserving targeted advertising. However, current legislative and technological efforts to prevent targeted ads or their ill effects have been markedly unsuccessful. Ad tech companies are more willing to fight lawsuits than they are to stop collecting user data [12, 55, 55, 88] and, despite anti-tracking measures, most peoples' browsers remain uniquely identifiable [20, 38, 67, 77, 105].

In AdVeil users have transparency into the ad targeting process and can avoid targeting on any features they deem sensitive. It does not require ad-tech companies to track users for targeting data, instead giving control over the collection and use of this data back to the users themselves. While AdVeil is not a complete solution to discriminatory or manipulative advertising, we believe it provides a platform for future work in this direction.

Usability motivated our decision to separate AdVeil's targeting and delivery stages. While AdVeil can deliver ads at approximately the same speed as the rest of a page's content, it cannot target them at the same rate. If targeting was performed on-demand, as delivery is, it could result in users navigating away from pages before ads finish loading. Instead, AdVeil targets ads in batches at fixed intervals such that they can be rapidly delivered on demand.

Similarly, the ad network requires that any computational overhead imposed by AdVeil be *sustainable*. While targeting and metrics are more computationally expensive than their non-private alternatives, we showed in Section 7.2.8 that AdVeil remains profitable. Additionally, as was discussed in Section 5.3, both protocols are versatile and allow the ad network its choice of targeting and measurement strategies to align with those used in practice.

Conclusions. We presented AdVeil, a system for privacy-preserving targeted advertising that addresses usability and privacy concerns of existing work. In doing so, we introduce a novel method of targeting ads that provides strong privacy guarantees for users, while ensuring targeting accuracy that is on-par with existing systems. We provide an open-source prototype implementation with an end-to-end evaluation. Our results show the practical and economic feasibility of deploying AdVeil for meeting user privacy demands and regulatory compliance, without compromising on the needs of the ad network.

10 ACKNOWLEDGEMENTS

We thank Peter Deutsch, Jules Drean, and Ben Murphy for for helpful discussion and feedback on this paper. In particular we would like to thank Patrick Zhang for his contributions that lead to early ideas for this work. We also gratefully acknowledge Nirvan Tyagi and Michael Specter for pointing out some subtleties in secure use of ad targeting and tokens respectively.

REFERENCES

[1] Consensys Quorum. https://consensys.net/quorum/. Accessed September 2021.
[2] General Data Protection Regulation. https://gdpr-info.eu/, 2016. Accessed September 2021.
[3] California Consumer Privacy Act of 2018. https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5, 2018. Accessed September 2021.
[4] Shashank Agrawal and Srinivasan Raghuraman. KVaC: Key-value commitments for blockchains and beyond. In *International Conference on the Theory and*

Application of Cryptology and Information Security, pages 839–869. Springer, 2020.

[5] Muhammad Ali, Piotr Sapiezynski, Aleksandra Korolova, Alan Mislove, and Aaron Rieke. Ad delivery algorithms: The hidden arbiters of political messaging. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 13–21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. doi: 10.1145/3437963.3441801. URL https://doi.org/10.1145/3437963.3441801.

[6] Alexandr Andoni, Piotr Indyk, and Ilya Razenshteyn. Approximate nearest neighbor search in high dimensions. *arXiv preprint arXiv:1806.09823*, 7, 2018.

[7] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 962–979. IEEE, 2018.

[8] Gilad Asharov, Ilan Komargodski, Wei-Kai Lin, Kartik Nayak, Enoch Peserico, and Elaine Shi. Optorama: Optimal oblivious RAM. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 403–432. Springer, 2020.

[9] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. Obliviad: Provably secure and practical online behavioral advertising. In *2012 IEEE Symposium on Security and Privacy*, pages 257–271. IEEE, 2012.

[10] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A framework for analyzing anonymous communication protocols. In *2013 IEEE 26th Computer Security Foundations Symposium*, pages 163–178. IEEE, 2013.

[11] Josh Daniel Cohen Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Yale University, 1987.

[12] Stephanie Bodoni. Facebook to fight Belgian ban on tracking users (and even non-users). https://www.bloomberg.com/news/articles/2019-03-27/facebook-attack-of-belgian-order-on-user-tracking-gets-hearing, 2019. Bloomberg.

[13] Dan Boneh, Benedikt Bünz, and Ben Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In *Annual International Cryptology Conference*, pages 561–586. Springer, 2019.

[14] Sanaz Taheri Boshrooyeh, Alptekin Küpçü, and Öznur Özkasap. PPAD: Privacy preserving group-based advertising in online social networks. In *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, pages 1–9. IEEE, 2018.

[15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 337–367. Springer, 2015.

[16] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure: SGX cache attacks are practical. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, Vancouver, BC, August 2017. USENIX Association.

[17] Brave. Brave: The browser reimagined. https://brave.com/, 2020. Accessed September 2021.

[18] Brave. Brave/bat ads and THEMIS request for comments and code (RFC&C) event. https://brave.com/themis-rfcc/, 2021. Accessed September 2021.

[19] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[20] Bill Budington. Panopticlick: Fingerprinting your web presence. San Francisco, CA, January 2016. USENIX Association.

[21] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution. In *27th USENIX Security Symposium (USENIX Security 18)*, page 991–1008, Baltimore, MD, August 2018. USENIX Association. ISBN 978-1-939133-04-5.

[22] José González Cabañas, Ángel Cuevas, and Rubén Cuevas. Unveiling and quantifying Facebook exploitation of sensitive personal data for advertising purposes. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 479–495, Baltimore, MD, August 2018. USENIX Association. ISBN 978-1-939133-04-5.

[23] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *International Workshop on Public Key Cryptography*, pages 55–72. Springer, 2013.

[24] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[25] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 41–50. IEEE, 1995.

[26] Benny Chor, Niv Gilboa, and Moni Naor. *Private information retrieval by keywords*. Citeseer, 1997.

[27] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 719–728, 2017.

[28] Victor Costan and Srinivas Devadas. Intel SGX explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[29] Bennett Cyphers. Google's FLoC is a terrible idea. https://www.eff.org/deeplinks/2021/03/googles-floc-terrible-idea, 2021. Accessed September 2021.

[30] George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *International Workshop on Information Hiding*, pages 293–308. Springer, 2004.

[31] George Danezis, Claudia Diaz, and Carmela Troncoso. Two-sided statistical disclosure attack. In *International Workshop on Privacy Enhancing Technologies*, pages 30–44. Springer, 2007.

[32] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.

[33] Amit Datta, Anupam Datta, Jael Makagon, Deirdre K Mulligan, and Michael Carl Tschantz. Discrimination in online advertising: A multidisciplinary inquiry. *Conference on Fairness, Accountability, and Transparency*, 81:20–34, 2018.

[34] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *Proceedings on Privacy Enhancing Technologies*, 2018(3):164–180, 2018.

[35] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[36] Cynthia Dwork. Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer, 2008.

[37] Gilad Edelman. Why don't we just ban targeted advertising? https://www.wired.com/story/why-dont-we-just-ban-targeted-advertising/, 2020. Accessed September 2021.

[38] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1388–1401, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978313.

[39] Juan Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 648–657. IEEE, 2013.

[40] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[41] Bill Goodwin and Sebastian Klovig Skelton. Facebook's privacy game – how Zuckerberg backtracked on promises to protect personal data. https://www.computerweekly.com/feature/Facebooks-privacy-U-turn-how-Zuckerberg-backtracked-on-promises-to-protect-personal-data, 2019. Accessed September 2021.

[42] Sergey Gorbunov, Leonid Reyzin, Hoeteck Wee, and Zhenfei Zhang. Pointproofs: aggregating proofs for multiple vector commitments. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2007–2023, 2020.

[43] Matthew Green, Watson Ladd, and Ian Miers. A protocol for privately reporting ad impressions at scale. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1591–1601, 2016.

[44] Adam Groce, Jonathan Katz, Aishwarya Thiruvengadam, and Vassilis Zikas. Byzantine agreement with a rational adversary. In *International Colloquium on Automata, Languages, and Programming*, pages 561–572. Springer, 2012.

[45] Saikat Guha, Bin Cheng, and Paul Francis. Privad: Practical privacy in online advertising. In *USENIX conference on Networked systems design and implementation*, pages 169–182, 2011.

[46] Leon J Helsloot, Gamze Tillem, and Zekeriya Erkin. AHEad: privacy-preserving online behavioural advertising using homomorphic encryption. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2017.

[47] Leon J Helsloot, Gamze Tillem, and Zekeriya Erkin. BAdASS: Preserving privacy in behavioural advertising with applied secret sharing. In *International Conference on Provable Security*, pages 397–405. Springer, 2018.

[48] Iab. Iab internet advertising revenue report. https://www.iab.com/news/iab-internet-advertising-revenue/, 2020. Accessed September 2021.

[49] Basileal Imana, Aleksandra Korolova, and John Heidemann. Auditing for discrimination in algorithms delivering job ads. WWW '21, page 3767–3778, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3450077. URL https://doi.org/10.1145/3442381.3450077.

[50] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[51] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Batch codes and their applications. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 262–271, 2004.

[52] Seb Joseph. 'contextual targeting is going to be the new black': As IDFA restrictions loom, advertisers brace for the fallout. https://digiday.com/media/contextual-targeting-is-going-to-be-the-new-black-as-idfa-restrictions-loom-advertisers-brace-for-the-fallout/, 2020. Accessed September 2021.

[53] Ari Juels. Targeted advertising... and privacy too. In *Cryptographers' Track at the RSA Conference*, pages 408–424. Springer, 2001.

[54] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *27th USENIX Security Symposium (USENIX) Security 18)*, pages 1651–1669, 2018.

[55] Michael Kaplan. Facebook and Google are already facing lawsuits under new data rules. https://money.cnn.com/2018/05/25/technology/gdpr-compliance-facebook-google/index.html, 2018. Accessed September 2021.

[56] Kai Kaspar, Sarah Lucia Weber, and Anne-Kathrin Wilbers. Personally relevant online advertisements: Effects of demographic targeting on visual attention and brand evaluation. *PloS one*, 14(2):e0212419, 2019.

[57] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In *Annual International Cryptology Conference*, pages 308–336. Springer, 2020.

[58] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 336–353. IEEE, 2020.

[59] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings 38th annual symposium on foundations of computer science*, pages 364–373. IEEE, 1997.

[60] Ben Laurie. Certificate transparency. *Communications of the ACM*, 57(10): 40–46, 2014.

[61] Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *Theory of Cryptography Conference*, pages 499–517. Springer, 2010.

[62] Robinson & Cole LLP. California's Consumer Privacy Rights Act: Opt-out rights and data profiling. https://www.natlawreview.com/article/california-s-consumer-privacy-rights-act-opt-out-rights-and-data-profiling, 2021. Accessed September 2021.

[63] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. A case study on measuring statistical data in the Tor anonymity network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, LNCS. Springer, January 2010.

[64] Nayantara Mallesh and Matthew Wright. The reverse statistical disclosure attack. In *International Workshop on Information Hiding*, pages 221–234. Springer, 2010.

[65] Match2One. Top banner sizes: The most effective banners of 2020. https://www.match2one.com/blog/standard-banner-sizes/, July 2020. Accessed September 2021.

[66] Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *International Workshop on Privacy Enhancing Technologies*, pages 17–34. Springer, 2004.

[67] Arunesh Mathur, Jessica Vitak, Arvind Narayanan, and Marshini Chetty. Characterizing the use of browser-based blocking extensions to prevent online tracking. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, pages 103–116, Baltimore, MD, August 2018. USENIX Association. ISBN 978-1-939133-10-6.

[68] S. C. Matz, M. Kosinski, G. Nave, and D. J. Stillwell. Psychological targeting as an effective approach to digital mass persuasion. *Proceedings of the National Academy of Sciences*, 114(48):12714–12719, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1710966114.

[69] Advance Media. Digital ad specs. https://www.advancemediany.com/wp-content/uploads/2018/11/DigitalAdSpecs_20181108.pdf, May 2021. Accessed September 2021.

[70] Marcela S Melara, Aaron Blankstein, Joseph Bonneau, Edward W Felten, and Michael J Freedman. CONIKS: Bringing key transparency to end users. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 383–398, 2015.

[71] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[72] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[73] Kate O'Flaherty. This is why people no longer trust Google and Facebook with their data. https://www.forbes.com/sites/kateoflahertyuk/2018/10/10/this-is-why-people-no-longer-trust-google-and-facebook-with-their-data, 2018. Accessed September 2021.

[74] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Annual international cryptology conference*, pages 31–53. Springer, 1992.

[75] Gonçalo Pestana, Iñigo Querejeta-Azurmendi, Panagiotis Papadopoulos, and Benjamin Livshits. THEMIS: Decentralized and trustless ad platform with reporting integrity, 2020.

[76] Angelisa C. Plane, Elissa M. Redmiles, Michelle L. Mazurek, and Michael Carl Tschantz. Exploring user perceptions of discrimination in online targeted advertising. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 935–951, Vancouver, BC, August 2017. USENIX Association. ISBN 978-1-931971-40-9.

[77] Gaston Pugliese, Christian Riess, Freya Gassmann, and Zinaida Benenson. Long-term observation on browser fingerprinting: Users' trackability and perspective. *Proceedings on Privacy Enhancing Technologies*, 2020(2):558 – 577, 01 Apr. 2020. doi: https://doi.org/10.2478/popets-2020-0041.

[78] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[79] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 325–342, 2020.

[80] Phil Schraeder. Contextual advertising leads the way in a post cookie reality. https://adage.com/article/gumgum/contextual-advertising-leads-way-post-cookie-reality/2293976, 2020. Accessed September 2021.

[81] Tjerand Silde and Martin Strand. Anonymous tokens with public metadata and applications to private contact tracing. *IACR Cryptol. ePrint Arch.*, 2021:203, 2021.

[82] Natasha Singer. What you don't know about how Facebook uses your data. https://www.nytimes.com/2018/04/11/technology/facebook-privacy-hearings.html, 2018. Accessed September 2021.

[83] Malcolm Slaney, Yury Lifshits, and Junfeng He. Optimal parameters for locality-sensitive hashing. *Proceedings of the IEEE*, 100(9):2604–2623, 2012.

[84] BY OOKLA SPEEDTEST. Speedtest global index. Technical report, 2021. Accessed September 2021.

[85] Kevin Springborn and Paul Barford. Impression fraud in on-line advertising via pay-per-view networks. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 211–226, 2013.

[86] Statista. Social media advertising cost-per-mille (CPM) worldwide from 2nd quarter 2018 to 4th quarter 2019. https://www.statista.com/statistics/873631/social-media-advertising-cpm/, 2020. Accessed September 2021.

[87] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An extremely simple oblivious RAM protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 299–310, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324779. doi: 10.1145/2508859.2516660.

[88] Jonathan Stempel. Google faces $5 billion lawsuit in U.S. for tracking "private" internet use. https://www.reuters.com/article/us-alphabet-google-privacy-lawsuit/google-faces-5-billion-lawsuit-in-u-s-for-tracking-private-internet-use-idUSKBN23933H, 2020. Accessed September 2021.

[89] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[90] Trefis Team and Great Speculations. Is Google advertising revenue 70%, 80%, or 90% of Alphabet's total revenue? https://www.forbes.com/sites/greatspeculations/2019/12/24/is-google-advertising-revenue-70-80-or-90-of-alphabets-total-revenue/, 2019-12-24. Accessed September 2021.

[91] Alin Tomescu, Yu Xia, and Zachary Newman. Authenticated dictionaries with cross-incremental proof (dis) aggregation. *IACR Cryptol. ePrint Arch.*, 2020: 1239, 2020.

[92] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings Network and Distributed System Symposium*, 2010.

[93] Minh-Dung Tran, Gergely Acs, and Claude Castelluccia. Retargeting without tracking. *arXiv preprint arXiv:1404.4533*, 2014.

[94] Theja Tulabandhula, Shailesh Vaya, and Aritra Dhar. Privacy-preserving targeted advertising. *arXiv preprint arXiv:1710.03275*, 2017.

[95] Nirvan Tyagi, Sofía Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A Wood. A fast and simple partially oblivious prf, with applications.

[96] Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. SGAxe: How SGX fails in practice. https://sgaxe.com/files/SGAxe.pdf, 2020. Accessed September 2021.

[97] Matteo Varvello, Iñigo Querejeta Azurmendi, Antonio Nappa, Panagiotis Papadopoulos, Goncalo Pestana, and Ben Livshits. VPN0: A privacy-preserving decentralized virtual private network. *arXiv preprint arXiv:1910.00159*, 2019.

[98] Frank Wang, Catherine Yun, Shafi Goldwasser, Vinod Vaikuntanathan, and Matei Zaharia. Splinter: Practical private queries on public data. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 299–313, 2017.

[99] Web Incubator CG. FLOC. https://github.com/WICG/floc, 2021. Accessed September 2021.

[100] Web Incubator CG. TURTLEDOVE. https://github.com/WICG/turtledove, 2021. Accessed September 2021.

[101] Miranda Wei, Madison Stamos, Sophie Veys, Nathan Reitinger, Justin Goodman, Margot Herman, Dorota Filipczuk, Ben Weinshel, Michelle L. Mazurek, and Blase Ur. What Twitter knows: Characterizing ad targeting practices, user perceptions, and ad explanations through users' own twitter data. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 145–162. USENIX Association, August 2020. ISBN 978-1-939133-17-5.

[102] John Wilander. Privacy preserving ad click attribution for the web. https://webkit.org/blog/8943/privacy-preserving-ad-click-attribution-for-the-web/, 2019. Accessed September 2021.

[103] David Isaac Wolinsky, Ewa Syta, and Bryan Ford. Hang with your buddies to resist intersection attacks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1153–1166, 2013.

[104] Charles Wright and Mayank Varia. Crypto crumple zones: Enabling limited access without mass surveillance. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 288–306. IEEE, 2018.

[105] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martin Abadi. Host fingerprinting and tracking on the web: Privacy and security implications. In *NDSS*, volume 62, page 66, 2012.

[106] Bassam Zantout, Ramzi Haraty, et al. I2P data communication system. In *Proceedings of ICN*, pages 401–409. Citeseer, 2011.

[107] Ruofei Zhang and Ying Cui. Similarity function in online advertising bid optimization, August 4 2011. US Patent App. 12/698,463.

[108] Xingquan Zhu, Haicheng Tao, Zhiang Wu, Jie Cao, Kristopher Kalish, and Jeremy Kayne. *Fraud prevention in online digital advertising*. Springer, 2017.

A EXTENDED RELATED WORK

FLoC [99] focuses specifically on the targeting phase of the advertising pipeline. It assigns users to groups based on their browsing behavior – users with similar browsing habits and, presumably, interests are assigned to the same group. FLoC users receive ads based on a group identifier rather than a personal feature vector. The proposal believes this to be secure because the number of users per-group is large and the total number of groups is small, indicating that no one group can be too specific to an individual user. Unfortunately, FLoC considers "thousands" of users to be a sufficiently large group size and the group name "43A7" to be a short group identifier. Of course, "thousands" is a *very* small group size compared to the 4.66 billion active internet users and 1,679,616 different groups can be supported, assuming 4 digit alphanumeric representation for group identifiers. Thus, it is unclear what, if anything, sending this group name instead of a feature vector hides about users of FLoC as groups could be highly specific to users' private features.

PPAD [14] is a group-based ad targeting system where the privacy for users is guaranteed relative to the *group* that a user belongs to. Each user is assigned to a group based on their attributes and hence reveals *coarse grained* interests of users. The Broker and a semi-trusted third party run a variant of a private-set intersection protocol (using shares of user-provided feature vectors) to determine which ads should be displayed to a group. The advantage of PPAD is that the Broker can target and deliver ads at a group level and can evaluate set intersection while the user is "offline" in order to serve ads when the user goes back online. However, the model of PPAD provides a tradeoff between targeting accuracy and privacy. Specifically, the more fine-grained the group selection, the more privacy leakage it incurs; PPAD does not provide an analysis of this leakage.

BAdASS [47] (and its precursor AHEAd [46]) are designed for online targeting and do not delve into other aspects of the advertising ecosystem such as reporting and fraud prevention. BAdASS uses a multi-party computation protocol executed between a group of honest-but-curious parties. Moreover, BAdASS requires splitting trust among a set of Demand-Side Platforms (DSPs) which manage content targeting in an ad network. Even a single malicious DSP can disrupt the correctness of the protocol (or worse yet de-anonymize the user) unless expensive zero-knowledge proofs are added to prevent deviations from protocol.

Tran *et al.* [93] is designed to address the *retargeting* aspect of online advertising. Their primary assumption is that the Broker (or ad exchange) will not collude with retargeting services. Clients engage in a protocol between the Broker and retargeter to fetch ads for products they have previously shown interest in (e.g., by adding a product to a shopping cart). The retargeter learns which ad was displayed but not which user requested it, while the Broker learns which user requested the ad but not which ad was retrieved via the retargeter. User privacy is ensured if the Broker and retargeter do not collude.

Tulabandhula *et al.* [94] propose a collection of functions for privacy-preserving association rule mining and recommendations. Their protocols work between a client and server, where the client stores the feature vector locally. Their results can be applied to AdVeil in the targeting process and may be useful for certain deployments.

AdScale [43] improves the reporting scheme proposed in Adnostic, but does not address other aspects of the pipeline. Users in Adscale respond with homomorphically encrypted reports that are aggregated by the Broker, but can only be decrypted by a designated trusted third party (TTP). This is intended to ensure that neither the Broker nor the TTP are individually capable of learning the plaintext responses of a single user. As a consequence, only aggregate information can be used for billing and targeting purposes.

B ANONYMOUS TOKEN CONSTRUCTION

For completeness, we describe the OSPP-without-NIZK construction of one-time-use anonymous tokens. Kreuter et al. [57] do not explicitly describe the construction that we present below, however, we note that it is a direct corollary of two constructions they do present (and is explicitly alluded to in their paper).

Motivation. In AdVeil, we require that only the verifier (i.e., the Broker) can generate a valid token, while anyone can generate an invalid token. Crucially, we must also guarantee that recipients of tokens *cannot tell whether the token is valid or invalid.* This subtle property is required to avoid alerting bots that they have been detected by the Broker.

The challenge is that anonymous token constructions [34, 95] *explicitly* reveal token validity to the prover (specifically, in [34, 95] the verifier provides a NIZK proof of correct token signing). Kreuter et al. [57] provide a construction for an anonymous token with a *private* metadata bit (only readable to the verifier) and does not require a NIZK of correct signing. While this does allow the Broker to identify and exclude "bot" reports, it does so using a metadata bit $b \in \{0, 1\}$ included in *valid* tokens. That is, there are *three* types of tokens, and thus three possible ways to divide clients, under this construction:

(1) valid token signatures with $b = 0$,
(2) valid token signatures with $b = 1$ and,
(3) invalid token signatures.

```
Setup(1^λ)
1: return crs := $(\mathbb{G}, g, h)$
```

```
KeyGen(crs)
1: parse crs = $(\mathbb{G}, g, h)$
2: $(x, y) \leftarrow \mathbb{Z}_p^* \times \mathbb{Z}_p^*$
3: pk := $(\mathbb{G}, g, h, g^x.h^y)$
4: sk := $(x, y)$
5: return (pk, sk)
```

```
TokenGen(pk)
1: parse pk = $(\mathbb{G}, g, h, g^x)$
2: $\tau \leftarrow \{0,1\}^\lambda$
3: $t \leftarrow H(\tau)$
4: $\alpha, \beta \xleftarrow{R} \mathbb{Z}_p^*$
5: $\not\tau \leftarrow (tg^{-\beta})^{\alpha^{-1}}$
6: $r := (\alpha, \beta)$
7: return $(\tau, \not\tau, r)$
```

```
Sign(sk, $\not\tau$)
1: parse sk = $(x, y)$
2: $z \xleftarrow{R} \mathbb{Z}_p^*$
3: $s \leftarrow H_z(\not\tau, z)$
4: $w \leftarrow \not\tau^x s^y$
5: return $\not\sigma = (z, w)$
```

```
Unblind(pk, $\not\tau$, $\not\sigma$, r)
1: parse pk = $(\mathbb{G}, g, h, g^x, h^y)$
2: parse $r = (\alpha, \beta)$
3: parse $\not\sigma = (z, w)$
4: $u \leftarrow H_z(\not\tau, z)^\alpha h^\beta$
5: $v \leftarrow w^\alpha (g^x h^y)^\beta$
6: return $\sigma = (u, v)$
```

```
Redeem(sk, $\ell$, $\tau$, $\sigma$)
1: parse sk = $(x, y)$
2: parse $\sigma = (u, v)$
3: $t \leftarrow H_\tau(\tau)$
4: $w \leftarrow t^x u^y$
5: if $w = v$ and $\tau \notin \ell$ then
6:     $\ell' \leftarrow \ell \cup \tau$
7:     return (valid, $\ell'$)
8: else return (invalid, $\ell$)
```

Figure 10: OSPP-without-NIZK anonymous one-time-use token construction.

This allows a malicious Broker to target a specific user by issuing only a single token with $b = 1$ while still excluding fraudulent requests using invalid tokens. In this way, the Broker is able to link the identity of a targeted user to their report contents without compromising on the accuracy of either metrics or fraud prevention.

To prevent this attack, we require a construction with at most *two* sets: clients with valid tokens and clients with invalid tokens. We observe that an implicit construction alluded to but not described by Kreuter et al. [57] gives us the required properties:

(1) any party can generate invalid token signatures,
(2) only the Broker can generate valid token signatures,
(3) clients do not learn the validity of their token signature.

We call the construction OSPP-without-NIZK, following the convention laid down by Kreuter et al. [57]. For simplicity, we present the OSPP-without-NIZK construction *without* embedded public metadata but hypothesize that the construction is extendable to that regime following the transformation of Silde and Strand [81]. We present the construction in Figure 10 and note the resemblance to the OSPP [57, Construction 2] and PMBT-without-NIZK [57, Construction 5]. Let $\mathbb{G}$ be any prime order group of order $p > 2^\lambda$ with generators g and h. For security, we assume that the Chosen-target Diffie–Hellman assumption holds in $\mathbb{G}$ [57].

Correctness and security of OSPP-without-NIZK. To see correctness, observe that

$$\begin{aligned}
w &= t^x u^y \\
&= t^x (H_z(\not\tau, z)^\alpha h^\beta)^y \\
&= t^x H_z(\not\tau, z)^{\alpha y} h^{\beta y} \\
&= t^x g^{-\beta x} H_z(\not\tau, z)^{\alpha y} g^{\beta x} h^{\beta y} \\
&= t^x g^{-\beta x} H_z(\not\tau, z)^{\alpha y} (g^x h^y)^\beta \\
&= (t^{\alpha^{-1} x} g^{-\beta \alpha^{-1} x} H_z(\not\tau, z)^y)^\alpha (g^x h^y)^\beta \\
&= ((t^{\alpha^{-1}} g^{-\beta\alpha^{-1}})^x H_z(\not\tau, z)^y)^\alpha (g^x h^y)^\beta \\
&= (((tg^{-\beta})^{\alpha^{-1}})^x H_z(\not\tau, z)^y)^\alpha (g^x h^y)^\beta \\
&= (\not\tau^x H_z(\not\tau, z)^y)^\alpha (g^x h^y)^\beta \\
&= w^\alpha (g^x h^y)^\beta \\
&= v
\end{aligned}$$

For security, we point to the definitions and proofs of Kreuter et al. [57]. The unlinkability (specifically 2-unlinkability [57, Definition 2]), unforgeability, and privacy of the token validity [57, Theorem 15] follow from the security analysis OSPP and PMBT-without-NIZK constructions of Kreuter et al. [57].

Application of OSPP-without-NIZK to AdVeil. Following the one-time setup and key generation, a client runs TokenGen to generate a new token with a targeting or delivery request. The blind token is sent to the Broker for signing. If the Broker believes the client to be a "human", then it generates a valid signature using Unblind. Otherwise, the Broker returns $\sigma = (z, w)$ with random $z, w \xleftarrow{R} \mathbb{Z}_p^*$ (invalid signature). The client cannot differentiate between a valid and invalid signature [Theorem 15][57]. The client unblinds the signature using Unblind. The unblided (and unlikable) token and signature is given to the Broker in reporting. The Broker rejects all reports where Redeem outputs invalid. As mentioned earlier, the Broker can only partition clients based on the token validity, partitioning all clients into at most two anonymity sets.

C PRIVATE KEYWORD ADVERTISING

While we present a protocol for privacy-preserving ad selection using nearest neighbor search (e.g., for selecting ads based on user interests), a different category of advertising involves displaying ads to users based on *keywords*. For example, searching for "hotels in madagascar" in DuckDuckGo displays an ad by `booking.com` for hotel deals in Madagascar as well as an ad for cheap plane tickets. These ads are served by matching keywords in the search query ("hotel" and "madagascar") to a "bag of words" associated with each ad. Our protocol for privacy preserving nearest neighbor search can easily be adapted to match a set of keywords.