BAdASS: Preserving Privacy in Behavioural Advertising with Applied Secret Sharing (Extended Version)[⋆]

Leon J. Helsloot, Gamze Tillem, and Zekeriya Erkin

Delft University of Technology, Delft, The Netherlands
leonhelsloot@gmail.com, {g.tillem,z.erkin}@tudelft.nl

Abstract Online advertising is a multi-billion dollar industry, forming the primary source of income for many publishers offering free web content. Serving advertisements tailored to users' interests greatly improves the effectiveness of advertisements, which benefits both publishers and users. The privacy of users, however, is threatened by the widespread collection of data that is required for behavioural advertising. In this paper, we present BAdASS, a novel privacy-preserving protocol for Online Behavioural Advertising that achieves significant performance improvements over the state of the art without disclosing any information about user interests to any party. BAdASS ensures user privacy by processing data within the secret-shared domain, using the heavily fragmented shape of the online advertising landscape to its advantage and combining efficient secret-sharing techniques with a machine learning method commonly encountered in existing advertising systems. Our protocol serves advertisements within a fraction of a second, based on highly detailed user profiles and widely used machine learning methods.

Keywords: Behavioural advertising · Machine learning · Secret sharing · Privacy · Cryptography.

1 Introduction

Online advertising is forming one of the driving economic factors behind free web services. With a global spend of $178 billion in 2016 [18], online advertising forms a primary financial pillar supporting free web content by allowing publishers to offer content to users free of charge [9]. In recent years, however, an increasing number of people object to advertisements being shown on web pages they visit, resulting in a sharp increase in the use of technological measures to block advertisements. According to a 2017 report, an estimated 615 million devices have ad blocking tools installed, amounting to 11% of the Internet population, and the use of such tools is expected to grow further in the future [22]. The consequence of the increased use of ad blockers is that publishers experience a

[⋆] This paper is an extended version of a paper that will appear in the proceedings of the 12th International Conference on Provable Security (ProvSec 2018).

significant loss of revenue from the advertising space they offer. The global loss of advertising revenue due to ad blocking was estimated to be $41.4 billion in 2016, or 23% of the total ad spend [23]. Some publishers request that users disable ad blockers on their web pages, or deny access to ad blocker users altogether, in an effort to limit revenue loss due to ad blocking [1]. The consequence of such practices is an arms race between ad blocking technologies, and circumvention of ad blockers. These developments pose a threat to the business models of many free web services, necessitating measures to alleviate the objections against online advertising in order to attain a sustainable advertisement-supported Internet economy.

A major concern for the users is their privacy which is threatened by the widespread data collection of advertising companies [30]. The collected data is used in behavioural targeting to determine which advertisements are shown to a user based on the user's browsing behaviour. Although such behavioural advertising is recognized as being beneficial to both users and publishers, a mistrust of advertising companies and a lack of control hinders acceptance of behavioural advertising [30]. In a recent survey among users of an ad blocking tool, 32% of respondents indicated that privacy concerns were among the reasons for their use of an ad blocker [1]. A similar survey on privacy and advertising showed that 94% of respondents considered online privacy an important issue, and 70% of respondents indicated that online advertising networks and online advertisers should be responsible for users' online privacy [29].

The practice of showing advertisements based on previously exhibited behaviour is known as Online Behavioural Advertising (OBA). In OBA, user interests are inferred from data such as visited web pages, search queries, and online purchases. Based on these user interests, advertisements are typically personalized using campaign-specific supervised machine learning models that predict users' responses to advertisements. Behavioural advertising greatly improves the expected advertising effectiveness by targeting advertisements at individual users [33]. Users benefit from OBA by being served more relevant advertisements, and advertisers can reach a specific desired audience by using accurate targeting. Moreover, publishers benefit from an increased value of their advertising space.

OBA utilises the Real-Time Bidding (RTB) model of buying and selling advertisements [31]. RTB facilitates real-time auctions of advertising space through marketplaces called ad exchanges (AdX), allowing buyers to determine bid values for individual ad impressions. The real-time nature of such auctions, in which bids are to be placed in a fraction of a second, allows fine-grained control over allocation of advertising budgets, but also requires the whole auction process to be carried out programmatically. Along with ad exchanges, other types of platforms emerged to manage the added complexity of RTB. Demand-Side Platforms (DSPs) provide advertisers, who may not possess the expertise required to accurately estimate impression values, with technologies to bid on individual impressions from multiple inventories. Likewise, Supply-Side Platforms (SSPs) support publishers in optimizing advertising yield.

In existing literature, a number of methods is proposed to address privacy concerns in OBA. These methods include blocking advertisements altogether [20], obfuscating browsing behaviour [8], and anonymization [24], as well as exposing only generalized user profiles to advertising companies [28]. Limiting the data that is available to advertising companies, however, is expected to decrease the targeting accuracy [10], and thus the value of advertisements to users, advertisers, and publishers. Other work proposes cryptographic approaches to aggregate click statistics [28] or select advertisements using secure hardware [3]. These approaches, however, are based on advertising models in which centralized networks perform simple keyword-based advertisement selection, and as such are unsuitable for use within the highly distributed RTB model. Recently, Helsloot et al. [17] proposed a protocol that uses threshold homomorphic encryption to preserve privacy in OBA within the RTB model. However, the use of expensive cryptographic operations throughout the protocol results in prohibitively large computational costs.

In this paper, we present BAdASS, a novel privacy-preserving protocol for OBA that is compatible with the RTB mechanism of buying ads and supports behavioural targeting based on highly detailed user profiles. BAdASS achieves significant performance improvements over the state of the art, using machine learning on secret-shared data to preserve privacy in OBA tasks. Our protocol uses the highly fragmented nature of the OBA landscape to distribute trust between parties, such that no single party can obtain sensitive information. We achieve performance multilinear in the size of user profiles and the number of DSPs, and perform the highly time-sensitive advertisement selection task in a fraction of a second.

In the rest of the paper, we summarize the preliminary methods in Section 2. In Section 3 we explain BAdASS in detail. In Section 4 we provide the performance analyses for our protocol based on communication and computation complexity and real-time experiments. We analyze the security of BAdASS in Section 5 and, we conclude the paper in Section 6.

2 Preliminaries

2.1 Logistic Regression

Logistic regression is one possible technique for user response estimation which has been commonly used by advertising companies such as Google [19], Facebook [16], and Criteo [5]. Given a d-dimensional feature vector $\boldsymbol{x}$ and model parameters $\boldsymbol{w}$, it estimates the probability of a binary outcome using the sigmoid function:

$$\hat{y} = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{w}^\top \boldsymbol{x}}}. \tag{1}$$

In the concept of OBA, $\boldsymbol{x}$ contains the user profile while the binary output y indicates click or no click. The model parameters are updated as $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \boldsymbol{g}$ using the gradient of the logistic loss $\boldsymbol{g} = (\hat{y} - y)\boldsymbol{x}$ as in [19]. η in the update function is the learning rate which can be a constant or can be set to decay per iteration and per coordinate of the gradient vector.

2.2 Feature Hashing

The feature space in logistic regression may become too large when categorical features with high cardinality are used. To avoid a high dimensional user vector, we use the hashing trick in [32] which enables to map the user profile into a lower-dimensional vector $\boldsymbol{x}$ by setting x_i to a count of the values whose hash is i. The resulting d-dimensional vector $\boldsymbol{x}$ (in [5], $d = 2^{24}$ is used) is the input feature vector to the logistic regression model.

2.3 Shamir Secret Sharing

Shamir's secret sharing scheme [26] is a t-out-of-n threshold scheme in which a secret $s \in \mathbb{Z}_p$ for a prime p is shared among n parties, from which any subset of size at least t can reconstruct the secret. We use the notation $\langle s \rangle$ to indicate a (t, n) Shamir secret sharing of a value s, for some predefined t and n, and $\langle \boldsymbol{v} \rangle$ denotes an element-wise Shamir sharing of the vector $\boldsymbol{v}$. Shamir's secret sharing scheme is additively homomorphic, such that $\langle s_1 \rangle + \langle s_2 \rangle = \langle s_1 + s_2 \rangle$. Parties holding shares of two secret values can thus compute shares of the sum of the two values, without interaction with other parties. Furthermore, a public value c can be added to a shared secret s without interaction by adding c to each of the shares, i. e. $\langle s \rangle + c = \langle s + c \rangle$. Likewise, a shared secret can be multiplied with a public value c by multiplying each of the shares with c, i. e. $\langle s \rangle \cdot c = \langle cs \rangle$.

Multiplication of the shares of two secret values s_1 and s_2 results in a $\big(2\,(t-1)\,, n\big)$ sharing of $s_1 \cdot s_2$, thus requiring $2t - 1$ shares to reconstruct the secret. Ben-Or et al. [4] describe a multiplication protocol in which the resulting polynomial is reduced to degree $t - 1$ and randomized. Given a sharing of a value s, where $\langle s \rangle_i$ is held by party i, the degree reduction step is performed by each party splitting their share $\langle s \rangle_i$ into a new (t, n) sharing $\langle s \rangle_{i,1}, \ldots, \langle s \rangle_{i,n}$. Each party i distributes their subshares among all parties, such that party j is given the subshare $\langle s \rangle_{i,j}$. Each party j then combines the received subshares $\langle s \rangle_{1,j}, \ldots, \langle s \rangle_{n,j}$ into a new share $\langle s \rangle'_j$. The resulting sharing $\langle s \rangle'$ is a new (t, n) sharing of the value s. Gennaro et al. [14] simplify the degree reduction and randomization steps into a single step, requiring a single round per multiplication. Note that n needs to be at least $2t - 1$ for degree reduction to work.

2.4 Universal Re-encryption

Universal re-encryption, presented as a technique for mix networks by Golle et al. [15], allows re-randomization of a ciphertext without access to the public key that was used during encryption. In BAdASS, we use the universal re-encryption technique presented in [15], based on a multiplicatively homomorphic cryptosystem such as ElGamal [12]. We use the notation $[\![x]\!]_u$ to denote the encryption of a value x under the public key of user u using a universal re-encryption scheme.

3 Protocol Design

In designing a privacy-preserving online advertising system, we aim to satisfy three goals. The first goal is to ensure profile privacy, such that information from which the interests of a user can be inferred is not revealed to any party other than the user. Moreover, we aim to ensure model privacy, such that model parameters used by a bidder are not revealed to any party other than the bidder. Finally, the system must be applicable to the RTB model and integrated into the online advertising landscape, allowing bidders to estimate the value of an ad impression based on historic observations.

Our protocol is based on a semi-honest security model. Since some existing companies act as both AdX and DSP, we assume that the AdX colludes with DSPs. To assist in privacy-preserving computations in the presence of colluding parties, we introduce an additional entity into our setting called Privacy Service Provider (PSP). The PSP is not trusted with private data in unencrypted form, but is assumed to follow the protocol specification without colluding with any other party. We assume that targeting is performed only by DSPs, such that DSPs are the only parties that operate on user data. The AdX only collect bids, and from these bids select the winner. SSPs only offer advertising space to multiple ad exchanges, and are not considered in our protocol.

Parties collaboratively compute bid values based on a logistic regression model using secret-shared user profiles and model parameters. We define a DSP group $\mathit{\Gamma}_i$ to be a set of DSPs of size at least $m = 2t - 1$ for a given reconstruction threshold t. Computations on behalf of a DSP $\gamma_{i,j} \in \mathit{\Gamma}_i$ are performed entirely within $\mathit{\Gamma}_i$. In our protocol descriptions, any operations performed by DSPs on secret-shared values are assumed to be performed by all DSPs in a DSP group $\mathit{\Gamma}_i$. Plaintext values and encrypted values are generated by the DSP responsible for the campaign on which computations are performed and, where necessary, published within $\mathit{\Gamma}_i$.

BAdASS is divided into four different phases: user profiling, bidding, auction, and model update. Prior to protocol execution, advertisers set up campaigns such that DSPs can bid on their behalf, and the PSP splits DSPs into groups of at least m parties. Moreover, each DSP shares campaign-specific parameters among the DSPs in their group. Finally, each user generates a key pair using any multiplicatively homomorphic asymmetric cryptosystem and publishes their public key. In the following subsections, we explain the four phases of BAdASS. A summary of symbols used in the description of the protocol design is provided in Table 4 in Appendix A.

3.1 User Profiling Phase

To preserve privacy during the user profiling phase, browsing behaviour is recorded locally within the user's web browser using an existing technique such as RePriv [11]. The resulting profile is captured in a d-dimensional feature vector $\boldsymbol{x}$ using feature hashing. To reduce the communication costs associated with sending the full d-dimensional feature vector for each request, feature vectors are cached at DSPs. Caching allows periodic background updates of feature vectors, minimizing delays experienced by the user during the time-sensitive advertisement selection phase. To securely share a feature vector among DSPs without knowing the topology of the OBA landscape, the user splits their profile into two additive shares, one of which is given to the ad exchange, the other to the PSP. Both the ad exchange and the PSP, in turn, create Shamir shares from their additive shares, which are distributed among the DSP groups for which the profile update is intended. Every DSP within the group thus receives two Shamir shares, one from each additive share created by the user, which are combined into a single Shamir share of the original value by calculating the sum of the two shares.

Depending on the feature hashing method used, the feature vector $\boldsymbol{x}$ is either binary or contains only small values. Assuming the use of a binary feature vector, it would be ideal from the user's perspective to create additive shares of the feature vector in $\mathbb{Z}_2$, rather than $\mathbb{Z}_p$, as smaller shares reduce the required communication bandwidth. Subsequent computations on the user profile, however, must be performed in $\mathbb{Z}_p$ to represent real values with sufficient precision. In our setting with two additive shares, securely converting shares in $\mathbb{Z}_2$ to shares in $\mathbb{Z}_p$ requires an invocation of the multiplication protocol for every feature vector dimension, resulting in high computation and communication costs for DSPs. We therefore favour sharing the user profile in $\mathbb{Z}_p$.

3.2 Bidding Phase

The bidding phase starts when a users contacts an AdX with an ad request. Receiving the ad request, AdX sends a bid request to DSP groups each of which cooperatively calculates the bidding prices for the campaigns they are responsible for. For each campaign, the user response $\hat{y}$ is estimated using a logistic regression model, and bidding values are derived from response estimations using linear bidding functions $B(\hat{y}) = c_1\hat{y} + c_2$ for campaign-specific constants c_1 and c_2. A challenge in logistic regression is to compute sigmoid function within the secret-shared domain. In existing literature the sigmoid function is computed either by approximation [34,6] or in clear [21,2,17]. In this work, we let the PSP compute the sigmoid function in the clear, as approximation leads to a degradation of predictive performance and incurs additional computational costs. The input to the sigmoid function $\boldsymbol{w}^\top\boldsymbol{x}$ is thus revealed to the PSP. In our setting, this is acceptable as the PSP knows neither the user, nor the campaign a value is associated with. Therefore, the PSP cannot infer any more information than that there exists a user who is interested in a topic. Moreover, a DSP group could submit additional randomly generated values to mask real inputs.

Protocol 1 outlines the bidding protocol. The model parameters $\boldsymbol{w}_k$ for campaigns $k \in K_{\Gamma_i}$, where K_{Γ_i} is the set of campaigns run within Γ_i, and the user profile $\boldsymbol{x}_u$ of a user u, are shared within Γ_i. The multiplications that are required for the calculation of the inner product of $\boldsymbol{w}_k$ and $\boldsymbol{x}_u$ in line 3 are performed locally, without degree reduction. Since the results of these local multiplications are not used in further multiplications, the sum of all multiplied values is a single sharing $\langle s_k \rangle$ of degree $2t-2$. As the PSP subsequently collects and combines all $m \geq 2t-1$ shares of s_k, no degree reduction step is required in calculating $\boldsymbol{w}^\top \boldsymbol{x}$.

Protocol 1 Bidding protocol, executed jointly by a DSP group Γ_i and the PSP, and invoked by the ad exchange for a user u at every DSP group.

```
1: procedure DSP:CALCULATE-BID($\{\langle \boldsymbol{w_k} \rangle, \langle \boldsymbol{c}_k \rangle \mid k \in K_{\Gamma_i}\}, u$)
2:     for all $k \in K_{\Gamma_i}$ do
3:         $\langle s_k \rangle \leftarrow \sum_{i=1}^{d} \langle x_{u,i} \rangle \cdot \langle w_{k,i} \rangle$
4:         Pick a unique random value $r_k$
5:         Store mapping $r_k \rightarrow ([\![a_k]\!]_u, \Gamma_i)$ at PSP via AdX
6:     end for
7:     Pick random permutation function $\pi(\cdot)$
8:     $\langle \boldsymbol{s'} \rangle \leftarrow \pi(\langle \boldsymbol{s} \rangle)$
9:     invoke $\langle \hat{\boldsymbol{y}}' \rangle \leftarrow$ PSP:CALCULATE-SIGMA($\langle \boldsymbol{s'} \rangle$) at PSP
10:    $\langle \hat{\boldsymbol{y}} \rangle \leftarrow \pi^{-1}(\langle \hat{\boldsymbol{y}}' \rangle)$
11:    for all $k \in K_{\Gamma_i}$ do
12:        $\langle b_k \rangle \leftarrow \langle c_{k,1} \rangle \cdot \langle \hat{y}_k \rangle + \langle c_{k,2} \rangle$
13:    end for
14: end procedure

15: procedure PSP:CALCULATE-SIGMA($\langle \boldsymbol{s} \rangle$)
16:     $\boldsymbol{s} \leftarrow$ combine $\langle \boldsymbol{s} \rangle$
17:     for all $s_i \in \boldsymbol{s}$ do
18:         $\hat{y}_i \leftarrow \sigma(s_i)$
19:     end for
20:     return $\langle \hat{\boldsymbol{y}} \rangle$
21: end procedure
```

Since campaign parameters $c_{k,1}$ and $c_{k,2}$ are private to the DSP responsible for campaign k, they are secret-shared among the parties in the DSP group. Calculation of the bid price $b_k = c_{k,1}\hat{y}_k + c_{k,2}$ therefore requires a single invocation of the multiplication protocol for every campaign, which can be parallelized such that all bid values are calculated in a single round of communication. To ensure profile privacy, each advertisement a_k is encrypted using the user's public key. The encrypted advertisement is submitted to the PSP, via the AdX such that the PSP cannot link the submission to a specific DSP, along with a random number r_k and the group descriptor Γ_i. Finally, the PSP stores a mapping $r_k \rightarrow ([\![a_k]\!]_u, \Gamma_i)$, which is used in the auction phase to retrieve the advertisement.

3.3 Auction Phase

The auction protocol uses a hierarchical auction in which each DSP group engages in a secure comparison protocol to select the highest of the bids within the DSP group, along with associated information that is used in the model update phase.

Shares of the information associated with the highest bid are stored for later use, after which each DSP group submits their highest bid to a global auction to select the final winner. Note that, due to the use of secret sharing, the global auction cannot be performed by the ad exchange alone. In order to maintain the same level of trust as in the bidding protocol, at least m parties are required in the auction protocol. Therefore, the global auction is not performed by the ad exchange, but by a randomly selected DSP group Γ^*.

The auction protocol is shown in Protocol 3 in Appendix B. The protocol relies on a secure comparison protocol that takes as input shares of two values a and b, and gives as output shares of 1 if $a \geq b$, and shares of 0 otherwise. Such a protocol is described by e. g. Reistad and Toft [25]. During the procedure to find the maximum bid, shares of the highest bid and additional information associated with the highest bid are obtained via multiplication with the result of the comparison. After the global comparison, shares of a random identifier r associated with the highest bid are sent to the PSP, where the shares are combined to retrieve the encrypted advertisement and group descriptor associated with the highest bid. To ensure unlinkability between the encrypted advertisement retrieved from the PSP after the auction and the values submitted prior to the auction, the PSP performs re-randomization of the encrypted advertisement on line 30 using universal re-encryption. Finally, the encrypted ad and the group descriptor, as well as the bid request identifier v, are sent via the ad exchange to the user, who decrypts and displays the advertisement.

3.4 Model Update Phase

During the model update phase, the response prediction model associated with the shown advertisement is updated using the update rule from Section 2.1. In order to ensure unlinkability between users and campaigns, the model update protocol is split into three stages. During the first stage, the user identifier is revealed to the DSP group responsible for the shown advertisement in order to calculate shares of the update gradient $\boldsymbol{g} = \eta(\hat{y} - y)\boldsymbol{x}$. In the second stage, each DSP submits a set of multiple gradient shares to the PSP, which mixes the received shares via random rotation. The PSP then re-shares the set of gradient shares among the DSP group. In the final stage, the campaign identifiers of the set of gradients are revealed to the DSP group, allowing the DSP group to apply the gradients calculated in the first stage to the correct parameter vector. Since the gradient shares have been mixed, the DSP group cannot link values revealed in the third phase to values revealed in the first phase.

The model update protocol, described in detail in Protocol 4 in Appendix B, is initiated by a user u, who reports shares of their response y directly to the responsible DSP group based on the group descriptor Γ_i received along with the advertisement. In the first phase, each DSP in Γ_i calculates shares of $\delta = \eta(\hat{y} - y)$, which is multiplied with each element of the user profile to form $\boldsymbol{g}$. These multiplications are performed locally, without reducing the degree of the result. The update gradient shares, bid value shares, and campaign identifier shares are locally stored as lists $\langle G \rangle$, $\langle B \rangle$, and $\langle K \rangle$ until sufficient values are aggregated for mixing. When sufficient values are accumulated, each DSP sends its shares to the PSP for mixing. To prevent recombination of shares by the PSP, each DSP $\gamma_{i,j}$ rotates their lists of shares by a random number $r_{i,j}$. Given a sufficiently large aggregation threshold, the average number of attempts needed for the PSP to successfully combine the received shares becomes prohibitively large. The PSP subsequently picks a random number r_{PSP}, and rotates each of the lists of shares r_{PSP} times, such that the positions of output values cannot be linked to the positions of input values.

The share values themselves need also be randomized before being sent back to the DSPs to prevent linking input values to output values. Since the PSP does not know which received shares belong to the same value due to the rotation by DSPs, randomization cannot be performed by adding shares of zero. Instead, the PSP splits each received share $\langle s \rangle_i$ into a new sharing $\langle s \rangle_{i,1}, \ldots, \langle s \rangle_{i,n}$. These subshares are distributed among the DSPs in Γ_i in a manner analogous to the degree reduction step described in Section 2.3, such that each party j receives subshares $\langle s \rangle_{1,j}, \ldots, \langle s \rangle_{n,j}$. The DSPs then recombine the received subshares to obtain new shares of the original values $\langle s \rangle_i$. To match subshares originating from the same value, each DSP rotates the subshares originating from DSP $\gamma_{i,j}$ back $r_{i,j}$ times. After recombining the rotated subshares, each DSP has lists $\langle G' \rangle$, $\langle B' \rangle$, and $\langle K' \rangle$, where each of the lists contains shares of the same values as were submitted to the PSP, rotated r_{PSP} times.

The DSPs combine $\langle K' \rangle$ to reveal the list of campaign identifiers to which the gradient and bid shares belong. Due to the mixing of the lists, DSPs cannot link the campaign identifiers revealed in the third phase to user identifiers revealed in the first phase, provided a sufficiently large mixing threshold is chosen. Finally, each DSP locally updates their shares of the model parameter vectors with the list of gradient shares, and adds the received bid shares to the bid value aggregates.

4 Performance Analysis

To evaluate the performance of BAdASS, we provide both a theoretical analysis of the computational and communication complexities of the subprotocols, and a set of measurements obtained from a proof-of-concept implementation.

4.1 Computational Complexity

The computational complexity of BAdASS depends on a number of variables, in particular the user profile dimensionality d, the number of campaigns K, and the update aggregation threshold ζ. The used variables are summarized in Table 5 in Appendix A. In the profile update protocol, the user creates d additive sharings, and the ad exchange and PSP both create d Shamir sharings. Moreover, each DSP performs d additions. If the profile update protocol is invoked for all DSP groups at once, the computational complexity is therefore $O(dn)$, where n is the total number of DSPs. In the bidding protocol, each DSP performs a multiplication for every campaign within its DSP group to calculate the bid value, and an encryption of the advertisement for all its own campaigns. Calculation of shares of the inner product $\langle \boldsymbol{w}^{\top}\boldsymbol{x}\rangle$ is performed locally, as explained in Section 3.2, and since the resulting value is reconstructed by the PSP, no degree reduction step is necessary. The multiplications involved in calculating the inner product are thus 'free'. In the auction protocol, each DSP group Γ_i performs $K_i - 1$ comparisons, where K_i is the number of campaigns of Γ_i, followed by a single DSP group Γ^* performing $g - 1$ comparisons, where g is the number of groups. Since the group Γ^* is chosen at random out of g groups for every auction, the amortized complexity of the the auction phase is $O(K)$. In the model update protocol, the d multiplications to compute the update gradient need no degree reduction step because the shares are mixed at the PSP, and are thus 'free'. The mixing step is performed for a batch of ζ updates once every ζ invocations, resulting in an amortized cost equal to that of processing a single update. To process a single update, the PSP re-shares all m shares of the d-dimensional update gradients, where m is the size of DSP groups, followed by the DSP performing d local recombinations of the created subshares. The total cost of the re-sharing, in terms of sharing and reconstructing secrets, is equal to that of dm multiplications. The group size m, however, can be considered a constant determined by the recombination threshold, resulting in an amortized complexity of $O(d)$ for the model update phase.

4.2 Communication Complexity

Table 1 lists the amortized number of bits transmitted by each party for each subprotocol, as well as the number of rounds of communication required by each subprotocol. The round complexities of the profile update and bidding protocols are constant. Since the group size m is bounded by a constant, and the share size σ is constant, the communication complexity of the profile update phase can be considered linear with respect to the profile size d. Note that if the user profile is distributed among multiple DSP groups, the complexity of the user profiling phase becomes multilinear in the profile size and the number of DSPs. During the bidding phase, the ad exchange acts as a proxy to transmit a total of K advertisement tuples, and the PSP transmits K sharings of the estimated user response. Each DSP performs K_i multiplications, each requiring m shares to be transmitted, sends K_i shares of inner products to the PSP, and sends an advertisement tuple for each of its κ own campaigns to the PSP via the ad exchange. The total communication complexity of the bidding phase is thus $O(K)$, or linear in the number of campaigns.

The auction protocol consists of λ comparisons, where λ is logarithmic with respect to the number of campaigns within a group and the number of groups.

The round complexity of the auction phase is thus $O(\log_2 K)$, or logarithmic in the number of campaigns, provided the round complexity of the comparison protocol is constant. Since each DSP performs an average of K_i comparisons per invocation of the auction protocol and transmits a fixed number of shares for each multiplication, the communication complexity of the auction protocol is

Table 1. Communication bandwidth in bits and number of rounds of communication per invocation of each subprotocol of BAdASS. ϵ denotes the size of a ciphertext, and γ and τ the number of bits transferred in the comparison and truncation protocols. ρ is the round complexity of the comparison protocol, and T is the round complexity of the truncation protocol.

Protocol	Rounds	User	AdX	DSP	PSP
Profiling	2	$2d\sigma$	$dm\sigma$		$dm\sigma$
Bidding	2		$K\xi$	$(m+1)\,K_i\sigma + \kappa\xi$	$Km\sigma$
Auction	$\lambda(\rho+1)+3$		ξ	$(5K_i-3)\,m\sigma + K_i\gamma + 2\frac{1}{g}\sigma$	ξ
Update	$T+1\frac{3}{\zeta}$	$m\sigma$		$(d+1)m\sigma + \tau + (d+3)\sigma$	$(d+2)m^2\sigma$

linear in the number of campaigns, provided the communication complexity of the comparison protocol is constant.

The model update protocol contains a single invocation of the truncation protocol, of which the number of rounds is considered constant, as well as one round of multiplication. Every ζ invocations, three more rounds for mixing and combining are performed. The amortized round complexity of the model update protocol is therefore constant. Although the amount of bits transmitted by the PSP contains a factor m^2, we can assume this to be a small constant due to the small upper bound on m. The average communication complexity of the is $O(d)$, provided the communication complexity of the truncation protocol is constant and the group size m is bounded by a constant.

4.3 Implementation

To measure the runtime of BAdASS, we made a proof-of-concept implementation of the protocol in C++. The secure comparison protocol is based on the protocol by Reistad and Toft [25] as implemented in VIFF[1] [13]. Real values, such as model weights, are represented as 16-bit fixed-point numbers. All operations using Shamir shares are performed in a prime field of order $p = 2^{31} - 1$, such that share values can be represented as 32-bit integers. The reconstruction threshold t is set to 3, resulting in a DSP group size m of 5. The key length for the ElGamal cryptosystem is set to 2048 bits to achieve a sufficiently high security level[2].

The setup used for runtime measurements is similar to that used in [17]. The tests were executed on a mobile workstation running Arch Linux on an Intel® Core™ i7-3610QM 2.3 GHz quad-core processor with 8 GB RAM. Similar to [17], all parties are simulated within a single process thread, thus performing all operations sequentially. Figure 1 shows a comparison between the runtimes of BAdASS and the state-of-the-art AHEad protocol for the different protocol

[1] Virtual Ideal Functionality Framework. Available online at `https://github.com/kljensen/viff/blob/master/viff/comparison.py`

[2] See e. g. `https://www.keylength.com` for key lengths as recommended by various organizations. The NIST considers a key length of 2048 sufficiently secure until 2030.

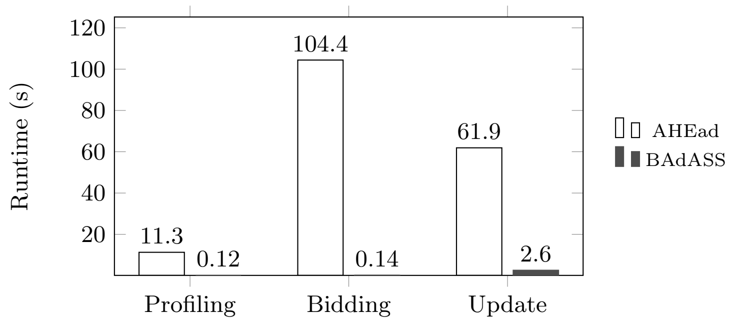

Figure 1. Performance comparison between BAdASS and the state-of-the-art AHEad protocol. The runtimes are measured in similar settings, using profile dimensionality $d = 2^{20}$ for both protocols. Note that the runtime measurements for AHEad are performed using a single DSP running a single campaign, whereas 5 DSPs with a total of 5 campaigns are simulated on a single process thread for BAdASS due to the recombination threshold.

phases. The comparison makes it evident that, for a realistically large profile size $d = 2^{20}$, BAdASS provides significant performance improvements over AHEad for every subprotocol, with the time-sensitive bidding phase requiring less than 150 ms for a DSP group. The computation time required by the model update protocol of BAdASS far exceeds that of the profile update, bidding, and auction protocols, due to the large number of subshare recombinations performed by the DSPs as well as the large number of sharings created by the PSP. Note that the computation time of the model update protocol is averaged, since the expensive mixing and update steps are only performed once every $\zeta = 10$ invocations of the protocol. Specifically, as shown in Table 2, one invocation of these steps is more

Table 2. Runtime measurements for each step of each model update for BAdASS in ms. Mix shares and update model steps are executed once every $\zeta = 10$ invocations while the update preparation step is performed for every viewed ad. The total shows the average of ζ invocations.

Protocol	DSP	PSP
Prepare update	16.87	—
Mix shares	—	2399.67
Update model	23406.70	—
Total	2597.51	

expensive than the preparation step by a factor of 140 and 1400, respectively. The cost of performing the model update protocol when the computation time is averaged becomes

$$\frac{10 \times 16.87 + 2399.67 + 23406.70}{10} = 2597.51 \text{ ms}, \qquad (2)$$

which is approximately 2.6 seconds, as shown in Figure 1. Based on our measurements, we estimate that if the computations performed by the DSPs are parallelized, the average time spent on the model update protocol for a profile dimensionality of $d = 2^{20}$ drops from 2.6 seconds to about 750 ms per invocation. The computation times of the bidding and auction phase and the profile update phase are both below 150 ms for large profile sizes, even when DSPs are simulated sequentially, and thus seem very well suited for use in a real-time setting as required by the RTB advertising model. The relatively large amount of computation performed in the model update phase is less time sensitive, and can thus be periodically performed as a background task without harming the user experience.

In Table 3, we list the average communication bandwidth of our protocol for specific parameters used in our implementation. We use a share size $\sigma = 32$ bits, and a group size of $m = 5$ DSPs. Each DSP is responsible for $\kappa = 10$ campaigns, and with two groups the total number of campaigns is $K = 100$. The profile dimensionality is $d = 2^{20}$, and the size of an advertisement descriptor is assumed to be 4160 bits, of which 4096 bits are the encrypted advertisement, 32 bits the random identifier, and 32 bits the group descriptor. From the table, it is evident that the profile update and model update require a significant amount of communication, with up to 100 MiB per invocation, on average, of the model update protocol. The time-sensitive bidding and auction protocols, however, require very little bandwidth. Moreover, very little bandwidth is used by the user, with only the periodically executed profile update protocol requiring more than a few dozen bytes at 8 MiB per invocation, making the bandwidth use of the user very acceptable for modern unmetered connections.

Table 3. Bandwidth usage of BAdASS in KiB per invocation of each subprotocol, based on realistic parameters used in our implementation of BAdASS.

Protocol	Rounds	User	AdX	DSP	PSP
Profiling	2	8192	20 480	0	20 480
Bidding	2	—	51	6.25	2
Auction	$\lambda(\rho + 1) + 3$	—	0.5	$5 + K_i\gamma$	0.5
Update	$T + 1\frac{3}{\zeta}$	0.02	—	$24\,576 + \tau$	102 400

5 Security of BAdASS

The security requirements of BAdASS are satisfied by the security of the underlying secret-sharing and encryption schemes in the semi-honest setting. In the noninteractive phases of the protocol, both the user profile and model parameters are shared among a DSP group using Shamir's secret sharing scheme, which provides information-theoretic security as long as no more than $t-1$ parties collude.

In the profile update protocol, the user profile is shared between the PSP and the AdX using a two-party additive secret sharing scheme, which, given the assumption that the PSP does not collude with any party, provides information-theoretic security. The additive shares are split into Shamir shares before being sent to a DSP group, where the additive shares are combined into a single Shamir share. Since the PSP and the AdX only receive additive shares, and DSPs obtain a single Shamir share of each additive share, the PSP, AdX and DSPs gain no knowledge of the contents of user profiles.

In the bidding protocol, the PSP obtains values $\boldsymbol{w}_k^\top \boldsymbol{x}_u$ and $\hat{\boldsymbol{y}}_k$ from DSP groups, but does not know the campaign k or user u to which the values belong, nor the specific DSP responsible for the campaign. Since the PSP knows neither $\boldsymbol{w}_k$ nor $\boldsymbol{x}_u$, inferring the individual values of $\boldsymbol{w}_k$ or $\boldsymbol{x}_u$ from $\boldsymbol{w}_k^\top \boldsymbol{x}_u$ is equivalent to the hardness of solving the subset-sum problem. Given a set of positive integers $S = \{a_1, a_2, \cdots a_n\}$ and an integer b, the subset-sum problem aims to find whether there exist a subset of S, for which the summation equals to b [7]. Finding such a subset, however, is an NP-complete problem. In BAdASS, $\boldsymbol{x}_u$ is a vector with binary or small values, and $\boldsymbol{w}_k$ contains 16-bit fixed-point numbers. Assuming a binary $\boldsymbol{x}_u$ the multiplication $\boldsymbol{w}_k^\top \boldsymbol{x}_u$ is actually a selection of indices of $\boldsymbol{w}_k$ based on the value in every index of $\boldsymbol{x}_u$. Finding a subset of $\boldsymbol{w}_k$, where the sum of the subset is equal to $\boldsymbol{w}_k^\top \boldsymbol{x}_u$ is hard, when $\boldsymbol{w}_k$ and $\boldsymbol{x}_u$ are private and both of them have size $d = 2^{20}$.

If the PSP receives multiple values of $\boldsymbol{w}_k^\top \boldsymbol{x}_u$ for the same $\boldsymbol{w}_k$ and $\boldsymbol{x}_u$, the PSP can link these values to the same user, but cannot learn any information about the user's interests as the PSP cannot link response predictions to campaigns. The PSP also receives a mapping between a randomly generated number and an advertisement encrypted using the universal re-encryption scheme, which is semantically secure under the DDH assumption [15]. The use of universal re-encryption provides key privacy, such that the PSP does not learn the identity of the user, and the randomization of ciphertexts in the ElGamal cryptosystem ensures that different submissions of the same advertisement cannot be linked. During the auction protocol, the PSP learns the random number associated with the winning bid in order to retrieve the winning advertisement, but since the mappings are anonymized by the AdX, the PSP cannot link this value to a DSP.

In the model update protocol, the PSP obtains rotated shares of update gradients, bid values, and campaign identifiers. Given unbounded computational power, the PSP can perform an exhaustive search of rotation coefficients until recombination of shares results in likely values. Choosing sufficiently large values for the update period ζ and recombination threshold t makes exhaustive searches infeasible. After the PSP mixes the shares, DSPs receive shares of the same values submitted earlier in the model update phase. Since the shares are re-shared by the PSP, however, DSPs cannot link the shares received after mixing to shares submitted before mixing. Moreover, the random rotation performed by the PSP prevents DSPs from linking inputs to outputs.

6 Conclusion

In this paper we present a novel protocol using machine learning over secret-shared data to preserve privacy in OBA with minimal user-noticeable delays. Trust is distributed among DSPs using threshold secret sharing, allowing DSPs to collaboratively compute bid prices and determine the highest bid without gaining any knowledge of a user's interests. Reports of individual clicks and views of advertisements are secret-shared among DSPs, where they are used to privately update model parameters via a mixing step at the PSP. At no point are the contents of user profiles, shown advertisements, and actual user responses revealed to any party other than the user, nor are model parameters revealed to any party other than the DSP responsible for the campaign. Individual bid prices are not revealed to any party, but are aggregated for billing purposes. Finally, the protocols are integrated into the RTB setting by forming DSP groups from existing DSPs, with the addition of a single new party.

BAdASS achieves significant performance improvements over previous work. The AHEad protocol presented in [17] requires more than 100 seconds of computation time to calculate a single bid value in the time-sensitive bidding phase. In comparison, BAdASS simulates the calculation of 5 bid values in less than 150 milliseconds in a similar setup, even without parallelization across DSPs. BAdASS is even efficient enough to serve advertisements in real time as required by the RTB model, provided the communication between DSPs incurs minimal latency. Despite the overhead of the model update protocol, the results obtained with BAdASS show that by applying secret sharing techniques a level of performance can be achieved in the time-sensitive bidding and auction phases that does not degrade the perceived responsiveness.

To the best of our knowledge, BAdASS is the first protocol to allow sub-second behavioural targeting of advertisements while preserving user privacy. The heavily fragmented shape of the online advertising landscape lends itself particularly well to the use of efficient secret-sharing techniques, giving advertising companies the opportunity to cooperatively move towards acceptable forms of behavioural advertising. Although the presented protocol should be adapted to the malicious setting, as DSPs may have an incentive to modify competitors' bid values, the results obtained with BAdASS show that it is possible to serve behaviourally targeted advertisements without disclosing those interests to any party, all within a fraction of a second. We believe that these results provide a first step towards adoption of privacy-preserving methods in the online advertising ecosystem.

References

1. An, M.: Why people block ads (and what it means for marketers and advertisers). Tech. rep., HubSpot Research (2016)

2. Aono, Y., Hayashi, T., Phong, L.T., Wang, L.: Scalable and secure logistic regression via homomorphic encryption. In: Proceedings of the Sixth ACM on Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9-11, 2016. pp. 142–144. ACM, New York (2016)
3. Backes, M., Kate, A., Maffei, M., Pecina, K.: Obliviad: Provably secure and practical online behavioral advertising. In: IEEE Symposium on Security and Privacy, SP 2012, San Francisco, California, USA, 21-23 May 2012. pp. 257–271. IEEE Computer Society, n.p. (2012)
4. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, Chicago, Illinois, USA, May 2-4, 1988. pp. 1–10. ACM, New York (1988)
5. Chapelle, O., Manavoglu, E., Rosales, R.: Simple and scalable response prediction for display advertising. ACM TIST **5**(4), 61:1–61:34 (2014)
6. Chen, T., Zhong, S.: Privacy-preserving backpropagation neural network learning. IEEE Trans. Neural Networks **20**(10), 1554–1564 (2009)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd Edition. MIT Press, Cambridge, Massachusetts - London, England (2009)
8. Degeling, M., Herrmann, T.: Your interests according to google - A profile-centered analysis for obfuscation of online tracking profiles. CoRR **abs/1601.06371** (2016), `http://arxiv.org/abs/1601.06371`
9. Deighton, J., Brierley, H.M.: Economic value of the advertising-supported internet ecosystem. Tech. rep., Interactive Advertising Bureau (2012)
10. Estrada-Jiménez, J., Parra-Arnau, J., Rodríguez-Hoyos, A., Forné, J.: Online advertising: Analysis of privacy threats and protection approaches. Computer Communications **100**, 32–51 (2017)
11. Fredrikson, M., Livshits, B.: Repriv: Re-imagining content personalization and in-browser privacy. In: 32nd IEEE Symposium on Security and Privacy, S&P 2011, Berkeley, California, USA, 22-25 May 2011. pp. 131–146. IEEE Computer Society, n.p. (2011)
12. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Information Theory **31**(4), 469–472 (1985)
13. Geisler, M.J.B.: Cryptographic Protocols: Theory and Implementation. Ph.D. thesis, Department of Computer Science, Aarhus University (2010), `http://www.forskningsdatabasen.dk/en/catalog/2186116803`
14. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In: Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998. pp. 101–111. ACM, New York (1998)
15. Golle, P., Jakobsson, M., Juels, A., Syverson, P.F.: Universal re-encryption for mixnets. In: Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27,2004. Lecture Notes in Computer Science, vol. 2964, pp. 163–178. Springer, Berlin, Heidelberg (2004)
16. He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., Candela, J.Q.: Practical lessons from predicting clicks on ads at facebook. In: Proceedings of the Eighth International Workshop on Data Mining for Online Advertising, ADKDD 2014, New York City, New York, USA, August 24, 2014. pp. 5:1–5:9. ACM, New York (2014)

17. Helsloot, L.J., Tillem, G., Erkin, Z.: AHEad: Privacy-preserving online behavioural advertising using homomorphic encryption. In: IEEE International Workshop on Information Forensics and Security, WIFS 2017, Rennes, France, December 4-7, 2017. pp. 1–6. IEEE, n.p. (2017)
18. Letang, V., Stillman, L.: Global advertising forecast. Tech. rep., MAGNA (2016)
19. McMahan, H.B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., Golovin, D., Chikkerur, S., Liu, D., Wattenberg, M., Hrafnkelsson, A.M., Boulos, T., Kubica, J.: Ad click prediction: a view from the trenches. In: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013. pp. 1222–1230. ACM, New York (2013)
20. Merzdovnik, G., Huber, M., Buhov, D., Nikiforakis, N., Neuner, S., Schmiedecker, M., Weippl, E.R.: Block me if you can: A large-scale study of tracker-blocking tools. In: 2017 IEEE European Symposium on Security and Privacy, EuroS&P 2017, Paris, France, April 26-28, 2017. pp. 319–333. IEEE, n.p. (2017)
21. Orlandi, C., Piva, A., Barni, M.: Oblivious neural network computing via homomorphic encryption. EURASIP J. Information Security **2007**, 18:1–18:10 (2007)
22. PageFair: The state of the blocked web. Tech. rep. (2017)
23. PageFair, Adobe: The cost of ad blocking. Tech. rep. (2015)
24. Papaodyssefs, F., Iordanou, C., Blackburn, J., Laoutaris, N., Papagiannaki, K.: Web identity translator: Behavioral advertising and identity privacy with WIT. In: Proceedings of the 14th ACM Workshop on Hot Topics in Networks, Philadelphia, PA, USA, November 16 - 17, 2015. pp. 3:1–3:7. ACM, New York (2015)
25. Reistad, T.I., Toft, T.: Secret sharing comparison by transformation and rotation. In: Information Theoretic Security - Second International Conference, ICITS 2007, Madrid, Spain, May 25-29, 2007, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4883, pp. 169–180. Springer, Berlin, Heidelberg (2007)
26. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
27. Szwabe, A., Misiorek, P., Ciesielczyk, M.: Logistic regression setup for RTB CTR estimation. In: Proceedings of the 9th International Conference on Machine Learning and Computing, Singapore, Singapore. pp. 61–70. ICMLC 2017, ACM, New York (2017)
28. Toubiana, V., Narayanan, A., Boneh, D., Nissenbaum, H., Barocas, S.: Adnostic: Privacy preserving targeted advertising. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2010, San Diego, California, USA, February 28 - March 3, 2010. The Internet Society, n.p. (2010)
29. TRUSTe: 2011 Consumer Research Results: Privacy and Online Behavioural Advertising. Tech. rep. (2011)
30. Ur, B., Leon, P.G., Cranor, L.F., Shay, R., Wang, Y.: Smart, useful, scary, creepy: perceptions of online behavioral advertising. In: Symposium On Usable Privacy and Security, SOUPS '12, Washington, DC, USA, July 11-13, 2012. pp. 4:1–4:15. ACM, New York (2012)
31. Wang, J., Zhang, W., Yuan, S.: Display advertising with real-time bidding (RTB) and behavioural targeting. CoRR **abs/1610.03013** (2016), `http://arxiv.org/abs/1610.03013`
32. Weinberger, K.Q., Dasgupta, A., Langford, J., Smola, A.J., Attenberg, J.: Feature hashing for large scale multitask learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009. ACM International Conference Proceeding Series, vol. 382, pp. 1113–1120. ACM, New York (2009)

33. Yan, J., Liu, N., Wang, G., Zhang, W., Jiang, Y., Chen, Z.: How much can behavioral targeting help online advertising? In: Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009. pp. 261–270. ACM, New York (2009)
34. Zhang, Q., Yang, L.T., Chen, Z.: Privacy preserving deep computation model on cloud for big data feature learning. IEEE Trans. Computers **65**(5), 1351–1362 (2016)

A Notation Tables

Table 4. Explanation of symbols used in BAdASS

Symbol	Explanation
u	Unique user identifier.
v	Unique bid request identifier.
k	Unique campaign identifier.
$\mathit{\Gamma}_i$	DSP group i.
$\gamma_{i,j}$	Unique DSP identifier, where $\gamma_{i,j}$ is the j^{th} DSP of DSP group $\mathit{\Gamma}_i$
$K_{\mathit{\Gamma}_i}$	Set of campaigns run by DSP group $\mathit{\Gamma}_i$.
$\boldsymbol{x}$	Input feature vector obtained by feature hashing.
$\boldsymbol{w}_k$	Model parameter vector for a campaign k, where $w_{k,i}$ is the i^{th} coordinate of $\boldsymbol{w}_k$.
$\boldsymbol{c}_k$	Bidding function parameters for a campaign k.
η_k	Learning rate parameter for campaign k.
b_k	Bid value for campaign k.
a_k	Advertisement associated with campaign k.
$\pi(\boldsymbol{x})$	Random permutation function, which re-orders the elements of vector $\boldsymbol{x}$.
$\pi^{-1}(\cdot)$	Inverse permutation of $\pi(\cdot)$, such that $\pi^{-1}(\pi(\boldsymbol{x})) = \boldsymbol{x}$.
M	List of vectors containing information associated with bid values for use in the auction protocol.

Table 5. Symbols used in the computational analysis of BAdASS.

Symbol	Description
d	Dimensionality of user profiles.
n	Number of DSPs.
g	Number of DSP groups.
m	Size of a DSP group.
κ	Number of campaigns of a DSP.
K	Total number of campaigns.
K_i	Number of campaigns within a DSP group $\mathit{\Gamma}_i$.
ζ	Number of model updates accumulated per DSP group.
σ	Size in bits of a secret share.
ξ	Size in bits of an advertisement tuple, consisting of an encrypted advertisement, a group descriptor, and a random identifier.
λ	Total number of comparisons required to find the maximum bid. Equal to $\lceil \log_2 (K_i - 1) \rceil + \lceil \log_2 (g - 1) \rceil$.
ρ	Number of rounds required by a single run of the comparison protocol.
γ	Number of bits transmitted in a single run of the comparison protocol.
T	Number of rounds required by a single run of the truncation protocol.
τ	Number of bits transmitted in a single run of the truncation protocol.

B Protocols

Protocol 2 Profile update protocol, executed jointly between a user, ad exchange, PSP and DSP group, and initiated periodically by every user.

```
procedure USER:SEND-PROFILE-SHARE($\boldsymbol{x}, u$)
    Pick $\boldsymbol{r} \in_R \mathbb{Z}_p^d$
    $\langle\!\langle \boldsymbol{x} \rangle\!\rangle_1 \leftarrow \boldsymbol{x} - \boldsymbol{r}$
    $\langle\!\langle \boldsymbol{x} \rangle\!\rangle_2 \leftarrow \boldsymbol{r}$
    invoke SHARE-PROFILE($u, \langle\!\langle \boldsymbol{x} \rangle\!\rangle_1$) at AdX
    invoke SHARE-PROFILE($u, \langle\!\langle \boldsymbol{x} \rangle\!\rangle_2$) at PSP
end procedure

procedure SHARE-PROFILE($u, \langle\!\langle \boldsymbol{x} \rangle\!\rangle_m$)
    $\langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_m \rangle \leftarrow$ SHAMIR-SHARE($\langle\!\langle \boldsymbol{x} \rangle\!\rangle_m$)
    for all $\gamma_{i,j} \in \Gamma_i$ do
        invoke DSP:COMBINE-PROFILE($u, \langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_m \rangle_{\gamma_{i,j}}$) at DSP $\gamma_{i,j}$
    end for
end procedure

procedure DSP:COMBINE-PROFILE($u, \langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_m \rangle$)
    store $\langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_m \rangle$ for user $u$
    if $\langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_1 \rangle$ and $\langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_2 \rangle$ are both stored then
        $\langle \boldsymbol{x}_u \rangle \leftarrow \langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_1 \rangle + \langle \langle\!\langle \boldsymbol{x} \rangle\!\rangle_2 \rangle$
    end if
end procedure
```

Protocol 3 Auction protocol, executed jointly by every DSP group Γ_i, an auction group Γ^*, and the PSP.

```
1: procedure DSP:PREPARE-AUCTION($v, \langle \boldsymbol{b} \rangle, \langle \boldsymbol{r} \rangle, \langle \hat{\boldsymbol{y}} \rangle, \langle \boldsymbol{\eta} \rangle, \langle \boldsymbol{k} \rangle$)
2:     for all $\Gamma_i$ do
3:         $\langle M_i \rangle \leftarrow \big( \langle \boldsymbol{r_i} \rangle, \langle \hat{\boldsymbol{y}}_{\boldsymbol{i}} \rangle, \langle \boldsymbol{\eta_i} \rangle, \langle \boldsymbol{k_i} \rangle \big)$
4:         $\big( \langle b_i^{max} \rangle, \langle r_i^{max} \rangle, \langle \hat{y}_i^{max} \rangle, \langle \eta_i^{max} \rangle, \langle k_i^{max} \rangle \big) \leftarrow$ MAX-BID$(\langle \boldsymbol{b_i} \rangle, \langle M_i \rangle)$
5:         Store mapping $v \rightarrow (\langle b_i^{max} \rangle, \langle \hat{y}_i^{max} \rangle, \langle \eta_i^{max} \rangle, \langle k_i^{max} \rangle)$
6:     end for
7:     invoke PERFORM-AUCTION($v, \langle \boldsymbol{b^{max}} \rangle, \langle \boldsymbol{r^{max}} \rangle$) at $\Gamma^*$
8: end procedure

9: procedure PERFORM-AUCTION($v, \langle \boldsymbol{b} \rangle, \langle \boldsymbol{r} \rangle$)
10:     $\big( \perp, \langle r^{max} \rangle \big) \leftarrow$ MAX-BID$(\langle \boldsymbol{b} \rangle, \langle \boldsymbol{r} \rangle)$
11:     invoke PSP:SEND-AD($\langle r^{max} \rangle$) at PSP
12: end procedure

13: procedure MAX-BID($\langle \boldsymbol{b} \rangle, \langle M \rangle$)
14:     $\langle \hat{b} \rangle \leftarrow \langle b_1 \rangle$
15:     for $j \leftarrow 1, \big| \langle M \rangle \big|$ do
16:         $\langle \hat{M}_j \rangle \leftarrow \langle M_{j,1} \rangle$
17:     end for
18:     for $i \leftarrow 2, \big| \langle \boldsymbol{b} \rangle \big|$ do
19:         $\langle \rho \rangle \leftarrow \langle b_i \rangle \geq \langle \hat{b} \rangle$
20:         $\langle \hat{b} \rangle \leftarrow \langle \rho \rangle \cdot \langle b_i \rangle + (1 - \langle \rho \rangle) \cdot \langle \hat{b} \rangle$
21:         for $j \leftarrow 1, \big| \langle M \rangle \big|$ do
22:             $\langle \hat{M}_j \rangle \leftarrow \langle \rho \rangle \cdot \langle M_{j,i} \rangle + (1 - \langle \rho \rangle) \cdot \langle \hat{M}_j \rangle$
23:         end for
24:     end for
25:     return $\langle \hat{b} \rangle, \langle \hat{M} \rangle$
26: end procedure

27: procedure PSP:SEND-AD($\langle r \rangle$)
28:     $r \leftarrow$ combine $\langle r \rangle$
29:     $\big( [\![a]\!]_u, \Gamma_i \big) \leftarrow$ lookup $r$
30:     Re-randomize $[\![a]\!]_u$
31:     Send $\big( [\![a]\!]_u, \Gamma_i \big)$ to user via AdX
32: end procedure
```

Protocol 4 Model update protocol, invoked by the user at the DSP group responsible for the displayed advertisement.

```
1: procedure DSP:PREPARE-MODEL-UPDATE($v, u, \mathit{\Gamma}_i, \langle y \rangle$)
2:     $\big(\langle b \rangle, \langle \hat{y} \rangle, \langle \eta \rangle, \langle k \rangle\big) \leftarrow$ lookup $v$
3:     $\langle \delta \rangle \leftarrow$ TRUNCATE($\langle \eta \rangle \cdot (\langle \hat{y} \rangle - \langle y \rangle)$)
4:     for $i \leftarrow 1, d$ do
5:         $\langle g_i \rangle \leftarrow \langle x_{u,i} \rangle \cdot \langle \delta \rangle$
6:     end for
7:     $\big(\langle G \rangle, \langle B \rangle, \langle K \rangle\big) \leftarrow \big(\langle G \rangle \cup \langle \boldsymbol{g} \rangle, \langle B \rangle \cup \langle b \rangle, \langle K \rangle \cup \langle k \rangle\big)$
8:     if sufficient values are accumulated then
9:         for all $\gamma_{i,j} \in \mathit{\Gamma}_i$ do
10:            Pick random $r_{i,j}$
11:            Rotate $\big(\langle G \rangle_{i,j}, \langle B \rangle_{i,j}, \langle K \rangle_{i,j}\big)$ $r_{i,j}$ times
12:        end for
13:        invoke PSP:MIX-SHARES($\mathit{\Gamma}_i, \langle G \rangle, \langle B \rangle, \langle K \rangle$) at PSP
14:    end if
15: end procedure

16: procedure PSP:MIX-SHARES($\mathit{\Gamma}_i, \langle G \rangle, \langle B \rangle, \langle K \rangle$)
17:    if shares from all $\gamma_{i,j} \in \mathit{\Gamma}_i$ have been received then
18:        Rotate $\big(\langle G \rangle, \langle B \rangle, \langle K \rangle\big)$ by a random value
19:        Re-share $\big(\langle G \rangle, \langle B \rangle, \langle K \rangle\big)$ as $\big(\langle G' \rangle, \langle B' \rangle, \langle K' \rangle\big)$
20:        invoke UPDATE-MODEL($\langle G' \rangle, \langle B' \rangle, \langle K' \rangle$) at $\mathit{\Gamma}_i$
21:    end if
22: end procedure

23: procedure DSP:UPDATE-MODEL($\langle G' \rangle, \langle B' \rangle, \langle K' \rangle$)
24:    for all $\gamma_{i,j} \in \mathit{\Gamma}_i$ do
25:        Rotate $\big(\langle G' \rangle_{i,j}, \langle B' \rangle_{i,j}, \langle K' \rangle_{i,j}\big)$ back $r_{i,j}$ times
26:    end for
27:    $K \leftarrow$ combine $\langle K' \rangle$
28:    for all $\langle \boldsymbol{g}' \rangle \in \langle G' \rangle,\ \langle b' \rangle \in \langle B' \rangle,\ k \in K$ do
29:        for $i \leftarrow 1, d$ do
30:            $\langle w_{k,i} \rangle \leftarrow \langle w_{k,i} \rangle - \langle g'_i \rangle$
31:        end for
32:        $\langle \tilde{b}_k \rangle \leftarrow \langle \tilde{b}_k \rangle + \langle b' \rangle$
33:    end for
34: end procedure
```