Privacy-preserving Location-aware Mobile Advertisement

Erald Troja
The Graduate Center
City University of New York
Email: etroja@gradcenter.cuny.edu

Spiridon Bakiras
Dept. of Computer Science
Michigan Technological University
Email: sbakiras@mtu.edu

***Abstract*—Location-aware mobile advertising is expanding very rapidly and is forecast to grow much faster than any other industry in the digital era. Unfortunately, with the rise and expansion of online behavioral advertising, consumers have grown very skeptical of the vast amount of data that is extracted and mined from advertisers. As a result, the consensus has shifted towards stricter privacy requirements. In this paper, we introduce a novel privacy-preserving location-aware mobile advertisement framework that is built with privacy in mind from the ground up. Our proposed methods ease the tension that exists between privacy and advertising by guaranteeing, through cryptographic constructions, that (i) mobile users receive advertisements relative to their location and interests in a privacy-preserving manner, and (ii) the advertisement network can only compute aggregate statistics of ad impressions and click-through-rates. Through extensive experimentation, we show that our methods are efficient in terms of both computational and communication cost, especially at the client side.**

I. Introduction

The rise in popularity of smartphone applications has led to an exponential growth in privacy concerns, mainly due to the constant consumer tracking and profiling that stems from abusive data capturing and sharing strategies. According to a recent study [23], 66% of location-aware applications have privacy policies that actually amount to very little when it comes to protecting the privacy of the data that is collected and shared by the application. A classic case study is that of the Google engineer who violated the company's strict privacy policy rules, by breaking into the Gmail and Voice accounts of Google users [3].

To eliminate the privacy concerns of online targeted advertisement, a privacy-preserving *ad network* should address two important issues, namely ad delivery and statistics collection. First, ad delivery should guarantee that the network is oblivious to the content sent to the clients. Second, statistics collection should keep track of the aggregate number of times that a certain ad is displayed to the users (known as ad impression or click-through-rate) without knowledge of individual statistics. These aggregate measurements are essential, because they form the basis on which the ad network bills the advertisers. Although several privacy-preserving ad networks exist in the literature, some do not support location-aware ads [12], [22], others do not encrypt the ads sent to the clients [9], [10], [22], a few employ a trusted third-party in their architecture [8], [22], others rely on non-colluding servers [10], [12], and some require specialized network infrastructures [9].

In this paper, we introduce a novel privacy-preserving location-aware mobile advertisement framework that is built with privacy in mind from the ground up. Our methods ease the tension that exists between privacy and advertising by guaranteeing, through cryptographic constructions, that (i) mobile users receive advertisements relative to their location and interests encrypted with their own public keys, and (ii) the ad network can only compute aggregate statistics of ad impressions and click-through-rates (CTRs). Unlike previous work, we do not employ trusted third-parties and all our protocols are secure against collusions. To the best of our knowledge, this is the first privacy-preserving location-aware ad network in the literature with such properties.

Our basic ad delivery method leverages a simplified version of the private stream searching protocol by Ostrovsky and Skeith III [17]. Specifically, we partition space into a regular grid, and allow clients to privately retrieve the ads from the grid cell that they currently reside in. This is done by sending to the ad network's server one Paillier [18] ciphertext for each grid cell. All ciphertexts contain encryptions of 0, except for the one corresponding to the client's cell that contains an encryption of 1. With these ciphertexts, the server prepares an encrypted ad buffer that is then decrypted by the client with his private key. Next, we identify a performance bottleneck at the client side when the number of cells is large, which leads to significant computational and communication costs. To this end, we introduce an improved protocol, based on the "somewhat" homomorphic cryptosystem of Boneh, Goh, and Nissim (BGN) [2]. The properties of the BGN cryptosystem allow us to identify the cell of interest through its row/column id and, therefore, does not require a unique ciphertext for each grid cell.

Our final contribution includes a privacy-preserving aggregation protocol that collects ad impression/CTR statistics at the ad network server, without leaking any information regarding individual customers. It is based on a distributed version of the ElGamal cryptosystem [4] and has several desirable properties. First, it is very efficient for dynamic membership groups, i.e., when new customers join or existing customers leave the system. Furthermore, it is resistant against collusions among the clients, and is computationally efficient. Through

extensive experimentation, we show that our protocols are efficient in terms of both computational and communication cost, especially at the client side.

II. Related work

The earliest work towards privacy-preserving advertisement is due to Juels [12]. He proposes the notion of a *negotiant*, which serves as a client-side proxy to protect user information and direct the targeting of advertisements. The idea is to allow each customer equipped with a public/private key to publish his ad request on a bulletin board. When enough ads are accumulated or some other triggering criterion occurs, a network of m non-colluding servers (advertisers) mixes the requests, and then uses distributed plaintext equality test to perform a blind lookup of consumer ad requests. The scheme by Juels does not consider other aspects of online advertising, such as privacy-preserving aggregation of ad statistics.

Adnostic [22] proposes a practical advertisement architecture that enables ad targeting without compromising user privacy. Specifically, behavioral profiling and ad targeting is privately done in the user's browser. The ad network periodically selects a number of ads that are sent (in plaintext) to the user's browser, where a rendering module chooses an ad for display. Adnostic's main goal is to complement existing behavioral advertising infrastructure, by providing efficient cryptographic billing based on homomorphic encryption and efficient zero-knowledge proofs (ZKPs). Nevertheless, their aggregation scheme makes use of a trusted third-party for decrypting the ad impression counters. In a similar fashion to Adnostic, Privad [8] preserves privacy by maintaining profiles on the user's computer instead of the server. Furthermore, it employs an anonymizing proxy, called *dealer*, which sits between the clients and the ad network in order to hide any personally identifying information. The main limitation of Privad is that it trusts that the ad network does not collude with the dealer.

MobiAd [9] proposes that end users keep a local private profile of categories of interest on their mobile devices. Ads are constantly broadcast on the local mobile base station, and the device's profile is responsible for downloading ads that are relevant to the user's interests. Statistics, such as CTRs or ad impressions, are updated via anonymization techniques. Finally, Hardt and Nath [10] provide a privacy-aware personalization scheme for mobile advertising. The main contribution of their work is the formalization of a common framework for personalized ad delivery, which can be instantiated at any required trade-off point between ad relevance, privacy, and efficiency. First, in an interactive manner, the client releases limited information to the ad network, such as a broad category of interest over a cloaked region. The server then pushes (in plaintext) the most relevant ads to the client, who filters them based on private criteria held at the device.

In the realm of privacy-preserving aggregation protocols, numerous approaches in the literature leverage a *trusted third-party* to perform specific tasks [10], [11], [15], [21]. Although trusted third-parties simplify the construction of privacy-preserving protocols, they are not realistic for practical applications. To this end, Acs and Castelluccia [1], and Erkin and Tsudik [5] generate random secrets in a distributed manner among the participating users. While their underlying encryption protocols are different, both methods are very similar in the way they construct the secrets. Their main limitation, however, is that they are prone to collusions among the users.

Jung et al. [13] and Yang et al. [24] allow each user to compute a random key that is used to mask the underlying measurement. The keys are computed in a way, such that they cancel out once they are aggregated at the server. Even though these schemes are not prone to collusions, they are very inefficient because they necessitate expensive re-keying operations for every aggregation instance.

III. Privacy-preserving Ad Delivery

We consider a system where mobile users receive ads, in a *streaming* fashion, from businesses in their proximity. The streaming nature of the data discourages the use of private information retrieval (PIR) protocols [7], [14], since such protocols assume a static database whose exact specifications are known to all users. Instead, our goal is for each user to submit a *single* query to the ad network (representing their current location) and have the ad network filter out the streaming ads based on each user's location. Ads are delivered through a dedicated ad network that bills the advertisers based on the number of times their ads are displayed to the users (ad impressions). To determine proximity, we assume that space is partitioned into a regular $N \times N$ grid, where each grid cell stores the ads that are physically located therein. Similarly, users map their GPS coordinates into a unique cell, and use the cell's position as an input to their query.

Ads consist of a tuple $\langle id, cid, loc, text \rangle$, where id is the ad's unique identifier, cid identifies the advertisement category (e.g., food, shopping), loc contains the GPS coordinates of the ad's product, and $text$ is the message that is displayed to the user. We assume that the text part of the ad is a few hundred bytes in size, and is padded (if needed) so that all ads have identical text size. At the client side, we assume the existence of a profiling engine that monitors the user's browsing behavior and builds a private profile for that user. Table I contains a summary of symbols used in the remainder of this paper.

TABLE I
Summary of Symbols

Symbol	Description
n	Number of users in the system
N	Grid granularity
$\mathcal{A}$	The set of all advertisements
$\mathcal{L}$	The set of all locations
$\mathcal{A}_l \subset \mathcal{A}$	The set of ads matching location $l \in \mathcal{L}$
$\lvert\mathcal{A}_l\rvert_{max}$	Max number of ads across all locations $l \in \mathcal{L}$
$\mathcal{Q}, \mathcal{R}, \mathcal{C}$	Encrypted query vectors
$\mathcal{PK}$	Any public key
$\mathcal{P}$	Any private key
$\mathcal{B}$	Encrypted buffer
m	Number of ciphertexts required to store one ad

Threat model. We consider the ad network (or any entity that has compromised the network's server) as the adversary, whose goal is to pinpoint a user's location into an area smaller than the whole data space. In this work, we aim at perfect privacy, i.e., the adversary should remain oblivious to the location of the users. We assume that the adversary is polynomial time and follows the *honest-but- curious* (or *semi-honest*) model, i.e., it will execute the protocol correctly, but will try to gain an advantage by examining the transcript of the messages that are exchanged during protocol execution.

A. Basic protocol

Our basic scheme is roughly based on the private stream searching protocol by Ostrovsky and Skeith III [17]. The intuition is to allow the client to specify the location of interest through an encrypted vector, and then use the additive homomorphism of the Paillier cryptosystem to encode the results into an encrypted buffer. We assume that all clients generate their Paillier keys locally when they first register in the system, and send the corresponding public keys to the server along with their queries. In what follows, we present our method in terms of three phases, namely, query generation, query processing, and result extraction.

Query generation. The client uses his public key $\mathcal{PK}$ to construct a vector $\mathcal{Q}$ of length $|\mathcal{L}| = N^2$, as shown in Algorithm 1. Every element in the vector is an encryption of 0, except for the element that corresponds to the user's location l, which is an encryption of 1. Due to the semantic security of the Paillier cryptosystem, these ciphertexts are indistinguishable to the adversary.

Algorithm 1 Query generation (Paillier)

```
1: procedure GEN-QUERY-PAILLIER(PK, loc)
2:     map GPS location loc into cell l ∈ L;
3:     for each location i ∈ L do
4:         if i == l then
5:             Q_i ← E_PK(1);
6:         else
7:             Q_i ← E_PK(0);
8:         end if
9:     end for
10:    return Q;
11: end procedure
```

The query generation algorithm incurs an $\mathcal{O}(N^2)$ computational and communication cost, which is significant when N is large. However, one way to eliminate the online computational cost is to precompute offline (e.g., during night time, when the phone is charging) encryptions of 0, which is the major performance bottleneck at the client.

Query processing. After receiving the query vector $\mathcal{Q}$ and public key $\mathcal{PK}$, the ad network must process all the ads in $\mathcal{A}$ and return the relevant ones to the client. We assume that every ad fits in exactly m ciphertexts, where m depends on the underlying cryptosystem and key size. For example, if we use a 1024-bit RSA modulus (for Paillier), a 512-byte ad requires $m = 4$ ciphertexts. The main idea is to construct an encrypted buffer $\mathcal{B}$ that is capable of holding the maximum number of ads across any location $l \in \mathcal{L}$, i.e., $|\mathcal{A}_l|_{max}$. That is, the buffer should consist of exactly $m \cdot |\mathcal{A}_l|_{max}$ ciphertexts.

Algorithm 2 illustrates the procedure that generates the encrypted buffer at the ad network server. $\mathcal{B}$ is initially populated with encryptions of 0 (lines 2–3) that are computed with the client's public key (for efficiency, we use the same ciphertext for all entries). Next, the server processes the ads on a per cell basis. Each ad is split into m pieces, which are subsequently added into m consecutive locations on the buffer (lines 9–12). Note that, when $\mathcal{Q}_l$ is an encryption of 0, the process has no effect on the buffer contents. It is also worth noting that the round-robin manner in which we iterate over the buffer guarantees that there are no collisions when writing back the results. Consequently, when the procedure terminates, $\mathcal{B}$ contains (i) the encrypted ads corresponding to the queried location in successive order starting from a random position in the buffer, and (ii) encryptions of 0 at all the remaining positions.

Algorithm 2 Query processing (Paillier)

```
1: procedure GEN-BUFFER(PK, Q, A)
2:     for i in 1 to m · |A_l|_max do
3:         B_i ← E_PK(0);
4:     end for
5:     i ← 0;
6:     for each cell l ∈ L do
7:         for each ad a ∈ A_l do
8:             split advertisement a into m pieces;
9:             for k in 1 to m do
10:                B_i ← B_i · Q_l^{a_k};
11:                i ← (++i)%(m · |A_l|_max);
12:            end for
13:        end for
14:    end for
15:    return B;
16: end procedure
```

The computational cost at the server is $\mathcal{O}(m \cdot |\mathcal{A}|)$ modular exponentiations and multiplications, while the communication cost entails the transmission of $\mathcal{O}(m \cdot |\mathcal{A}_l|_{max})$ ciphertexts. The Paillier cryptosystem is an ideal choice in this case, because it can decrypt arbitrarily large plaintexts (i.e., the value of m is small).

Result extraction. The result extraction procedure is fairly straightforward, i.e., the client simply decrypts with his private key $\mathcal{P}$ the ciphertexts that comprise buffer $\mathcal{B}$. Algorithm 3 summarizes the decryption process. Starting with the first ciphertext, the client decrypts it and checks whether it contains a useful ad. In particular, if the resulting plaintext M is 0, the corresponding position is empty. Furthermore, by decrypting the first part of the ad, the client recovers its cid value, thus determining (with the help of the profiling engine) whether the ad matches the user's profile. If the ad is not helpful to the client, the algorithm skips the remaining $m - 1$ ciphertexts and moves to the next ad (lines 4–6). Otherwise, the remaining ciphertexts are decrypted to reconstruct the entire ad (lines 8–12). When the algorithm terminates, the profiling engine moves the decrypted ads into the queue that is scheduled for display. The computational cost of this algorithm is $\mathcal{O}(m \cdot |\mathcal{A}_l|_{max})$ decryption operations.

Algorithm 3 Result extraction

```
1: procedure GET-RESULTS(P, B)
2:     for i in 1 to m · |A_l|_max do
3:         M ← D_P(B_i);
4:         if M == 0 or ad does not match interest then
5:             i ← i + m;
6:             continue;
7:         else
8:             initialize new ad with M;
9:             for j in 1 to m − 1 do
10:                M ← D_P(B_{i+j});
11:                add M to currently constructed ad;
12:            end for
13:            i ← i + m;
14:        end if
15:    end for
16:    return ads;
17: end procedure
```

Security. Note that the three phases of our basic scheme constitute a secure two-party computation protocol between the client and the server. As such, we can prove the security of the protocol for honest-but-curious adversaries, following the simulation paradigm [16]. It suffices to show that we are able to simulate the distribution of the messages that each party receives, given only the party's input and output in the protocol. The intuition is that, if we can simulate a party's messages knowing only their input and output, then the messages themselves cannot reveal any additional information.

First, the client's input consists of a binary query vector, and the output contains a number of ads matching a location. The only messages that the client receives from the server are a series of Paillier ciphertexts corresponding to $\mathcal{B}$. The simulator has knowledge of the client's public key and it also knows the decrypted ads. Therefore, it can simply reconstruct a version of the encrypted buffer from scratch. For the server, the input is the set of ads $\mathcal{A}$ and there is no output. The server only receives N^2 Paillier ciphertexts from the client, so the simulator can simply generate N^2 encryptions of 0. Given the semantic security of Paillier's cryptosystem, the server cannot distinguish these ciphertexts from the ones that are produced by the client's real input.

B. Query-efficient protocol

Our basic scheme is very efficient in terms of query processing and result extraction, but suffers from a $\mathcal{O}(N^2)$ cost in the query generation phase. As a result, it does not scale well for finer grids and is impractical for online queries, i.e., without offline pre-computations. To this end, we propose an enhanced version of the protocol that eliminates the need to send a unique ciphertext for every cell of the grid. The main idea is to identify the cell of interest with two encrypted binary vectors: one representing the rows and the other representing the columns. The server would then need to multiply the corresponding bits for every cell, in order to determine whether the user is interested in that location or not (i.e., whether the result is 1 or 0). Next, we present the query generation and processing algorithms of this approach. Note that the result extraction phase is identical to the one in the basic protocol and is, thus, omitted from our discussion.

Query generation. Clearly, our idea necessitates a cryptosystem that allows for both multiplication and addition of plaintexts in the ciphertext domain. This is the definition of fully homomorphic encryption [6] which, unfortunately, is not practical yet for real world applications. Fortunately, in our case, we only need to perform a single multiplication and an arbitrary number of additions, which is precisely what the BGN cryptosystem offers. Therefore, as shown in Algorithm 4, the client uses his BGN public key $\mathcal{PK}$ to construct two encrypted vectors, $\mathcal{R}$ and $\mathcal{C}$, of length N. All the elements contain encryptions of 0, except for the ones that correspond to the user's row/column id. Due to the semantic security of the BGN cryptosystem, the ciphertexts are indistinguishable to an adversary. The algorithm is clearly more efficient than its Paillier counterpart, incurring a $\mathcal{O}(N)$ computational and communication cost at the client.

Algorithm 4 Query generation (BGN)

```
1: procedure GEN-QUERY-BGN(PK, loc)
2:     map GPS location loc into cell (i, j);
3:     for k in 0 to N − 1 do
4:         if k == i then
5:             R_k ← E_PK(1);
6:         else
7:             R_k ← E_PK(0);
8:         end if
9:         if k == j then
10:            C_k ← E_PK(1);
11:        else
12:            C_k ← E_PK(0);
13:        end if
14:    end for
15:    return R, C;
16: end procedure
```

Query processing. The query processing phase is identical to the one described in the basic protocol, i.e., the server prepares an empty (encrypted) buffer $\mathcal{B}$ that eventually stores the ads that are relevant to the user's location. However, in this case, the server must first compute the query vector $\mathcal{Q}$ corresponding to all locations $l \in \mathcal{L}$. As shown in Algorithm 5, for each location $l \in \mathcal{L}$, the server computes $\mathcal{Q}_l = E_{\mathcal{PK}}(b_i \cdot b_j)$, where b_i and b_j are the query bits of the location's row and column ids. The ciphertext is computed through a bilinear map of the respective elements in $\mathcal{R}$ and $\mathcal{C}$ (line 4).

Algorithm 5 Query processing (BGN)

```
1: procedure GEN-BUFFER-BGN(PK, R, C, A)
2:     for each location l ∈ L do
3:         map l into cell (i, j);
4:         Q_l ← e(R_i, C_j);
5:     end for
6:     GEN-BUFFER(PK, Q, A);
7: end procedure
```

The above algorithm entails $\mathcal{O}(N^2)$ bilinear map operations, as well as $\mathcal{O}(m \cdot |\mathcal{A}|)$ modular exponentiations and multiplications for constructing the encrypted buffer. On the other hand, the communication cost involves the transfer of $\mathcal{O}(m \cdot |\mathcal{A}_l|_{max})$

ciphertexts. Compared to the basic scheme, we make the following two observations. First, the BGN-based protocol shifts the computational burden from the clients to the server. Instead of having the clients compute the query vector $\mathcal{Q}$, we provide the server with the minimal information needed to compute $\mathcal{Q}$ locally. This is significant improvement, because mobile devices have limited computational capabilities compared to a state-of-the-art many-core server.

Second, due to the discrete log nature of BGN, the value of m is significantly larger compared to the basic scheme. In our implementation, we choose to encrypt ads in 3-byte chunks, in order to take advantage of a pre-computed table to speed up the discrete log computation at the client. As a result, for a 512-byte ad, m is equal to 171, as opposed to 4 in the case of a 1024-bit Paillier key. Nevertheless, as we show in our experimental results, the overall cost is considerably lower compared to the basic scheme.

IV. Privacy-preserving Collection of Ad Impressions

A fundamental component in a privacy-preserving ad network is its ability to compute aggregate statistics regarding the ads that are displayed to the users (ad impressions). In our system, we assume that the ad network collects the statistics at the end of each day, when most users' devices are idle and connected on the home WiFi network. The server first informs the clients of the total number of ads $|\mathcal{A}|$ that were scheduled that day and, for every ad in $\mathcal{A}$, each user must submit (privately) a bit indicating whether that ad was displayed or not. The server then aggregates the bits from all users, and updates the billing information for the underlying advertiser. Our aggregation protocol is based on a distributed version of the ElGamal cryptosystem, as given by Pedersen [20]. We chose the ElGamal cryptosystem, because of its simple key generation process and overall computational efficiency. The protocol consists of two phases, namely key generation and interactive aggregation.

Threat model. We consider both the ad network and the clients as adversaries in this setting. They all follow the semi-honest adversarial model, and their goal is to identify any non-trivial information regarding individual measurements submitted by users. Note that, semi-honest behavior does not prohibit collusions among the different players, so users may collude by sharing private information. Our protocol has two desirable properties: (i) it does not employ a trusted third-party and (ii) it is secure against any number of colluding parties.

Key generation. All users and the server share a description of a cyclic group $\mathbb{G}$ of prime order q, and two generators of $\mathbb{G}$, namely g and y. The objective is for the n users to collectively compute a public key $h = g^x$, such that $x = \sum_{i=1}^{n} x_i$ is the private key and x_i is user i's secret share of the key. In other words, the private key is distributed to all users in the system and, therefore, decryption necessitates input from all n users.

Algorithm 6 illustrates the key generation process. Initially, each user selects a random secret $x_i \in \mathbb{Z}_q$ and commits [19] to input g^{x_i}, by sending a commitment $C_i(g^{x_i}, r_i)$ to the server. The commitment is simply an encryption of the user's input with a random key r_i (line 4), and its purpose is to prohibit users from modifying their inputs in the later stages of the algorithm. After a user downloads the set of all commitments, he reveals his public input by sending the tuple (g^{x_i}, r_i) to the server (lines 7–8). Finally, each user verifies all commitments and computes locally the ElGamal public key h (lines 11–13). In terms of performance, the algorithm requires $\mathcal{O}(n)$ modular exponentiations and multiplications, and involves the exchange of $\mathcal{O}(n)$ ciphertexts and random secrets.

Algorithm 6 Distributed key generation

```
1:  procedure GEN-KEY(G, q, g, y)
2:      for each user i do
3:          select x_i and r_i uniformly at random from Z_q;
4:          upload commitment C_i(g^{x_i}, r_i) = g^{x_i} · y^{r_i};
5:      end for
6:      for each user i do
7:          download commitments C_1, ..., C_n;
8:          upload (g^{x_i}, r_i);
9:      end for
10:     for each user i do
11:         download (g^{x_i}, r_i) for i ∈ {1, ..., n};
12:         verify C_i(g^{x_i}, r_i) = g^{x_i} · y^{r_i} for i ∈ {1, ..., n};
13:         compute public key h = ∏_{i=1}^{n} g^{x_i};
14:     end for
15: end procedure
```

An important feature in a privacy-preserving aggregation protocol is efficient key management. Prior methods that do not employ trusted third-parties, such as [13] and [24], require expensive re-keying operations for each aggregated value. On the other hand, our approach leverages the ElGamal cryptosystem with a pre-established key, so all values are aggregated under the same public key. Furthermore, our method handles user deletions trivially. In particular, when user i leaves the system, the public key is updated as $h' = h \cdot (g^{x_i})^{-1}$, i.e., the user's secret share is removed from the private/public key. New users, however, necessitate the invocation of the distributed key generation algorithm. To reduce the cost of frequent key generation operations, the server may choose to perform batch insertions.

Interactive aggregation. Having established the public encryption key h, the aggregation of the ad impressions is performed at the ad network's server. Algorithm 7 summarizes the aggregation protocol for a single ad. The first step is for all users to upload their encrypted bits b_i at the server (lines 3–4). Next, the server leverages the additive homomorphism of ElGamal to produce the encryption of $b = \sum_{i=1}^{n} b_i$ (lines 6–7). However, the server is unable to decrypt the result, and has to rely on the n users to perform the decryption. Observe that the decryption function necessitates the computation of h^{-r}, which is equal to $(g^r)^{-x}$. To this end, each user downloads g^r and submits to the server $(g^r)^{-x_i}$ (lines 9–10). Finally, the server aggregates these values to compute h^{-r} and proceeds to recover b (lines 12–14). The overall (across all ads) computational cost at the client consists of $\mathcal{O}(|\mathcal{A}|)$ ElGamal encryptions and $\mathcal{O}(|\mathcal{A}|)$ modular exponentiations, while the communication cost entails the exchange of $\mathcal{O}(|\mathcal{A}|)$

ciphertexts. At the server side, each client contributes $\mathcal{O}(|\mathcal{A}|)$ modular multiplications, while the discrete log computations can be avoided by using a hash table of pre-computed values.

Algorithm 7 Aggregation protocol for a single value

procedure GEN-AGGREGATE($\mathbb{G}, q, g, h$)
  for each user i **do**
    select r_i uniformly at random from $\mathbb{Z}_q$;
    upload encrypted bit b_i as tuple $(g^{r_i}, h^{b_i+r_i})$;
  end for
  server: compute $g^r = \prod_{i=1}^{n} g^{r_i}$;
  server: compute $h^{b+r} = \prod_{i=1}^{n} h^{b_i+r_i}$;
  for each user i **do**
    download g^r;
    upload $(g^r)^{-x_i}$;
  end for
  server: compute $h^{-r} = \prod_{i=1}^{n} (g^r)^{-x_i}$;
  server: compute $h^b = h^{-r} \cdot h^{b+r}$;
  server: solve discrete log to recover $b = \sum_{i=1}^{n} b_i$;
  RETURN b;
end procedure

Security. Our aggregation protocol inherits the semantic security of the underlying ElGamal cryptosystem. Furthermore, the distributed implementation guarantees that no group of less than n users is able to decrypt a submitted bit. As such, to compromise the privacy of a single user, all other $n-1$ users have to reveal their own bits, which is not a weakness of the protocol itself. Note that, to maintain privacy across dynamic populations, we may easily apply the concept of differential privacy [1], by having each user submit noisy measurements. Nevertheless such methods are orthogonal to this work.

V. Experimental Evaluation

We implemented the three homomorphic cryptosystems with the C programming language, using the GMP[1] multiple precision arithmetic library. For the BGN cryptosystem, we leveraged Ben Lynn's PBC library[2] that is also written on top of GMP. We tested the protocols on two different architectures, namely a 3.5 GHz Intel i7 CPU (x86_64) representing the server, and an Apple A8 CPU (arm64) representing the client. Note that, due to some porting problems with GMP's source code, we were unable to compile GMP with the assembly optimizations for the arm64 device. As a result, the client CPU times may be underestimated.

For security, we chose a 1024-bit RSA modulus for the Paillier and BGN cryptosystems, and a 160-bit ElGamal key. Table II summarizes the computational cost of the basic cryptographic primitives at the two different architectures. Missing values imply that the underlying operation is not required. The resulting ciphertext sizes are 256 bytes for Paillier and ElGamal, and 260 bytes for BGN.

In the following section, we measure the computational and communication costs of the various components of our methods, at both the client and server. Table III lists the parameters that control our experiments. In each experiment we vary a single parameter and keep the remaining ones to their default values (the ad size is fixed to 512 bytes). Note that, given the unique properties of our mobile advertisement framework, we do not compare against other, more computationally efficient approaches, because they do not provide the same level of security.

[1]https://gmplib.org/

[2]https://crypto.stanford.edu/pbc/

TABLE II
COST OF CRYPTOGRAPHIC PRIMITIVES AT CLIENT AND SERVER (ALL TIMES IN ms)

Crypto primitive	Server	Client
Paillier encryption	–	15.7
Paillier decryption	–	15.7
Paillier exponentiation (128-byte exponent)	1.2	–
Paillier multiplication	0.002	–
BGN encryption	–	9.8
BGN decryption	–	7.2
BGN exponentiation (3-byte exponent)	0.013	–
BGN multiplication	0.002	–
BGN bilinear map	5.5	–
ElGamal encryption	–	1.4
ElGamal exponentiation (20-byte exponent)	–	0.7
ElGamal multiplication	0.001	0.01

TABLE III
EXPERIMENTAL PARAMETERS

Parameter	Values	Default
Grid granularity (N)	50, 100, 200, 300	100
Number of ads ($\|\mathcal{A}\|$)	1K, 2K, 5K, 10K	2K
Number of users (n)	10K, 20K, 50K, 100K	20K
Max ads per location ($\|\mathcal{A}_l\|_{max}$)	20, 50, 100, 200	50

In the first experiment, we investigate the performance of the client's query generation algorithm for the two ad retrieval protocols. Fig. 1a shows the CPU time at the client (logarithmic scale) as a function of the grid granularity. For the default 100×100 grid, the Paillier-based scheme takes over 150 seconds of compute time, whereas BGN terminates in just 2 seconds. For finer grids, the cost of the basic protocol becomes prohibitive. This is due to the quadratic complexity of Algorithm 1 that generates one ciphertext for each of the N^2 cells. On the other hand, the BGN-based protocol is very efficient, requiring less than 7 seconds of compute time even for the 300×300 grid. Fig. 1b illustrates the communication cost for the same experiment. Paillier's quadratic cost is again evident, as it necessitates 0.6–23 MB of data transfer per query. Alternatively, the BGN-based scheme incurs less than 160 KB of communication cost under all settings.

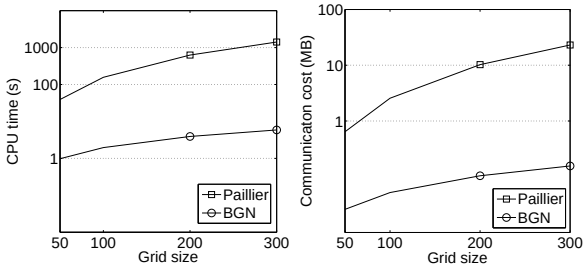
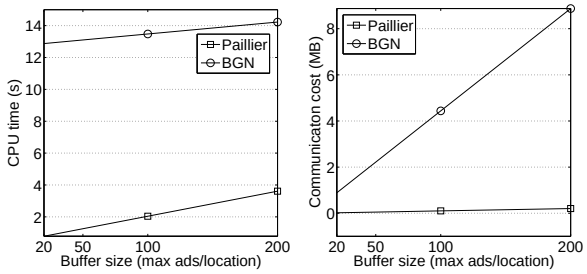


Fig. 1. Query generation cost at the client vs. grid size (a) CPU time (a) Communication cost

Staying at the client side, we measure the cost of the result extraction procedure (i.e., buffer decryption) as a function of the buffer size ($|\mathcal{A}_l|_{max}$). We assume that approximately 20% of the total ads in each location will match the client's profile. Therefore, approximately 80% of the ads entail a single decryption operation, while the rest invoke all m decryptions (as explained in Algorithm 3). Fig. 2a demonstrates the computational efficiency of the basic scheme, which is about 12 seconds faster than BGN for all buffer sizes. This is due to the discrete log nature of BGN that necessitates numerous ciphertexts to encrypt a single ad. In particular, for our default settings, $m = 171$ for BGN and $m = 4$ for Paillier. Even though the decryption operation is twice as fast with BGN (Table II), the sheer amount of operations needed negate this advantage. Fig. 2b shows the communication cost for downloading the encrypted buffer $\mathcal{B}$ from the server. Both methods scale linearly with the buffer size, but the basic protocol is clearly superior, due to its better m value.

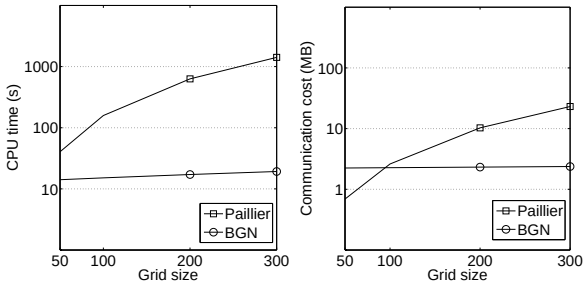


(a) (b)

Fig. 2. Result extraction cost at the client vs. buffer size (max ads/location) (a) CPU time (b) Communication cost

To get a concrete picture of the relative performance of the two ad delivery protocols, Fig. 3 shows, in logarithmic scale, the cumulative cost (query generation plus result extraction) at the client as a function of the grid size. In terms of CPU time (Fig. 3a), the BGN protocol is the clear winner in all settings. Even for a coarse 50×50 grid, BGN is three times faster than Paillier (41 vs. 14 seconds), with that gap growing fast as N increases. The only advantage of the basic scheme is in the cumulative communication cost (Fig. 3b), where it slightly outperforms BGN for coarse grids. Note that the communication cost of BGN remains almost constant at around 2.2 MB, i.e., the cost of downloading the encrypted buffer. Even though the basic scheme is clearly outperformed by its BGN counterpart, it could still be very useful in certain situations, given its efficiency in the result extraction phase. As explained in Section III-A, the bottleneck of the Paillier query generation algorithm is the computation of N^2 ciphertexts. However, these ciphertexts consist almost entirely of encryptions of 0, and are independent of the client's location. Therefore, it is not inconceivable to imagine a scenario where the client precomputes offline a large pool of ciphertexts that can be used in future queries.

We next shift our focus towards the server, and investigate its performance in the query processing phase of our protocol.



(a) (b)

Fig. 3. Cumulative cost at the client (query generation + result extraction) vs. grid size (a) CPU time (b) Communication cost

Fig. 4 illustrates the query processing time at the server as a function of the total number of ads $|\mathcal{A}|$. As expected, both methods scale linearly with $|\mathcal{A}|$, since the server must perform m modular exponentiations and multiplications for every ad in the system. Recall that the intuition behind the BGN protocol was to shift the computational burden from the mobile devices to the server. As such, the server must perform N^2 bilinear map computations before processing the actual ads (Algorithm 5). As shown in Table II, this is by far the most expensive cryptographic operation at the server and, for the 100×100 grid, this preprocessing alone takes 55 seconds. Nevertheless, the actual processing of the ads is faster with BGN, so the performance gap against Paillier closes when $|\mathcal{A}|$ increases.

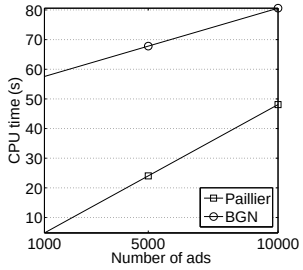
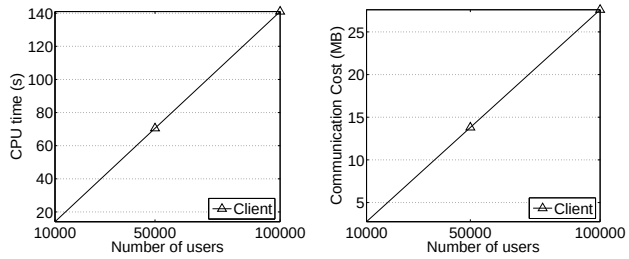


Fig. 4. CPU time at server vs. number of ads.

In the last set of experiments, we measure the costs associated with the privacy-preserving collection of ad impressions. We start by examining the cost of the distributed key generation algorithm. Fig. 5a illustrates the CPU time spent at the client as a function of the total number of users n. This cost is dominated by the modular exponentiations that are required to verify the commitments submitted by the remaining $n - 1$ users. As such, the cost grows linearly with n, ranging from 14 to 141 seconds. This is also true for the communication cost (Fig. 5b) that consists of the cost of downloading the users' commitments and public key shares (Algorithm 6). The communication cost is moderate, ranging from 2.7 to 27 MB.

There are two observations to be made here. First, the key generation algorithm is not invoked frequently (only when new users enter the system), and the ad network has the option to delay this process in order to perform batch insertions. In

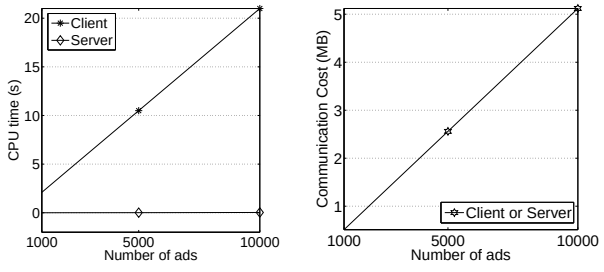


(a) (b)

Fig. 5. Key generation cost at the client vs. number of clients in the system (a) CPU time (b) Communication cost

addition, key generation can be performed at night, when the clients' devices are idle. The second observation is that the ad network could split users into multiple groups, in order to speed up the key generation process. That is, every group of users (say 10,000) would generate their own public key to use in the aggregation algorithm.

Finally, Fig. 6 shows the cost of the interactive aggregation process at both the client and server, as a function of the total number of ads $|\mathcal{A}|$. Clearly, both the CPU and communication costs are linear in $|\mathcal{A}|$, because the client has to submit one ciphertext (ad impression) for every ad in the system, regardless of whether that ad was displayed or not. In addition, the client is involved in the decryption process of $|\mathcal{A}|$ ciphertexts that contain the aggregated ad impressions by all users. The CPU time at the client (Fig. 6a) is dominated by the $|\mathcal{A}|$ encryption operations, each costing 1.4 ms. On the other hand, the decryption process necessitates just $|\mathcal{A}|$ modular exponentiations. At the server side, the computational cost per client is very low, as it entails $2 \cdot |\mathcal{A}|$ modular multiplications (with just 1 μs per operation). The communication cost (Figure. 6b) for the two parties is low, ranging from 0.5–5 MB. It consists of the interactive exchange of ciphertexts between the client and the server (Algorithm 7).



(a) (b)

Fig. 6. Aggregation cost vs. number of ads (for one client) (a) CPU time (b) Communication cost

VI. Conclusions

In this paper, we proposed the first location-aware mobile advertising framework that offers stringent privacy guarantees through cryptographic constructions. Unlike previous work, our methods guarantee that the ad network is oblivious to both the content sent to the clients, and the ad impressions submitted by the clients. Furthermore, we do not employ trusted third-parties and our protocols are secure against any number of colluding parties. We implemented the underlying cryptographic primitives on mobile devices and showed that our framework is practical for real world applications. Currently, our methods are secure against semi-honest adversaries. In the future, we want to address malicious users, particularly the case of click-fraud where users submit incorrect ad impressions to the server.

Acknowledgments

This research has been funded by the NSF CAREER Award IIS-0845262.

References

[1] G. Acs and C. Castelluccia. I have a dream! (differentially private smart metering). In *Information Hiding*, pages 118–132, 2011.
[2] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–341. 2005.
[3] CNN. *Google fires engineer for privacy breach*, 2010. http://www.cnn.com/2010/TECH/web/09/15/google.privacy.firing/index.html?hpt=T2.
[4] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18, 1985.
[5] Z. Erkin and G. Tsudik. Private computation of spatial and temporal power consumption with smart meters. In *ACNS*, pages 561–577, 2012.
[6] C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM STOC*, volume 9, pages 169–178, 2009.
[7] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *ICALP*, pages 803–815, 2005.
[8] S. Guha, B. Cheng, and P. Francis. Privad: Practical privacy in online advertising. In *NSDI*, 2011.
[9] H. Haddadi, P. Hui, and I. Brown. MobiAd: private and scalable mobile advertising. In *ACM MobiArch*, pages 33–38, 2010.
[10] M. Hardt and S. Nath. Privacy-aware personalization for mobile advertising. In *ACM CCS*, pages 662–673, 2012.
[11] M. Jawurek and F. Kerschbaum. Fault-tolerant privacy-preserving statistics. In *PETS*, pages 221–238, 2012.
[12] A. Juels. Targeted advertising... and privacy too. In *CT-RSA*, pages 408–424. 2001.
[13] T. Jung, X. Li, and M. Wan. Collusion-tolerable privacy-preserving sum and product calculation without secure channel. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 12(1):45–57, 2015.
[14] E. Kushilevitz and R. Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.
[15] Q. Li, G. Cao, and T. F. La Porta. Efficient and privacy-aware data aggregation in mobile sensing. *IEEE Transactions on Dependable and Secure Computing (TDSC)*, 11(2):115–129, 2014.
[16] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):59–98, 2009.
[17] R. Ostrovsky and W. E. Skeith III. Private searching on streaming data. In *CRYPTO*, pages 223–240, 2005.
[18] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
[19] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO*, pages 129–140, 1991.
[20] T. P. Pedersen. A threshold cryptosystem without a trusted party. In *EUROCRYPT*, pages 522–526, 1991.
[21] E. Shi, T.-H. H. Chan, E. G. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.
[22] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *NDSS*, 2010.
[23] J. Y. Tsai, P. G. Kelley, L. F. Cranor, and N. Sadeh. Location-sharing technologies: Privacy risks and controls. *ISJLP*, 6:119, 2010.
[24] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SDM*, pages 92–102, 2005.