Private Location Centric Profiles for GeoSocial Networks

Bogdan Carbunar  
FIU Miami,FL

Mahmudur Rahman  
FIU Miami,FL

Jaime Ballesteros  
FIU Miami,FL

Naphtali Rishe  
FIU Miami,FL

ABSTRACT

Providing input to targeted advertising, profiling social network users is an important source of revenue for geosocial networks. Since profiles contain personal information, their construction introduces a trade-off between user privacy and incentives of participation for businesses and geosocial network providers. In this paper we introduce location centric profiles (LCPs), aggregates built over the profiles of users present at a given location. We introduce ProfilR, a suite of mechanisms that construct LCPs in a private and correct manner. Our Android implementation shows that ProfilR is efficient: the end-to-end overhead is small even under strong correctness assurances.

1. INTRODUCTION

Online social networks have become a significant source of personal information. Facebook alone is used by more than 1 out of 8 people today. Social network users voluntarily reveal a wealth of personal data, including age, gender, contact information, preferences and status updates. A recent addition to this space, geosocial networks (GSNs) such as Yelp [1], Foursquare [2] or Facebook Places [3], further provide access even to personal locations, through check-ins performed by users at visited venues.

From the user perspective, personal information allows GSN providers to offer targeted advertising and venue owners to promote their business through spatio-temporal incentives (e.g., rewarding frequent customers through accumulated badges). The profitability of social network providers and participating businesses rests on their ability to collect, build and capitalize upon customer and venue profiles. Profiles are built based on user information – the more detailed the better. Providing personal information exposes however users to significant risks, as social networks have been shown to leak [4] and even sell [5] user data to third parties. Conversely, from the provider and business perspective, being denied access to user information discourages participation. There exists therefore a conflict between the needs of users and those of providers and participating businesses: Without privacy people may be reluctant to use geosocial networks, without feedback the provider and businesses have no incentive to participate.

In this paper we take first steps toward breaking this deadlock, by introducing the concept of location centric profiles (LCPs). LCPs are aggregate statistics built from the profiles of users that have visited a certain location.

We introduce ProfilR, a framework that allows the construction of LCPs based on the profiles of present users, while ensuring the privacy and correctness of participants. Informally, we define privacy as the inability of venues and the GSN provider to accurately learn user information, including even anonymized location trace profiles.

Correctness is a by-product of privacy: under the cover of privacy users may try to bias LCPs. We consider two correctness components (i) location correctness – users can only contribute to LCPs of venues where they are located and (ii) LCP correctness – users can modify LCPs only in a predefined manner. Location correctness is an issue of particular concern. The use of financial incentives by venues to reward frequent geosocial network customers, has generated a surge of fake check-ins [6]. Even with GPS verification mechanisms in place, committing location fraud has been largely simplified by the recent emergence of specialized applications for the most popular mobile eco-systems (LocationSpoofer [7] for iPhone and GPSCheat [8] for Android).

We propose a venue centric ProfilR. To relieve the GSN provider from a costly involvement in venue specific activities, ProfilR stores and builds LCPs at venues. Participating venue owners need to deploy an inexpensive device inside their business, allowing them to perform LCP related activities and verify the physical presence of participating users. ProfilR relies on (Benaloh’s) homomorphic cryptosystem and zero knowledge proofs to enable oblivious and provable correct LCP computations.

2. BACKGROUND AND MODEL

We model the geosocial network (GSN) after Yelp [1]. It consists of a provider, S, hosting the system along with information about registered venues, and serving a number of subscribers. To use the provider’s services, a client application needs to be downloaded and installed. Users register and receive initial service credentials, including a unique user id. We use the terms subscriber and user interchangeably to refer to users of the service and the term client to denote the software provided by the service and installed by users on their devices.
The provider supports a set of businesses or venues, with an associated geographic location (e.g., restaurants, yoga classes, towing companies, etc). Users are encouraged to write reviews for visited locations, as well as report their location, through check-ins at venues where they are present.

Participating venue owners need to install inexpensive equipment, present on most recent smartphones. An important assumption that we do not make, is that the equipment installed has Internet connectivity and is able to communicate directly with the GSN provider. Besides ensuring the portability of our approach (e.g., can be installed anywhere inside the venue) this also implies solely a one-time cost for the venue owner (no monthly fees).

2.1 Location Centric Profiles

Each user has a profile $P_U = \{u_1, u_2, ..., u_d\}$, consisting of values on $d$ dimensions (e.g., age, gender, home city, etc). Each dimension has a range, or a set of possible values. Given a set of users $U$ at location $L$, the location centric profile at $L$, denoted by $LCP(L)$ is the set $\{S_1, S_2, ..., S_d\}$, where $S_i$ denotes the aggregate statistics over the $i$-th dimension of profiles of users from $U$.

In the following, we focus on a single profile dimension, $D$. We assume $D$ takes values over a range $R$ that can be discretized into a finite set of sub-intervals (e.g., set of continuous disjoint intervals or discrete values). Then, given an integer $b$, chosen to be dimension specific, we divide $R$ into $b$ intervals/sets, $R_1, ..., R_b$. For instance, gender maps naturally to discrete values ($b = 2$), while age can be divided into disjoint sub-intervals, with a higher $b$ value. We define the aggregate statistics $S$ for dimension $D$ of $LCP(L)$ to consist of $b$ counters $c_1, ..., c_b$; $c_i$ records the number of users from $U$ whose profile value on dimension $D$ falls within range $R_i$, $i = 1..b$.

2.2 Solution Definition

We define a private LCP solution to be a set of functions, $PP(k) = \{Setup, Spotr, CheckIn, PubStats\}$. Setup is run by each venue where user statistics are collected, to generate parameters for user check-ins. To perform a check-in, a user first runs Spotr, to prove her physical presence at the venue. Spotr returns error if the verification fails, success otherwise. If $Spotr$ is successful, CheckIn is run between the user and the venue, and allows the collection of profile information from the user. Specifically, if the user’s profile value $v$ on dimension $D$ falls within the range $R_i$, the counter $c_i$ is incremented by 1. Finally, PubStats publishes collected LCPs.

Let $C_V$ be the set of counters defined at a venue $V$. Let $\hat{C}_V$ denote the set of $b$ sets of counters derived from $C_V$, such that each set in $\hat{C}_V$ has exactly one counter incremented over the set $C_V$.

3. PROFI$\mathcal{L}$

Let Spotr$_V$ denote the device installed at venue $V$. For each user profile dimension $D$, Spotr$_V$ stores a set of encrypted counters – one for each sub-range of $R$. Initially, and following each cycle of $k$ check-ins executed at venue $V$, Spotr$_V$ initiates Setup, to request the provider $S$ to generate a new Benaloh key pair. Thus, at each venue we divide time into cycles: a cycle completes once $k$ users have checked-in at the venue.

When a user $U$ checks-in at venue $V$, it first engages in the Spotr protocol with Spotr$_V$. This allows the venue to verify $U$’s physical presence through a challenge/response protocol between Spotr$_V$ and the user device. Furthermore, a successful run of Spotr provides $U$ with a share of the secret key employed in the Benaloh cryptosystem at the current cycle. For each venue and user profile dimension, $S$ stores a set $Sh$ of shares of the secret key that have been revealed so far.

Subsequently, $U$ runs CheckIn with Spotr$_V$, to first send its share of the secret key and to receive the encrypted counter sets. During CheckIn, for each dimension $D$, $U$ increments the counter corresponding to her range, re-encrypts all counters and sends the resulting set to Spotr$_V$. $U$ and Spotr$_V$ engage in a zero knowledge protocol that allows Spotr$_V$ to verify $U$’s correct behavior: exactly one counter has been incremented. Spotr$_V$ stores the latest, proved to be correct encrypted counter set, and inserts the secret key share into the set $Sh$. Once $k$ users successfully complete the CheckIn procedure, marking the end of a cycle, Spotr$_V$ runs PubStats to reconstruct the private key, decrypt all encrypted counters and publish the tally.

3.1 The Solution

Let $C_i$ denote the set of encrypted counters at $V$, following the $i$-th user run of CheckIn. $C_i = \{C_i[1], ..., C_i[b]\}$, where $C_i[j]$ denotes the encrypted counter corresponding to $R_j$, the $j$-th sub-range of $R$. We write $C_i[j] = E(u_j, u'_j, c_j, j) = [E(u_j, c_j), E(u'_j, j)]$, where $u_j$ and $u'_j$ are random obfuscating factors and $E(u, m)$ denotes the Benaloh encryption of message $m$ using random factor $u$. That is, an encrypted counter is stored for each sub-range of domain $R$ of dimension $D$. The encrypted counter consists of two records, encoding the number of users whose values on dimension $D$ fall within a particular sub-range of $R$.

Let $RE(v_j, v'_j, E(u_j, u'_j, c_j, j))$ denote the re-encryption of the $j$-th counter with two random values $v_j$ and $v'_j$: $RE(v_j, v'_j, E(u_j, u'_j, c_j, j)) = [RE(v_j, E(u_j, c_j)), \ RE(v'_j, E(u'_j, j))] = [E(u_j v'_j, c_j), E(u'_j v'_j, j)]$. Let $C_i[j] + + = E(u_j, u'_j, c_j + 1, j)$ denote the encryption of the incremented $j$-th counter. Note that incrementing the counter can be done without decrypting $C_i[j]$ or knowing the current counter’s value: $C_i[j] + + = [E(u_j, c_j + 1, u'_j, E(u'_j, j))] = [g^{y_j + 1} u'_j, E(u'_j, j)] = [E(u_j, c_j + 1), E(u'_j)]$.

In the following we use the above definitions to introduce PROFI$\mathcal{L}$. PROFI$\mathcal{L}$ instantiates $PP(k)$, where $k$ is the privacy parameter. The notation $P(A(params_A), B(params_B))$ denotes the fact that protocol $P$ involves participants $A$ and $B$, each with its own parameters.

Setup$(V(), S(k))$. The provider $S$ runs the key generation function $K(k)$ of the Benaloh cryptosystem [9]. Let $p$ and $q$ be the private key and $n$ and $y$ the public key. $S$ sends the public key to Spotr$_V$. Spotr$_V$ generates a signature key pair and registers the public key with $S$. For each user profile dimension $D$ of range $R$, Spotr$_V$ performs the following steps:

- Initialize counters $c_1, ..., c_b$ to 0. $b$ is the number of $R$’s sub-ranges.
- Generate $C_0 = \{E(x_1, x'_1, c_1, 1), ..., E(x_b, x'_b, c_b, b)\}$, where $x_i, x'_i, i = 1..b$ are randomly chosen values. Store $C_0$ indexed on dimension $D$.
- Initialize the share set $S_{key} = \emptyset$. 
Spoter(U(K),V(),S(p)): U sets up a connection with SpotrV using fresh, random MAC and IP address values. SpotrV initiates a challenge/response protocol, by sending to U the currently sampled time $T$, an expiration interval $\Delta T$ and a fresh random value $R$. The user’s device generates a hash of these values and sends the result back to SpotrV. SpotrV ensures that the response is received within a specific interval from the challenge (see Section 4 for values and discussion). If the verification succeeds, SpotrV uses its private key to sign a timestamped token and sends the result to U. U contacts S over Mix and sends the token signed by SpotrV. S verifies V’s signature as well as the freshness (and single use) of the token. Let U be the i-th user checking-in at V. If the verifications pass and $i \leq k$, S uses the $(k,n)$ TSS to compute a share of $p$ (Benaloh secret key, factor of the modulus n). Let $p_i$ be the share of $p$. S sends the (signed) share $p_i$ to U. If $i > k$, S calls Setup to generate new parameters for V.

CheckIn(U(p, n, V), V(n, y, C_i−1, Skey)): Executes only if the previous run of Spoter is successful. Let U be the i-th user checking-in at V. Then, $C_i−1$ is the current set of encrypted counters. SpotrV sends $C_i−1$ to U. Let $v, U$’s value on dimension $D$, be within $R$’s j-th sub-range, i.e., $v \in R$. U runs the following steps:

- Generate $b$ pairs of random values $\{(v_1, v'_1), ..., (v_b, v'_b)\}$. Compute the new encrypted counter set $C_i$, where the order of the counters in $C_i$ is identical to $C_i−1$: $C_i = \{RE(v_1, v'_1, C_i−1[l])| l = 1..b, l \neq j \} \cup RE(v_j, v'_j, C_i−1[j]+1)$.
- Send $C_i$ along with the (by S) share $p_i$ of the private key $p$ to V.

If SpotrV successfully verifies the signature of S on the share $p_i$, U and SpotrV engage in a zero knowledge protocol ZK-CTR (see Section 3.2). ZK-CTR allows U to prove that $C_i$ is a correct re-encryption of $C_i−1$: only one counter of $C_i−1$ has been incremented. If the proof verifies, SpotrV replaces $C_i−1$ with $C_i$ and adds the share $p_i$ to the set $S_{key}$.

PubStats(V(C_i, Sh, V), S(p,q)): SpotrV performs the following actions:

- If $|Sh| < k$, abort.
- If $|Sh| = k$, use the k shares to reconstruct p, the private Benaloh key.
- Use p and q = n/p to decrypt each record in $C_k$, the final counters at set of counters. Publish results.

3.2 ZK-CTR: Proof of Correctness

We now present the zero knowledge proof of the set $C_i$ being a correct re-encryption of the set $C_i−1$, i.e., a single counter has been incremented. Let ZK-CTR(l) denote the protocol run for sets $C_i−1$ and $C_i$. U and SpotrV run the following steps s times:

- $U$ generates random values $(t_1, t'_1), ..., (t_b, t'_b)$ and random permutation $\pi$, then sends to SpotrV the proof set $P_{i−1} = \pi(\{RE(t_i, t'_i, C_i−1[l])| l = 1..b\})$.
- $U$ generates random values $(w_1, w'_1), ..., (w_b, w'_b)$, then sends to SpotrV the proof set $P_i = \pi(\{RE(w_i, w'_i, C_i[l])| l = 1..b\})$.
- SpotrV generates a random bit $a$ and sends it to U.
- If $a = 0$, U reveals random values $(t_1, t'_1), ..., (t_b, t'_b)$ and $(w_1, w'_1), ..., (w_b, w'_b)$. SpotrV verifies that for each $l = 1..b$, $RE(t_i, t'_i, C_i−1[l])$ occurs in $P_{i−1}$ exactly once, and that for each $l = 1..b$, $RE(w_i, w'_i, C_i[l])$ occurs in $P_i$ exactly once.

4. EVALUATION

We have implemented ProfilR using Android. We used the standard Java security library to implement the cryptographic primitives employed by ProfilR. For secret sharing, we used Shamir’s scheme and for digital signatures we used RSA. We also used the kSOAP2 library to enable SOAP functionality on the Android app. We used the Google map API to facilitate the location based service enabled by our approach.

For testing purposes we have used Samsung Admire smartphones running Android OS Gingerbread 2.3 with a 800MHz CPU and a Dell laptop equipped with a 2.4GHz Intel Core i5 processor and 4GB of RAM for the server. For local connectivity the devices used their 802.11b/g Wi-Fi interfaces. All reported values are averages taken over at least 10 independent protocol runs.

We have first measured the overhead of the Setup operation. We set the number of ranges of the domain D to be 5, Shamir’s TSS group size to 1024 bits and RSA’s modulus size to 1024 bits. Figure 1 shows the Setup overhead on the smartphone and laptop platforms, when the Benaloh modulus size ranges from 64 to 2048 bits. Note that even a resource constrained smartphone takes only 2.2s for 1024 bit sizes (0.9s on a laptop). A marked increase can be noticed for the smartphone when the Benaloh bit size is 2048 bit long - 13.5s. We note however that this cost is amortized over multiple check-in runs.

We now focus on the most resource consuming compo-
storage overhead is only a fraction of the (single round) communication overhead, 2NBN. For a single dimension, with 20 sub-ranges, the overhead is 5KB.

Figure 2 shows also the storage overhead (at a venue). The venue component is 29ms and the client component is 106ms. Furthermore, Figure 4 shows the overheads of these components as a function of the number of ZK-CTR rounds.

The privacy is proved at "runtime": if the pollster leaks private data, it will be exposed probabilistically. Our work also allows entities to collect private user data, however, the collectors are never allowed direct access to private user data.

Toubiana et. al [11] proposed Adnostic, a privacy preserving ad targeting architecture. Users have a profile that allows the private matching of relevant ads. While PROFIL_R can be used to privately provide location centric targeted ads, its main goal is different - to compute location (venue) centric profiles that preserve the privacy of contributing users.

5. RELATED WORK

Golle et al. [10] proposed techniques allowing pollsters to collect user data while ensuring the privacy of the users. We further examine the communication overhead in terms of bits transferred during ZK-CTR between a client and a venue (SPOTrV ) computation overhead as well as their communication overhead. We set the number of sub-ranges of domain D to 5. We tested the client side running on the smartphone and the venue component executing on the laptop. Figure 3 shows the dependence of the three costs for a single round of ZK-CTR on the Benaloh modulus size. The second component of the sum is due to the average outcome of the challenge bit. Figure 2 shows the communication overhead in terms of bits transferred during ZK-CTR between a client and a venue. Let N be the Benaloh modulus size and B the subrange count of domain D. The communication overhead in a single ZK-CTR round is 4BN + 3BN = 7BN. The second component of the sum is due to the average outcome of the challenge bit. Figure 2 shows the dependency of the communication overhead (in KB) on B, when N = 1024. Even when B = 20, the communication overhead is around 17KB. Figure 2 shows also the storage overhead (at a venue). The storage overhead is only a fraction of the (single round) communication overhead, 2BN. For a single dimension, with 20 sub-ranges, the overhead is 5KB.

We show that our solution is efficient, even when running on resource constrained mobile devices.

6. CONCLUSIONS

In this paper we proposed novel mechanisms for building aggregate location-centric profiles while maintaining the privacy of participating users and ensuring their honesty during the process. We propose centralized variants of the solution.

7. ACKNOWLEDGMENTS

We thank Radu Sion, Rahul Potharaju and Ahmed Jawad for insightful discussions. This material is based in part upon work supported by the National Science Foundation under Grant Nos. CNS-0821345, CNS-1126619, HRD-0833093, IIP-0829576, CNS-1057661, IIS-1052625, CNS-0959985, OISE-1157372, IIP-1237818, IIP-1215201, IIP-1230661, IIP-1026265, IIP-1058606, IIS-1213026.

8. REFERENCES