Eat the Cake and Have It Too: Privacy Preserving Location Aggregates in Geosocial Networks

Bogdan Carbunar, Mahmudur Rahman, Jaime Ballesteros, Naphtali Rishe
School of Computing and Information Sciences
Florida International University
Miami, Florida 33199
Email: {carbunar, mrahm004, jball008, rishen}@cs.fiu.edu

Abstract—Geosocial networks are online social networks centered on the locations of subscribers and businesses. Providing input to targeted advertising, profiling social network users becomes an important source of revenue. Its natural reliance on personal information 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 PROFIL, a suite of mechanisms that construct LCPs in a private and correct manner. We introduce iSafe, a novel, context aware public safety application built on PROFIL. Our Android and browser plugin implementations show that PROFIL is efficient: the end-to-end overhead is small even under strong correctness assurances.

I. 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 (i) users that have visited a certain location or (ii) a set of co-located users.

We introduce PROFIL, 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. Thus, location privacy is an inherent PROFIL Requirement.

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 GPS Cheat [8] for Android).

We propose first a venue centric PROFIL. To relieve the GSN provider from a costly involvement in venue specific activities, PROFIL 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. We extend PROFIL with the notion of snapshot LCPs, built by user devices from the profiles of co-located users, communicated over ad hoc wireless connections. Snapshot LCPs are not bound to venues, but instead user devices can compute LCPs of neighbors at any location of interest. PROFIL relies on (Benaloh’s) homomorphic cryptosystem and zero knowledge proofs to enable oblivious and provable correct LCP computations.

We further introduce iSafe, a context aware safety application, that uses PROFIL to privately build safety LCPs. The constant population density increase, and the recent surge of natural and man-made disasters, riots and lootings, make safety aware applications of paramount importance. The goal of iSafe is to make users aware of the safety of their sur-
roundings while preserving the privacy of participants. Safety information can empower a suite of applications, including safe walking/evacuation directions and safety dependent mobile authentication.

We implemented iSafe and PROFIL$_R$ as mobile application and browser plugin components. Our experiments show that on a smartphone, with a client cheating probability of 1 in a million, the end-to-end overhead of an LCP update operation is 2.5s. We further rely on data collected from Yelp [1], a popular geosocial network, to build user and venue safety labels. The iSafe browser plugin introduces an overhead of under 1s for collecting and processing 500 Yelp reviews.

The paper is organized as follows. Section II describes the system and adversary model and defines the problem. Section III introduces PROFIL$_R$ and proves its privacy and correctness. Section IV introduces snapshot LCPs and presents the distributed, real-time variant of PROFIL$_R$. Section V introduces iSafe and its implementation. Section VI evaluates the performance of the proposed constructs. Section VII describes related work and Section VIII concludes.

II. 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 (e.g., a $25 Raspberry PI [9], a BeagleBoard or any Android smartphone). Such equipment can also be used for other tasks including detecting fake user check-ins [10] and preventing fake badges and incorrect rewards, and validating social network (e.g., Yelp [1]) reviews, thus eliminating fake negative reviews. The advantages provided by such solutions can motivate the small investment.

We have collected data from 16,199 venues throughout the U.S. Besides the name, location and type of venue, we have also collected all the reviews provided for these venues, for a total of 1,096,044 reviews. For each review we extracted the reviewer id, the date the review was written and the number of check-ins performed. Moreover, we have collected data from 10,031 Yelp users, including their id, location, number of friends and reviews, for a total of 646,017 reviews. Figure 1(a) shows the long-tail distribution of the number of reviews per venue, for the collected venues.

A. 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 $\mathcal{U}$ at location $L$, the location centric profile at $L$, denoted by $LCP(L)$ is the set $\{S_1, S_2, ..., S_j\}$, where $S_i$ denotes the aggregate statistics over the $i$-th dimension of profiles of users from $\mathcal{U}$.

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 $c_1, c_2, .., c_b$; $c_i$ records the number of users from $\mathcal{U}$ whose profile value on dimension $D$ falls within range $R_i, i = 1..b$.

Figure 1(b) illustrates an LCP dimension: the distribution of the (great-circle) distance in miles from a venue (“Ike’s Place” in San Francisco, CA) and the home cities of its (4000+) reviewers. Note that more than 3000 reviews were left by locals, information that can be used by the venue to better cater to its customers.

B. Private LCP Requirements

We define a private LCP solution to be a set of functions, $PP(k) = \{\text{Setup}, \text{Spoter}, \text{CheckIn}, \text{PubStats}\}$, see Figure 2. $\text{Setup}$ 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 $\text{Spoter}$, to prove her physical presence at the venue. $\text{Spoter}$ returns error if the verification fails, success otherwise. If $\text{Spoter}$ is successful, $\text{CheckIn}$ 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, $\text{PubStats}$ publishes collected LCPs.
users can be malicious, attempting to bias LCPs constructed at C. Attacker Model outputs a bit this step, CheckIn.

Check-In Indistinguishability (CI-IND): Let a challenger that controls a venue V. A running as a user U not present at V, has negligible probability to successfully complete Spoter at V.

LCP Correctness: Let A denote an adversary that controls the GSN provider and any number of users. Let C be a challenger that controls a venue V. Let C’ denote the set of counters at V before A runs CheckIn at V and let C’’ be the set of counters afterward. If C’’ ≠ C’, the CheckIn completes successfully with only negligible probability.

k-Privacy: Let A denote an adversary that controls any number of venues and let C denote a challenger controlling k users. C runs Spoter followed by CheckIn at a venue V controlled by A on behalf of i < k users. Let C’ denote the resulting counter set. For each j = 1..b, A outputs c’ j, its guess of the value of the j-th counter of C’. The advantage of A,Adv(A) = |Pr[C’[j] = c’ j] − 1/(i + 1)|, defined for each j = 1..b, is negligible.

Check-In Indistinguishability (CI-IND): Let a challenger C control two users U0 and U1 and let an adversary A control any number of venues. A generates randomly q bits, b1, ..., bq, and sends them to C. For each bit bi, i = 1..q, C runs Spoter followed by CheckIn on behalf of user Ubi. At the end of this step, C generates a random bit b and runs Spoter followed by CheckIn on behalf of Ubi at a venue not used before. A outputs a bit b’, its guess of b. The advantage of A,Adv(A) = |Pr[b’ = b] − 1/2| is negligible.

C. Attacker Model

We assume venue owners are malicious and will attempt to learn private information from subscribers. Clients installed by users can be malicious, attempting to bias LCPs constructed at target venues. We assume the GSN provider does not collude with venues, but will try to learn private user information.

D. Cryptographic Tools

Homomorphic Cryptosystems. We use the Benaloh cryptosystem [11], an extension of the Goldwasser-Micali [12]. It consists of three functions (K, E, D), defined as follows:

- K(k) - key generation: k, an odd integer, is the size of the input block. Select two large primes p and q such that k|(p−1) and gcd(k, (p−1)/k) = 1 and gcd(k, q−1) = 1. Let n = pq. Select y ∈ Z∗ n, such that y(p−1)(q−1)/k mod n ≠ 1. n and y are the public key and p and q are the private key.
- E(u, m): Encrypt message m ∈ Zk, using a randomly chosen value u ∈ Zk n. Output ymuk mod n.
- D(z): Decrypt ciphertext z. Let z = ymuk mod n. If z(p−1)(q−1)/r = 1, then return m = 0. Otherwise, for i = 1..k, compute s = y−i z mod n. If s = 1, return m = i.

Benaloh’s cryptosystem is additively homomorphic: E(u1, m1)E(u2, m2) = E(u1u2, m1 + m2). We further define the re-encryption function RE(v, E(u, m)) to be ymukvk = E(vw, m). Note that the re-encryption function can be invoked without knowledge of the message m. Furthermore, it is possible to show that two ciphertexts are the encryption of the same plaintext, without revealing the plaintext. That is, given E(u, m) and E(v, m), reveal w = u−1v. Then, E(v, m) = RE(w, E(u, m)).

Anonymizers. We use an anonymizer[13], [14], [15], [16] that (i) operates correctly – the output corresponds to a permutation of the input and (ii) provides privacy – an observer is unable to determine which input element corresponds to a given output element in any way better than guessing. In the following we denote the anonymizer by Mix.

Secret Sharing. Our constructions use a (k, n) threshold secret sharing (TSS) [17] solution. Given a value R, TSS generates n shares such that at least k shares are needed to reconstruct R. A (k, n)-TSS solution satisfies the property of hiding: An adversary (provided with access to a TSS oracle) controlling the choice of two values R0 and R1 and given less than k shares of R0, b ∈ {0, 1}, can guess the value of b with probability only negligibly higher than 1/2.

III. PROFILR

Let SPOTRV denote the device installed at venue V. For each user profile dimension D, SPOTRV stores a set of encrypted counters – one for each sub-range of D.

Solution overview: Initially, and following each cycle of k check-ins executed at venue V, SPOTRV initiates Setup, to request the provider S to generate a new Benaloh key pair. Thus, at each venue time is partitioned into cycles: a cycle completes once k users have checked-in at the venue. The communication during Setup takes place over an authenticated and secure channel (see Figure 2).

When a user U checks-in at venue V, it first engages in the Spoter protocol with SPOTRV. As shown in Figure 2, this step is performed over an anonymous channel, to preserve the user’s (location) privacy. Spoter allows the venue to
verify U’s physical presence through a challenge/response protocol between SpoterV and the user device. Furthermore, a successful run of Spoter provides U with a share of the secret key employed in the Benaloh cryptosystem of the current cycle. For each venue and user profile dimension, S stores a set $S_h$ of shares of the secret key that have been revealed so far.

Subsequently, U runs CheckIn with SpoterV, to send its share of the secret key and to receive the encrypted counter sets. As shown in Figure 2, the communication takes place over an anonymous channel to preserve U’s privacy. During CheckIn, for each dimension $D$, U increments the counter corresponding to her range, re-encrypts all counters and sends the resulting set to SpoterV. U and SpoterV engage in a zero knowledge protocol that allows SpoterV to verify U’s correct behavior: exactly one counter has been incremented. SpoterV stores the latest, proved to be correct encrypted counter set, and inserts the secret key share into the set $S_h$.

Once $k$ users successfully complete the CheckIn procedure, marking the end of a cycle, SpoterV runs PubStats to reconstruct the private key, decrypt all encrypted counters and publish the tally. The communication during PubStats takes place over an authenticated channel (see Figure 2).

A. 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 $RE(v_j, v'_j, E(u_j, u'_j, c_j, j), d)$ denote the re-encryption of the $j$-th record 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 decrementing $C_i[j]$ or knowing the current counter’s value: $C_i[j] + = [E(u_j, c_j), E(u'_j, j)] = \{y^c \cdot 1 + u'_j, E(u'_j, j)\} = [E(u_j, c_j + 1), E(u'_j, j)]$.

In the following we use the above definitions to introduce PROFIL-R. PROFIL-R 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(U, S)): The provider $S$ runs the key generation function $K(k)$ of the Benaloh cryptosystem (see Section II-D). Let $p$ and $q$ be the private key and $n$ and $y$ the public key. $S$ sends the public key to SpoterV. SpoterV generates a signature key pair and registers the public key with $S$. For each user profile dimension $D$ of range $R$, SpoterV 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(U, S(k))): U sets up an anonymous connection with SpoterV, e.g., by using fresh, random MAC and IP address values. SpoterV 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 SpoterV. SpoterV ensures that the response is received within a specific interval from the challenge (see Section VI for values and discussion). If the verification succeeds, SpoterV 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 SpoterV. 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 $n$).

CheckIn(U(p_i, n, V, V(n, y, C_i−1, S_{key}))): $U$ uses the same random MAC and IP addresses as in the previous Spoter run. 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. SpoterV sends $C_i−1$ to U. Let $v_i, U$’s value on dimension $D$, be within $R$’s $j$-th sub-range, i.e., $v_i \in R_j$. 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 $C_i−1$: $C_i = \{RE(v_1, v'_1, C_i−1[1])|l = 1..b, l \neq j\} \cup RE(v_j, v'_j, C_i−1[j] + 1)$.

- Send $C_i$ along with the signed (by $S$) share $p_i$ of the private key $p$ to V.

If SpoterV successfully verifies the signature of $S$ on the share $p_i$, $U$ and SpoterV engage in a zero knowledge protocol ZK-CTR (see Section III-B). 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, SpoterV replaces $C_i−1$ with $C_i$ and adds the share $p_i$ to the set $S_{key}$.

PubStats(V(C_k, Sh, V), S(p, q)): SpoterV 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 set of counters at V. Publish results.
B. 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(i) denote the protocol run for sets $C_{i-1}$ and $C_i$. $U$ and SPOTR$_V$ run the following steps $s$ times:

- $U$ generates random values $(t_1, t'_1), \ldots, (t_s, t'_s)$ and random permutation $\pi$, then sends to SPOTR$_V$ the proof set $P_{i-1} = \pi\{RE(t_i, t'_i, C_{i-1}[l]), l = 1..b\}$.
- $U$ generates random values $(w_1, w'_1), \ldots, (w_b, w'_b)$, then sends to SPOTR$_V$ the proof set $P_i = \pi\{RE(w_i, w'_i, C_i[l]), l = 1..b\}$.
- SPOTR$_V$ generates a random bit $a$ and sends it to $U$.
- If $a = 0$, $U$ reveals random values $(t_1, t'_1), \ldots, (t_b, t'_b)$ and $(w_1, w'_1), \ldots, (w_b, w'_b)$. SPOTR$_V$ 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 exactly.
- If $a = 1$, $U$ reveals $o_1 = v_1w'_1t'_1$ and $o'_1 = v'_1w't'_1$, for all $l = 1..b$ along with $j$, the position in $P_{i-1}$ and $P_i$ of the incremented counter. SPOTR$_V$ verifies that for all $l = 1..b, l \neq j$, $RE(o_l, o'_l, P_{i-1}[l]) = P_i[l]$ and $RE(o_j, o'_j, P_{i-1}[j]) = P_i[j]$.
- If any verification fails, SPOTR$_V$ aborts the protocol.

C. Preventing Illegal Votes

For simplicity of presentation, we have avoided the Sybil attack problem: participants that cheat through multiple accounts they control or by exploiting the anonymizer. For instance, a rogue venue owner, controlling $k$ Sybil user accounts or simulating $k$-1 check-ins, can use PROFIL$_L$ to reveal the profile of a real user. Conversely, a rogue user (including the venue) could bias the statistics built by the venue (and even deny service) by checking-in multiple times in a short interval.

Sybil detection techniques (see Section VII) can be used to control the number of fake, Sybil accounts. However, the use of the anonymizer prevents the provider and the use of the unique IP and MAC addresses prevents the venue from differentiating between interactions with the same or different accounts. In this section we propose a solution, that when used in conjunction with Sybil detection tools, mitigates this problem. The solution introduces a trade-off between privacy and security.

Specifically, we divide time into epochs (e.g., one day long). A user can check-in at any venue at most once per epoch. When active, once per epoch $e$, each user $U$ contacts the provider $S$ over an authenticated channel. $U$ and $S$ run a blind signature [18] protocol: $U$ obtains the signature of $S$ on a random value, $R_{U,e}$. $S$ does not sign more than one value for $U$ for any epoch. In runs of Spoter and CheckIn during epoch $e$, $U$ uses $R_{U,e}$ as its pseudonym (i.e., MAC and IP address). Venues can verify the validity of the pseudonym using $S$'s signature. A venue accepts a single CheckIn per epoch from any pseudonym, thus limiting the user's impact on the LCP.

D. Analysis

Given a set of encrypted counters $C$, let $\tilde{C}$ denote the set of re-encryptions of records of $C$, where only one record has its counter incremented. We introduce the following theorem.

**Theorem 1:** ZK-CTR(i) is a ZK proof of $C_i \in \tilde{C}_{i-1}$.

**Proof:** We need to prove completeness, soundness and zero-knowledge. For completeness, if $C_i \in \tilde{C}_{i-1}$, in each of the $s$ steps, $U$ succeeds to convince $S$, irrespective of the challenge bit $a$. If $a = 0$, $U$ can produce the random obfuscating values proving that the proof sets $P_{i-1}$ and $P_i$ are correctly generated from $C_{i-1}$ and $C_i$. If $a = 1$, $U$ can build the obfuscating factors proving that $P_i \in \tilde{P}_{i-1}$.

For soundness, we need to prove that if $C_i \notin \tilde{C}_{i-1}$, $U$ cannot convince $S$ unless with negligible probability. For simplicity reasons, we assume $C_i \notin \tilde{C}_{i-1}$ due to a single record in $C_i$ being "bad": $C_{i-1}[j] = E(u_j, v_j, c_j, j)$ and $C_i[j] = E(v_j, v'_j, c'_j, j')$. In any round of the ZK-CTR protocol, $U$ has two options for cheating. First, $U$ could count on the bit $a$ to come up 0. Then, $U$ builds $P_{i-1}[j] = E(u_j, t_j, v'_j, c'_j, j)$ and $P_j[j] = E(v_j, w_j, v'_j, w'_j, c'_j, j')$. If however $a = 1$, $U$ has to come up with a value $\alpha_j$, such that $RE(\alpha_j, E(u_j, c_j) = E(v_j, c'_j) \text{ or } RE(\alpha_j, E(u_j, c_j + 1) = E(v'_j, c'_j))$. In the first case, this means $\frac{y^{3\alpha} (u_j \alpha_j)^k}{y_{c'_j} v_{c'_j}^k \mod n}$. Without knowing $\alpha_j$'s factorization, $U$ cannot compute $k$'s inverse modulo $\phi(n)$ Then, the equation is satisfied only if $c'_j = c_j + z_k$, for an integer $z$.Note however that Benaloh’s cryptosystem only works for values in $\mathbb{Z}_n^*$, making this condition impossible to satisfy. The second case is similar. The second cheating option is to assume $a = 1$ and build $P_{i-1}[j]$ to be a re-encryption of $P_{i-1}[j]$. It is then straightforward to see that if $a = 0$, $U$ can only succeed in convincing $S$, if $c'_j = c_j + zk$, which we have shown is impossible for $z \neq 0$. Thus, in each round, $U$ can only cheat with probability $1/2$. Following $s$ rounds, this probability becomes $1/2^s$.

We now show that ZK-CTR conveys no knowledge to any verifier, even one that deviates arbitrarily from the protocol. We prove this by following the approach from [19], [20]. Specifically, let $S^*$ be an arbitrary, fixed, expected polynomial time ITM. We generate an expected polynomial time machine $M^*$ that, without being given access to the client, produces an output whose probability distribution is identical to the probability distribution of the output of $< C, S^* >$.

We now build $M^*$ that uses $S^*$ as a black box many times. Whenever $M^*$ invokes $S^*$, it places input $x = (L_0, L_1)$ on its input tape $IT_S$ and a fixed sequence of random bits on its random tape, $RT_S$. The input $x$ consists of $L_0 = C_0$ and $L_1 = C_1$. The content of the input communication tape for $S^*$, $CT_S$ will consist of tuples $(P_{2i}, P_{2i+1}, \pi_i)$, where $P_{2i}$ and $P_{2i+1}$ are sets and $\pi_i$ is a permutation. The output of $M^*$ consists of two tapes: the random-record tape $RT_M$ and the communication-record tape $CT_M$. $RT_M$ contains the prefix of the random bit string $r$ read by $S^*$. The machine $M^*$ works as follows (round $i$):

**Step 1** $M^*$ chooses a random bit $a \in \{0, 1\}$. If $a = 0$, $M^*$ picks a random permutation $\pi_i$, generates $t_i, t'_i, l = 1..b$ ran-
randomly and computes $P_{2i} = \pi_i \{ RE(t_t, t'_t, C_{i-1}[l]), l = 1..b \}$. It then generates random values $w_i, w'_i, l = 1..b$, randomly and computes the set $P_{2i+1} = \pi_i \{ RE(w_t, w'_t, C_i[l]), l = 1..b \}$. Note that $M^*$ does not need to know the counters to perform this operation. If $a = 1$, $M^*$ generates a random set $P_{2i}$, then generates random values $o_i, o'_i$, randomly, $l = 1..b$. It then generates a random $j \in 1..b$ and computes $P_{2i+1}$ such that for all $l = 1..b, l \neq j$, $RE(o_j, o'_j, P_{2i}[l]) = P_{2i+1}[j]$ and for the $j$-th position, $RE(o_j, o'_j, P_{2i}[j]) = P_{2i+1}[j]$. 

**Step 2** $M^*$ sets $b = S^*(x, r; P_0, P_1, \pi_0, \ldots, P_{2i-2}, P_{2i-1}, \pi_{i-1}, P_{2i}, P_{2i+1})$. That is, $b$ is the output of $S^*$ on input $x$ and random string $r$ after receiving $i - 1$ pairs $P_{2j}, P_{2j+1}, \pi_j, j = 1..i - 1$ and proof $P_{2i}, P_{2i+1}$ on its communication tape $CT_s$. We have the following three cases.

1. $a = b = 0$. $M^*$ can produce $t_t, t'_t, w_t, l = 1..b$ and $\pi_i$ to prove that $P_{2i} = \pi_i \{ RE(t_t, t'_t, C_{i-1}[l]), l = 1..b \}$ and $P_{2i+1} = \pi_i \{ RE(w_t, w'_t, C_i[l]), l = 1..b \}$. $M^*$ sets $b_i$, $b$ to $b$, appends the tuple $(P_{2i}, P_{2i+1}, \pi_i, b_i)$ to $CT_M$ and proceeds to the next round (i+1).

2. $a = b = 1$. $M^*$ can produce $o_i, o'_i$, $l = 1..b$, and index $j$ such that $RE(o_j, o'_j, P_{2i}[l]) = P_{2i+1}[l]$, $l = 1..b, l \neq j$ and $RE(o_j, o'_j, P_{2i}[j]) = P_{2i+1}[j]$. $M^*$ sets $b_i$, $b$ to $b$, appends the tuple $(P_{2i}, P_{2i+1}, \pi_i, b_i)$ to $CT_M$ and proceeds to the next round (i+1).

3. $a \neq b$. $M^*$ discards all the values of the current iteration and repeats the current round (Step 1 and 2).

If all rounds are completed, $M^*$ halts and outputs $(x, r', CT_M)$, where $r'$ is the prefix of the random bits $r$ scanned by $S^*$ on input $x$. We first prove that $M^*$ terminates in expected polynomial runtime and then that the output distribution of $M^*$ is the same as the output distribution of $S^*$ when interacting with the client, on input $(L_0, L_1)$.

**Lemma 1:** $M^*$ terminates in expected polynomial time.

**Proof:** Given $C_0$ and $C_1$, during the $i$-th round $P_{2i}$ and $P_{2i+1}$ are either built from $C_0$ and $C_1$ or from each other. During each run of round $i$, the bit $a$ is chosen independently. Then $P_{2i}$ and $P_{2i+1}$ are also chosen independently. This implies that the probability that $a = b$ is $1/2$ and the expected number of repetitions of round $i$ is $2$. $S^*$ is expected polynomial time, which implies that $M^*$ is also polynomial time.

**Lemma 2:** The probability distribution of $< C, S^* > (L_0, L_1) >$ and of $M^*(L_0, L_1)$ are identical.

**Proof:** The output of $< C, S^* > (L_0, L_1)$ and of $M^*(L_0, L_1)$ consists of a sequence of $t$ tuples of format $(P_{2i}, P_{2i+1}, \pi_i, b_i)$. Let $\Pi_{M^*}^{(x,r,i)}$ and $\Pi_{S^*}^{(x,r,i)}$ be the probability distributions of the first $i$ tuples output by $M^*$ and $< C, S^* >$. We need to show that for any fixed random input $r$, $\Pi_{M^*}^{(x,r,i)} = \Pi_{S^*}^{(x,r,i)}$. We prove this by induction. The base case, where $i = 0$, holds immediately. In the induction step we assume that $\Pi_{M^*}^{(x,r,i)} = \Pi_{S^*}^{(x,r,i)} = T(i)$. We need to prove that the $i + 1$ tuples in $\Pi_{M^*}^{(x,r,i+1)}$, denoted by $\Pi_{M^*}^{(x,r,i+1)}$, and in $\Pi_{S^*}^{(x,r,i+1)}$, denoted by $\Pi_{S^*}^{(x,r,i+1)}$, have the same distribution.

We show that $\Pi_{M^*}^{(x,r,i+1)}$ and $\Pi_{S^*}^{(x,r,i+1)}$ are uniform over the set $V = \{(P_{2i}, P_{2i+1}, \pi_i, b) | b = S^*(x, r, T(i)) \}$ and $(P_{2i} = p_i, RE(C_p), P_{2i+1} = p_i, RE(C_1), \text{if } b = 0 \} \lor \{ p_{2i+1}[l] = \text{RE}(P_{2i}[l]), l = 1..b, l \neq j, P_{2i+1}[j] = yRE(P_{2i}[j]), \text{if } b = 1\}$. For $\Pi_{M^*}^{(x,r,i+1)}$, this is the case by construction. If $\Pi_{M^*}^{(x,r,i+1)}$ has output, it is also uniformly distributed in $V$.

$M^*$ terminates in expected polynomial time and its output has the same distribution as the output of the interaction between $S^*$ and a client. Thus, the theorem follows.

We can now prove the following results.

**Theorem 2:** PROFIL$_R$ is $k$-private.

**Proof:** (Sketch) Following the definition from Section II-B, let us assume that the adversary $A$ has access to an encrypted counter set $C_i$ generated after $C$ has run $S$-spoter followed by $C$-checkIn on behalf of $i < k$ different users. The records of set $C_i$ are encrypted and $A$ has $i$ shares of the private key. For any $j = 1..b$, let $c'_j$ be $A$’s guess of the value of the $j$-th counter in $C_i$. If $Pr[C_j] = c'_j - 1/(k+1) = \epsilon$ is non-negligible we can use $A$ to construct an adversary $B$ that has $\epsilon$ advantage in the (i) semantic security game of Benaloh or in the (ii) hiding game of $(k, n)$ TSS. We start with the first reduction. $B$ generates two messages $M_0 = 0$ and $M_1 = 1$ and sends them to the challenger $C$. $C$ picks a bit $d \in_R \{0, 1\}$ and sends to $B$ the value $E(u, M_d)$. Where $u$ is random and $E$ denotes Benaloh’s encryption function. $B$ initiates a new game with $A$, with counters set to 0. $B$ runs $S$-spoter and $C$-checkIn acting as challenger with $A$. $B$ re-encrypts all counters from $A$, except the $j$-th one, which it replaces with $E(u, M_d)$. $B$ runs $ZK$-CTR with $A$ (used as a black box) a polynomial number of times until it succeeds. $A$ outputs its guess of the values of all counters. $B$ sends the guess for the $j$-th counter to $C$. The advantage of $B$ in this game comes entirely from the advantage provided by $A$.

For the second reduction, $B$ runs $C$-setup as the provider and obtains the secret key $p_0$ and $p_1$ (renamed from $p$ and $q$). $B$ sends $p_0$ and $p_1$ to the challenger $C$, as its choice of two random values. $C$ generates a random bit $a$, uses the $(k, n)$ TSS to generate $i < k$ shares of $p_a, sh_1, .., sh_i$, and sends them to $B$. $B$ generates a new random prime $q$ and picks randomly a bit $d$. Let the Benaloh modulus be $n = pq$. Then, acting as $i$ different users, $U_j, j = 1..i \in_R \{0, 1\}$ $B$ runs $S$-spoter with $S$ (which it also controls) to obtain $S$’s signature on $sh_d$. For each of the $i$ users, $B$ runs $C$-checkIn with $A$. At the end of the process, $A$ outputs its guess of the encrypted counters. If the guess is correct on more than $d/(j + 1)$ counters, $B$ sends $d$ to $C$ as its guess for $a$. Otherwise, it sends $d$. Thus, $B$'s advantage in the hiding game of TSS is equivalent to $A$’s advantage against PROFIL$_R$.

**Theorem 3:** PROFIL$_R$ ensures location correctness.

**Proof:** The user's location is verified in the $S$-spoter protocol. A single malicious user, not present at venue $V$, is unable to establish a connection with the device deployed at $V$, $S$-spoter. Thus, the user is unable to participate in the challenge/response protocol and receive at its completion a provider signed share of the Benaloh secret key. Without the share, the user is unable to initiate the $C$-checkIn protocol. Two (or more) attackers can launch wormhole attacks: one attacker present at $V$, acts a proxy and relays information between
SPOTR\textsubscript{V} and a remote attacker. This may allow the remote attacker to successfully run \textit{Spoter} and \textit{CheckIn} at \textit{V}. In Section VI we present experimental proof that \textit{Spoter} detects wormhole attacks.

**Theorem 4:** \textit{PROFIL\textsubscript{R}} is LCP correct.

**Proof:** (Sketch) A user \(U\) can alter the LCP of a venue \(V\) in two ways. First, during the ZK-CTR protocol, it modifies more than one counter or corrupts (at least ) one counter. The soundness property of ZK-CTR, proved in Theorem 1 shows this attack succeeds with probability \(1/2^k\). Second, it attempts to prevent \(V\) from decrypting the counter sets after \(k\) users have run CheckIn. This can be done by preventing SPOTR\textsubscript{V} from reconstructing the private Benaloh key. Key shares are however signed by the provider, allowing SPOTR\textsubscript{V} to detect invalid shares.

**Theorem 5:** \textit{PROFIL\textsubscript{R}} provides CI-IND.

**Proof:** (Sketch) Let \(\mathcal{A}\) be an adversary that has an \(\epsilon\) advantage in the CI-IND game. We assume the challenger does not run \textit{Spoter} and \textit{CheckIn} twice for the same (user, epoch) pair – otherwise the use of the signed pseudonyms provides an advantage to \(\mathcal{A}\). Note that if pseudonyms are not used, this requirement is not necessary. Moreover, no identifying information is sent by users during \textit{Spoter} and \textit{CheckIn}; the pseudonyms are \textit{blindly} signed by \(S\), and all communication with \(S\) takes places over \textit{Mix}.

IV. Snapshot LCP

We extend \textit{PROFIL\textsubscript{R}} to allow not only venues but also users to collect snapshot LCPs of other, co-located users. To achieve this, we take advantage of the ability of most modern mobile devices (e.g., smartphones, tablets) to setup ad hoc networks. Devices establish local connections with neighboring devices and privately compute the instantaneous aggregate LCP of their profiles.

A. Snapshot \textit{PROFIL\textsubscript{R}}

We assume a user \(U\) co-located with \(k\) other users \(U_1, \ldots, U_k\). \(U\) needs to generate the LCP of its profiles, without infrastructure, GSN provider or venue support. An additional difficulty then, is that participating users need assurances that their profiles will not be revealed to \(U\). However, one advantage of this setup is that location verification is not needed: \(U\) intrinsically determines co-location with \(U_1, \ldots, U_k\). Snapshot \textit{PROFIL\textsubscript{R}} consists of three protocols, \{\textit{Setup, LCPGen, PubStats}\}:

**Setup**\((U(r), U_1, \ldots, U_k())\): \(U\) performs the following steps:

- Run the key generation function \(K(r)\) of the Benaloh cryptosystem (see Section II-D). Send the public key \(n\) and \(y\) to each user \(U_1, \ldots, U_k\).
- Engage in a multi-party secure function evaluation protocol \cite{21, 22} with \(U_1, \ldots, U_k\) to generate shares of a public value \(R < n\). At the end of the protocol, each user \(U_i\) has a share \(R_i\), such that \(R_1 \cdots R_k = R \mod n\) and \(R_i\) is only known to \(U_i\).

\(\rightarrow\)

![Fig. 3. Static crime indexes computed over crimes reported during 2010 in the Miami-Dade county.](image)

Fig. 3. Static crime indexes computed over crimes reported during 2010 in the Miami-Dade county.

- Assign each of the \(k\) users a unique label between 1 and \(k\). Let \(U_1, \ldots, U_k\) denote this order.
- Generate \(C_0 = \{E(x_1, x_1', 0, 1), \ldots, E(x_b, x_b', 0, b)\}\), where \(x_1, x_1'\), \(i = 1..b\) are randomly chosen. Store \(C_0\) indexed on dimension \(D\).

Each of the \(k\) users engages in a 1-on-1 \textit{LCPGen} with \(U\) to privately and correctly contribute her profile to \(U\)’s LCP.

\(\textit{LCPGen}(U(C_{i-1}), U_i())\): \(U\) repeats \(C_{i-1}\) be the encrypted counters after \(U_1, \ldots, U_{i-1}\) have completed the protocol with \(U\). \(U\) sends \(C_{i-1}\) to \(U_i\); \(U_i\) runs the following:

- Generate random values \((v_1, v_1'), \ldots, (v_b, v_b')\). Let \(j\) be the index of the range where \(U_i\) fits on dimension \(D\).
- Compute the new encrypted counter set \(C_i\) as: \(C_i = \{RE(v_1, v_1', C_{i-1}[l]) \mod n | l = 1..b, l \neq j\} \cup RE(v_j, v_j', C_{i-1}[j] + +)R_i \mod n\) and send it to \(U\).
- Engage in a ZK-CTR protocol to prove that \(C_i \in C_{i-1}\).

The only modification to the ZK-CTR protocol is that all re-encrypted values are also multiplied with \(R_i \mod n\), \(U_i\)’s share of the public value \(R\). If the proof verifies, \(U\) replaces \(C_{i-1}\) with \(C_i\).

After completing \textit{LCPGen} with \(U_1, \ldots, U_k\), \(U\)’s encrypted counter set is \(C_k = \{E_j = E(u_j, u_j', c_j, j)| R_1 \cdots R_k| j = 1..d\}\), where \(u_j\) and \(u_j'\) are the product of the obfuscation factors used by \(U_1, \ldots, U_k\) in their re-encryptions. The following protocol enables \(U\) to retrieve the snapshot LCP.

\(\textit{PubStats}(U(C_k))\): Compute \(E_j K\), \(\forall j = 1..d\), where \(K = R^{-1} \mod n\) \((R = R_1 \cdots R_k)\), decrypt the outcome using the private key \((p, q)\) and publish the resulting counter value. Even though \(U\) has the private key allowing it to decrypt any Benaloh ciphertext, the use of the secret \(R_i\) values prevents it from learning the profile of \(U_i\), \(i = 1..k\).

V. iSAFE: CONTEXT AWARE SAFETY

We introduce iSafe, an application built on \textit{PROFIL\textsubscript{R}}. iSafe uses the context of users, in terms of their location, time, other people present, to build a \textit{safety} representation. Quantifying the safety of a user based on her current context
can be further used to provide safe walking directions and context-aware smartphone authentication protocols (i.e., more complex authentication protocols in unsafe locations). iSafe combines information collected from Yelp with Census [23] and historical crime databases as well as context collected by the users’ mobile devices. We have access to the Miami-Dade county [24] area crime and Census datasets since 2007. Each record in the crime dataset is labeled with a crime type (e.g., homicide, larceny, robbery) as well as the geographic location and time of occurrence.

iSafe assigns static safety labels to Census-defined geographic blocks. While beyond the scope here, we note that the safety index is inversely proportional to the weighted average of the crimes committed in the block. Figure 3 shows the color-coded safety index for each block group in the Miami-Dade county (FL) in 2010. iSafe uses the static block safety indexes to compute safety labels of mobile users. The safety label of a user is an average over the safety indexes of the blocks visited by the user. Blocks visited more frequently, have an inherently higher impact on the user’s safety label. Block and user safety labels take values in the $[0, 1]$ interval; 1 is the safest label.

iSafe uses PROFIL$_R$ to privately compute the safety labels for Yelp venues: the distribution of safety indexes of users that reviewed them. To achieve this, iSafe divides the $[0, 1]$ safety range into a discrete set of disjoint sub-intervals, and assigns a counter to each sub-interval. Each venue privately retrieves the distribution of the safety values of its reviewers (the counters of users fitting the corresponding sub-intervals). Finally, the safety index of the venue is the weighted average of the aggregated counts. The normalized weights are either the upper bound value or the middle point of their corresponding sub-intervals.

Besides this venue-centric approach, iSafe also uses snapshot PROFIL$_R$ to privately aggregate the safety labels of colocated user devices and distributively obtain the real-time image of the safety of their location.

A. Implementation

We implemented iSafe as a (i) web server, (ii) a browser plugin running in the user’s browser and (iii) a mobile application. We use Apache Tomcat 6.0.35 to route requests (exposed to the client through a REST API interface) to our server-side component. The server-side component relies on the latest servlet v3.0 which offers additional features including asynchronous support, making the server-side processing much more efficient. We implemented the browser plugin for the Chrome browser using HTML, CSS and Javascript. The plugin interacts with Yelp pages and the web server, using content scripts (Chrome specific components that let us access the browser’s native API) and cross-origin XMLHttpRequests.

The browser plugin becomes active when the user navigates to a Yelp page. For user and venue pages, the plugin parses their HTML file and retrieves their reviews. We employ a stateful approach, where the server’s DB stores all reviews of pages previously accessed by users. This enables significant time savings, as the plugin needs to send to the web server only reviews written after the date of the last user’s access to the page. Given the venue’s set of reviews, the server determines the corresponding reviewers. Since we do not have access to the location history of users, to compute a user’s security label we rely on the venues reviewed by the user: The user safety is computed as an average over the safety labels of the blocks containing the venues reviewed by the user. Given the safety labels of reviewers, we run PROFIL$_R$ to determine their distribution and identify the safety level of the venue. The server sends back the safety level of the venue, which the plugin displays in the browser. Figure 4 shows iSafe’s extension to the Yelp page of the venue “Top Value Trading Inc.” in Hialeah, FL (central left yellow rectangle containing iSafe’s safety recommendations).

We have also implemented an Android front-end for iSafe’s snapshot LCPs. We used the standard Java security library to implement the cryptographic primitives employed by PROFIL$_R$. 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. Figure 5 shows a snapshot of the iSafe Android app on a Samsung Admire smartphone. We used the Google map API to facilitate the location based service employed by our approach.

VI. Evaluation

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.

iSafe: Figure 6(a) shows the overhead of the iSafe plugin when collecting the reviews of a venue browsed by the user, as a function of the number of reviews the venue has. It includes the cost to request each review page, parse and process the data for transfer. The experiments were performed on the Dell laptop. It exhibits a sub-linear dependence on the number of reviews of the venue (under 1s for 10 reviews but under 30s for 4000 reviews), showing that Yelp’s delay for successive requests decreases. While even for 500 reviews the overhead...
is less than 5s, we note that this cost is incurred only once per venue. Subsequent accesses to the same venue, by any other user will no longer incur this overhead.

**Spoter’s wormhole defenses:** Wormhole attacks are best detected through timing analysis. We have tested Spoter using a smartphone connected over ad hoc Wi-Fi to the laptop. The round-trip Wi-Fi latency is under 3ms. On the Android device, the time required to compute a (SHA-512) hash is 0.6ms. The overhead imposed by Spoter on a wormhole attack is the Wi-Fi round-trip latency, plus the hash time (0.003ms on the Chicago). In total, 19ms (we tested with systems in Miami, San Francisco and two attackers, if performed over the wired network, is at least latency. The one-way communication overhead between the smartphone and the venue component executing on the laptop. Figure 7(a) shows the dependence of the three costs imposed by a (SHA-512) hash is 0.6ms. The round-trip Wi-Fi latency is under 3ms. On the Android device, the time required to compute a (SHA-512) hash is 0.6ms. The overhead imposed by Spoter on a wormhole attack is the Wi-Fi round-trip latency, plus the hash time (0.003ms on the Chicago). In total, Spoter imposes an overhead on a wormhole attack (43ms) that is almost 12 times the overhead imposed on an honest user (3.6ms). Thus, wormhole attacks are easily detectable in Spoter.

**A. PROFIL<sub>R</sub> Evaluation**

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 6(b) 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 component of PROFIL<sub>R</sub>: the ZK-CTR protocol. We measure the client and venue (\textsc{SpotR}_\text{V}) 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 7(a) shows the dependence of the three costs for a single round of ZK-CTR on the Benaloh modulus size. Given the more efficient venue component and the superior computation capabilities of the laptop, the venue component has a much smaller overhead. The communication overhead is the smallest, exhibiting a linear increase with bit size. For a Benaloh key size of 1024 bits, the average end-to-end overhead of a single ZK-CTR round is 135ms. The venue component is 29ms and the client component is 106ms. Furthermore, Figure 7(b) shows the overheads of these components as a function of the number of ZK-CTR rounds, when the Benaloh key size is 1024 bit long. For 30 rounds, when a cheating client’s probability of success is \( 2^{-30} \), the total overhead is 3.6s.

We further examine 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 sub-range 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 6(c) 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 6(c) 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.

**VII. RELATED WORK**

Golle et al. [25] proposed techniques allowing pollsters to collect user data while ensuring the privacy of the users. The privacy is proved at “runtime”: if the pollster leaks private data, it will be exposed probabilistically. Our work also allow entities to collect private user data, however, the collectors are never allowed direct access to private user data.

Toubiana et al. [26] proposed Adnostic, a privacy preserving ad targeting architecture. Users have a profile that allows 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.

Manweiler et al. [27] proposed SMILE, a privacy-preserving “missed-connections” service similar to Craigslist, where the
service provider is untrusted and users do not have existing relationships. The solution is distributed, allowing users to anonymously prove to each other the existence of a past encounter. While we have a similar setup, our work addresses a different problem, of privately collecting location centric user profile aggregates.

Location and temporal cloaking techniques, or introducing errors in reported locations in order to provide 1-out-of-$k$ anonymity have been initially proposed in [28], followed by a significant body of work [29], [30], [31], [32]. We note that PROFIL$_R$ provides an orthogonal notion of $k$-anonymity: instead of reporting intervals containing $k$ other users, we allow the construction of location centric profiles only when $k$ users have reported their location. Computed LCPs hide the profiles the users: user profiles are anonymous, only aggregates are available for inspection, and interactions with venues and the provider are indistinguishable.

Our work relies on the assumption that participants cannot control a large number of fake, Sybil accounts. One way to ensure this property is to use existing Sybil detection techniques. Danezis and Mittal [33] proposed a centralized SybilInfer solution based in Bayesian inference. Yu et al. proposed distributed solutions, SybilGuard [34] and SybilLimit [35], that use online social networks to protect peer-to-peer network against Sybil nodes. They rely on the fast mixing property of social networks to collect certain user statistics. We have shown that our solutions are efficient, even when executed on resource constrained mobile devices.

**REFERENCES**


Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. J. ACM, 38(3), 1991.


