PROFIL$_R$: Toward Preserving Privacy and Functionality in Geosocial Networks

Bogdan Carbunar, Mahmudur Rahman, Jaime Ballesteros, Naphtali Rishe, and Athanasios V. Vasilakos

Abstract—**Profit is the main participation incentive for social network providers. Its reliance on user profiles, built from a wealth of voluntarily revealed personal information, exposes users to a variety of privacy vulnerabilities. In this paper, we propose to take first steps toward addressing the conflict between profit and privacy in geosocial networks. We introduce PROFIL$_R$, a framework for constructing location centric profiles (LCPs), aggregates built over the profiles of users that have visited discrete locations (i.e., venues). PROFIL$_R$ endows users with strong privacy guarantees and providers with correctness assurances. In addition to a venue centric approach, we propose a decentralized solution for computing real time LCP snapshots over the profiles of colocated users. An Android implementation shows that PROFIL$_R$ is efficient; the end-to-end overhead is small even under strong privacy and correctness assurances.**

Index Terms—**Social implications of technology, technology social factors, privacy.**

I. INTRODUCTION

ONLINE social networks have become a significant source of personal information. Their 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] and Foursquare [2] further collect fine grained location information, through *check-ins* performed by users at visited venues.

Overtly, personal information allows GSN providers to offer a variety of applications, including personalized recommendations and targeted advertising, and venue owners to promote their businesses through spatio-temporal incentives, e.g., rewarding frequent customers through accumulated badges. Providing personal information exposes however users to significant risks, as social networks have been shown to leak [3] and even sell [4] user data to third parties. There exists therefore a conflict. Without privacy people may be reluctant to use geosocial networks; without user information the provider and venues cannot support applications and have no incentive to participate.

In this paper, we take first steps toward addressing this conflict. Our approach is based on the concept of *location centric profiles* (LCPs). LCPs are statistics built from the profiles of (i) users that have visited a certain location or (ii) a set of co-located users.

Contributions. We introduce PROFIL$_R$, 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. Verifying the correctness of user data is necessary to compensate for this privacy constraint: users may cheat and bias LCPs anonymously. We consider two user correctness components. First, location correctness, where users should only contribute to LCPs of venues where they are located. This requirement is imposed by the recent surge of fake check-ins [5], motivated by their use of financial incentives. Second, LCP correctness, where users should be able to modify LCPs only in a predefined manner.

First, we propose a venue centric PROFIL$_R$, that relieves the GSN provider from a costly involvement in venue specific activities. To achieve this, PROFIL$_R$ stores and builds LCPs at venues. Furthermore, it relies on Benaloh's homomorphic cryptosystem and zero knowledge proofs to enable oblivious and provable correct LCP computations. We prove that PROFIL$_R$satisfies the introduced correctness and privacy properties.

Second, we propose a completely decentralized PROFIL$_R$extension, built around the notion of *snapshot* LCPs. The distributed PROFIL$_R$ enables user devices to aggregate the profiles of co-located users, without assistance from a venue device. Snapshot LCPs are not bound to venues, but instead user devices can compute LCPs of neighbors at any location of interest. Communications in both PROFIL$_R$ implementations are performed over ad hoc wireless connections. The contributions of this paper are then the following:

- Introduce the problem of computing location centric profiles (LCPs) while simultaneously ensuring the privacy and correctness of participants.
- Propose PROFIL$_R$, a framework for computing LCPs. Devise both a venue centric and a decentralized solution. Prove that PROFIL$_R$ satisfies the proposed privacy and correctness properties.

Manuscript received September 18, 2013; revised November 25, 2013 and January 16, 2014; accepted February 19, 2014. Date of publication February 24, 2014. This work was supported in part by DoD under Grant W911NF-13-1-0142 and in part by NSF under Grants IIP-1338922, IIP-1237818, IIP-1330943, and IIS-1213026. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Shantanu D. Rane.

B. Carbunar, M. Rahman, J. Ballesteros, and N. Rishe are with the School of Computing and Information Sciences, Florida International University, Miami, FL 33199 USA (e-mail: carbunar@gmail.com; mrahm004@cs.fiu.edu; jball008@cs.fiu.edu; rishen@cs.fiu.edu).

A. V. Vasilakos is with the University of Western Macedonia, 50100 Kozani, Greece (e-mail: vasilako@ath.forthnet.gr)

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2014.2307697

- Provide two applications for PROFIL$_R$: (i) privacy preserving, personalized public safety recommendations and (ii) privately building real time statistics over the profiles of venue patrons with Yelp accounts.
- Evaluate PROFIL$_R$ through an Android implementation. Show that PROFIL$_R$ is efficient even when deployed on previous generation smartphones.

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 the notion of snapshot LCPs and presents a distributed, real-time variant of PROFIL$_R$. Section V describes two PROFIL$_R$ applications. Section VI evaluates the performance of the proposed constructs. Section VII describes related work and Section VIII concludes.

II. Model and Background

We consider a core functionality that is supported by the most influential geosocial network (GSN) providers, Yelp [1] and Foursquare [2]. This functionality is simple and general enough to be applicable to most other GSNs (e.g., Facebook Places, Google Latitude). In this model, a provider S hosts the system, along with information about registered venues, and serving a number of users. To use the provider's services, a client application, the "client", needs to be downloaded and installed. Users register and receive initial service credentials, including a unique user id.

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 report their location, through *check-ins* at venues where they are present. During a check-in operation, performed upon an explicit user action, the user's device retrieves its GPS coordinates, reports them to the server, who then returns a list of nearby venues. The device displays the venues and the user needs to choose one as her current check-in location.

Participating venue owners need to install inexpensive equipment (e.g., a \$25 Raspberry PI [6], a BeagleBoard [7] or any Android smartphone). This equipment can be installed and used for other purposes as well, including detecting fake user check-ins [8] preventing fake badges and incorrect rewards, and validating social network (e.g., Yelp [1]) reviews. Venue deployed equipment provides a necessary ingredient: ground truth information from remote locations.

A. Location Centric Profiles

Each user has a profile $P_U = \{p_{U_1}, p_{U_2}, .., p_{U_d}\}$, 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 $\{LCP_1, LCP_2, .., LCP_d\}$, where LCP_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.

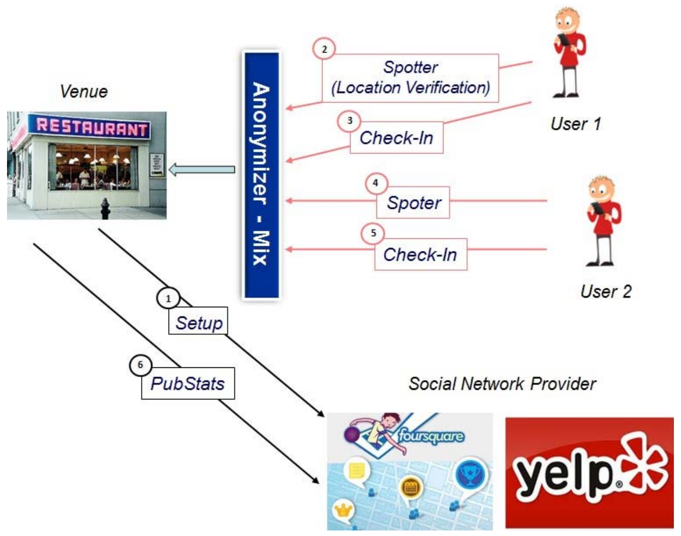

Fig. 1. Solution architecture ($k = 2$). The red arrows denote anonymous communication channels, whereas black arrows indicate authenticated (and secure) communication channels.

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 $\mathcal{U}$ whose profile value on dimension D falls within range R_i, $i = 1..b$.

B. Private LCP Requirements

Let k be a security parameter, denoting the level of privacy we need to provide for users at any location. We then define a private LCP solution to be a set of functions, $PP(k) = \{Setup, Spotter, CheckIn, PubStats\}$, see Fig. 1. $Setup$ 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 $Spotter$, to prove her physical presence at the venue. $Spotter$ returns error if the verification fails, success otherwise. If $Spotter$ 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. In the following, we use the notation $Prot(P_1(args_1), .., P_n(args_n))$ to denote protocol $Prot$ run between participants $P_1, .., P_n$, each with its own arguments.

Let C_V be the set of counters defined at a venue V. We use $\bar{C}_V$ to denote the set of sets derived from C_V as follows. Each set in $\bar{C}_V$ differs from C_V in exactly one counter, whose value increments the value of the corresponding counter in C_V. For instance, if $C_V = \{2, 5, 9\}$, then $\bar{C}_V = \{\{3, 5, 9\}, \{2, 6, 9\}, \{2, 5, 10\}\}$. A private LCP solution needs to satisfy the following properties:

k-Privacy: Let $\mathcal{A}$ denote an adversary that controls any number of venues and let $\mathcal{C}$ denote a challenger controlling k users. $\mathcal{C}$ runs $Spotter$ followed by $CheckIn$ at a venue V

controlled by $\mathcal{A}$ on behalf of $i < k$ users. Let C_i denote the resulting counter set. For each $j = 1..b$, $\mathcal{A}$ outputs c'_j, its guess of the value of the j-th counter of C_i. The advantage of $\mathcal{A}$, $Adv(\mathcal{A}) = |Pr[C_i[j] = c'_j] - 1/(i+1)|$, defined for each $j = 1..b$, is negligible.

Location Correctness: Let $\mathcal{A}$ denote an adversary that controls the GSN provider and any number of users. Let $\mathcal{C}$ be a challenger that controls a venue V. $\mathcal{A}$ running as a user U not present at V, has negligible probability to successfully complete *Spotter* at V.

LCP Correctness: Let $\mathcal{A}$ denote an adversary that controls the GSN provider and any number of users. Let $\mathcal{C}$ be a challenger that controls a venue V. Let C_V denote the set of counters at V before $\mathcal{A}$ runs *CheckIn* at V and let C'_V be the set of counters afterward. If $C'_V \notin \bar{C}_V$, the *CheckIn* completes successfully with only negligible probability.

Check-In Indistinguishability (CI-IND): Let a challenger $\mathcal{C}$ control two users U_0 and U_1 and let an adversary $\mathcal{A}$ control any number of venues. $\mathcal{A}$ generates randomly q bits, $b_1, .., b_q$, and sends them to $\mathcal{C}$. For each bit b_i, $i = 1..q$, $\mathcal{C}$ runs *Spotter* followed by *CheckIn* on behalf of user U_{b_i}. At the end of this step, $\mathcal{C}$ generates a random bit b and runs *Spotter* followed by *CheckIn* on behalf of U_b at a venue not used before. $\mathcal{A}$ outputs a bit b', its guess of b. The advantage of $\mathcal{A}$, $Adv(\mathcal{A}) = |Pr[b' = b] - 1/2|$ is negligible.

C. Attacker Model

We assume venue owners are malicious and will attempt to learn private information from their patrons. 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. Tools

Homomorphic Cryptosystems. We use the Benaloh cryptosystem [9], an extension of the Goldwasser-Micali [10]. It consists of three functions (KG, E, D), defined as follows:

- $KG(l)$ (Key Generation): l, an odd integer, is a system parameter, known to all participants, that denotes the size of the input block. Select two large primes p and q such that $l|(p-1)$ and $gcd(l, (p-1)/l) = 1$ and $gcd(l, q-1) = 1$. Let $n = pq$. Select $y \in \mathbb{Z}_n^*$, such that $y^{(p-1)(q-1)/l} \ mod\ n \neq 1$. n and y are the public key and p and q are the private key.
- $E(u, m)$: Encrypt message $m \in \mathbb{Z}_l^*$, using a randomly chosen value $u \in \mathbb{Z}_n^*$. Output $y^m u^l \ mod\ n$.
- $D(z)$: Decrypt ciphertext z. Let $z = y^m u^l \ mod\ n$. If $z^{(p-1)(q-1)/l} = 1$, then return $m = 0$. Otherwise, for $i = 1..l$, compute $s_i = y^{-i} z \ mod\ n$. If $s_i = 1$, return $m = i$.

Benaloh's cryptosystem is additively homomorphic: $E(u_1, m_1)E(u_2, m_2) = E(u_1u_2, m_1 + m_2)$. We further define the *re-encryption* function $RE(v, E(u, m))$ to be $y^m u^l v^l = E(uv, m)$. 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 $w = u^{-1}v$. Then, $E(v, m) = RE(w, E(u, m))$.

The above properties are ideal to enable a user to (i) increment the counter of a bucket even without knowing the counter's value or the encryption key and (ii) to re-encrypt all counters without knowing the encryption key.

Anonymizers. We use an anonymizer [11]–[13] 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. We use Orbot [14], an Android implementation of Tor [13].

Location Verification. We use one of the protocols proposed in [8] to verify the location claims of users checking-in. For completeness, we now briefly describe this protocol. Let SPOTR$_V$ denote the device installed at venue V. When a user U expresses interest to check-in at venue V, SPOTR$_V$ initiates a challenge/response protocol. It sends to U the currently sampled time T, an expiration interval ΔT and a fresh random value R. U's device generates a keyed hash of these values and sends the result back to SPOTR$_V$. SPOTR$_V$ verifies the authenticity of the hash and ensures that the response is received within a short interval from the challenge. If the verification succeeds, SPOTR$_V$ uses its private key to sign a time stamped token and sends the result to U. U contacts the server S over the anonymizer (see above) and sends the token signed by SPOTR$_V$. S verifies V's signature as well as the freshness (and single use) of the token.

Secret Sharing. Our constructions use a (k, m) threshold secret sharing (TSS) [15] solution. Given a value R, TSS generates m shares such that at least k shares are needed to reconstruct R. A (k, m)-TSS solution satisfies the property of *hiding*: An adversary (provided with access to a TSS oracle) controlling the choice of two values R_0 and R_1 and given less than k shares of R_b, $b \in_R \{0, 1\}$, can guess the value of b with probability only negligible higher than 1/2.

Secret sharing will enable the provider to decrypt encrypted counters only when at least k users (out of m) have checked-in at a venue. The k out of m property supports failures: users who check-in but do not participate in the protocol.

III. PROFIL$_R$

As mentioned before, 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.

Overview. 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 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 Fig. 1).

When a user U checks-in at venue V, it first engages in the *Spotter* protocol with SPOTR$_V$, allowing the venue to verify U's physical presence. A successful run of *Spotter* 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 Sh of shares of the secret key that have been revealed so far.

Subsequently, U runs $CheckIn$ with SPOTR$_V$, to send its share of the secret key and to receive the encrypted counter sets. As shown in Fig. 1, 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 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. The communication during $PubStats$ takes place over an authenticated channel (see Fig. 1).

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 a 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 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 decrypting $C_i[j]$ or knowing the current counter's value: $C_i[j] + + = [E(u_j, c_j)y, E(u'_j, j)] = [y^{c_j+1} u_j^r, E(u'_j, j)] = [E(u_j, c_j + 1), E(u'_j, j)]$.

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(),S(k)): The provider S runs the key generation function $KG(l)$ 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 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 with b sub-ranges, SPOTR$_V$ performs the following steps:

- Initialize counters $c_1, .., c_b$ to 0.
- 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 $S_{key} = \emptyset$.
- Generate system wide parameters k and $m > k$ and initialize the (k, m) TSS.

Spotter(U(L,T),V(),S(k)): Let L and T denote U's location and current time. To ensure anonymity, U generates fresh random MAC and IP addresses. These addresses are used for a single execution of the $Spotter$ and $CheckIn$ protocols. SPOTR$_V$ uses one of the location verification procedures proposed in [8] to verify U's presence at L and T (see Section II-D).

Let U be the i-th user checking-in at V. If the verification succeeds and $i \leq k$, S uses the (k, m) 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_i, n, V), V(n, y, C_{i-1}, S_{key})): Executes only if the previous run of $Spotter$ is successful. U uses the same random MAC and IP addresses as in the previous $Spotter$ run. Let U be the i-th user checking-in at V. Then, C_{i-1} is the current set of encrypted counters. SPOTR$_V$ sends C_{i-1} to U. Let v, U's value on dimension D, be within R's j-th sub-range, i.e., $v \in R_j$. 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}: $C_i = \{RE(v_l, v'_l, C_{i-1}[l]) | l = 1..b, l \neq j\} \cup RE(v_j, v'_j, C_{i-1}[j] + +)\}$.
- Send C_i and the signed (by S) share p_i of p to V.

If SPOTR$_V$ successfully verifies the signature of S on the share p_i, U and SPOTR$_V$ 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, SPOTR$_V$ replaces C_{i-1} with C_i and adds the share p_i to the set S_{key}. Otherwise, SPOTR$_V$ drops C_i and rolls back to C_{i-1}.

PubStats(V(C_k,Sh,V),S(p,q)): SPOTR$_V$ 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 $(t_1, t'_1), .., (t_b, t'_b)$ and random permutation π, then sends to SPOTR$_V$ the proof set $P_{i-1} = \pi\{RE(t_l, t'_l, C_{i-1}[l]), l = 1..b\}$.
- U generates random values $(w_1, w'_1), .., (w_b, w'_b)$. It sends to SPOTR$_V$ the proof set $P_i = \pi\{RE(w_l, w'_l, C_i[l]), l = 1..b\}$
- SPOTR$_V$ generates a random bit a and sends it to U.

- If $a = 0$, U reveals random values $(t_1, t'_1), .., (t_b, t'_b)$ and $(w_1, w'_1), .., (w_b, w'_b)$. SPOTR$_V$ verifies that for each $l = 1..b$, $RE(t_l, t'_l, C_{i-1}[l])$ occurs in P_{i-1} exactly once, and that for each $l = 1..b$, $RE(w_l, w'_l, C_i[l])$ occurs in P_i exactly once.
- If $a = 1$, U reveals $o_l = v_l w_l t_l^{-1}$ and $o'_l = v'_l w'_l t'^{-1}_l$, 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 $RE(o_j, o'_j, P_{i-1}[j]y) = P_i[j]$.
- If any verification fails, SPOTR$_V$ aborts the protocol.

C. Preventing Venue-User Collusion

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-1 Sybil user accounts or simulating k-1 check-ins, can use PROFIL$_R$ 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 [16] 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 $Spotter$ 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. The privacy breach mentioned above is due to the fact that now S can correlate $CheckIn$s executed using the same $R_{U,e}$. However, S does not know the real user identity behind $R_{U,e}$ – due to the use of blind signatures.

D. Analysis

Given a set of encrypted counters C, let $\bar{\mathcal{C}}$ denote the set of re-encryptions of records of C, where only one record has its counter incremented. To show that ZK-CTR(i) is a ZK proof of $C_i \in \bar{\mathcal{C}}_{i-1}$, we need to prove completeness, soundness and zero-knowledge.

Theorem 1: ZK-CTR(i) is complete.

Proof: If $C_i \in \bar{\mathcal{C}}_{i-1}$, 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, showing 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 $P_i \in \bar{\mathcal{P}}_{i-1}$. ■

Theorem 2: ZK-CTR(i) is sound.

Proof: We need to prove that if $C_i \notin \bar{\mathcal{C}}_{i-1}$, U cannot convince S unless with negligible probability. For simplicity, we assume $C_i \notin \bar{\mathcal{C}}_{i-1}$ due to a single record in C_i being "bad": $C_{i-1}[j] = E(u_j, u'_j, c_j, j)$ 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 $P_{i-1}[j] = E(u_j t_j, u'_j t'_j, c_j, j)$ and $P_i[j] = E(v_j w_j, v'_j w'_j, c'_j, j')$. If however $a = 1$, U has to produce a value α_j, such that $RE(\alpha_j, E(u_j, c_j)) = E(v'_j, c'_j)$ or $RE(\alpha_j, E(u_j, c_j + 1)) = E(v'_j, c'_j)$. In the first case, this means $y^{c_j}(u_j \alpha_j)^l = y_{c'_j} v'^l_j \ mod \ n$. Without knowing n's factorization, U cannot compute l's inverse modulo $\phi(n)$. Then, the equation is satisfied only if $c'_j = c_j + zl$, for an integer z. Note however that Benaloh's cryptosystem only works for values in $\mathbb{Z}^*_l$, making this condition impossible to satisfy.

The second case is similar. The second cheating option is to assume a will be 1 and build $P_i[j]$ 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 $c'_j = c_j + zl$, 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 $1/2^s$. ■

Theorem 3: ZK-CTR(i) is "zero-knowledge".

Proof: We 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 [17], [18]. Specifically, let S^* be an arbitrary, fixed, expected polynomial time interactive Turing machine (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 $\langle C, S^* \rangle$ (which denotes the protocol run by a client C and 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 π_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 $a \in_R \{0, 1\}$. If $a = 0$, M^* picks a random permutation π_i, generates t_l, t'_l, $l = 1..b$ randomly and computes $P_{2i} = \pi_i\{RE(t_l, t'_l, C_{i-1}[l]), l = 1..b\}$. It then generates random values w_l, w'_l, $l = 1..b$, randomly and computes the set $P_{2i+1} = \pi_i\{RE(w_l, w'_l, C_i[l]), l = 1..b\}$. 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_l, o'_l randomly, $l = 1..b$. It then generates a random $j \in 1..b$ and computes P_{2i+1}

such that for all $l = 1..b, l \neq j$, $RE(o_l, o'_l, P_{2i}[l]) = P_{2i+1}[l]$ and for the j-th position, $RE(o_j, o'_j, P_{2i}[j]y) = P_{2i+1}[j]$.

- *Step 2*: M^* sets $b = S^*(x, r; P_0, P_1, \pi_0, .., P_{2i-2}, P_{2i-1}, \pi_{i-1}, P_{2i}, P_{2i+1})$. 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.

  (Case 1). $a = b = 0$. M^* can produce t_l, t'_l, w_l, w'_l, $l = 1..b$ and π_i to prove that $P_{2i} = \pi_i\{RE(t_l, t'_l, C_{i-1}[l]), l = 1..b\}$ and $P_{2i+1} = \pi_i\{RE(w_l, w'_l, C_i[l]), l = 1..b\}$. M^* sets b_i 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).

  (Case 2). $a = b = 1$. M^* can produce o_l, o'_l, $l = 1..b$, and index j such that $RE(o_l, o'_l, P_{2i}[l]) = P_{2i+1}[l]$, $l = 1..b, l \neq j$ and $RE(o_j, o'_j, P_{2i}[j]y) = P_{2i+1}[j]$. M^* sets b_i 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).

  (Case 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 time 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:** Let $\langle C, S^*\rangle|(L_0, L_1)$ denote the output of the interaction between client C and the ITM S^*, given input L_0, L_1. The probability distribution of $\langle C, S^*\rangle|(L_0, L_1)$ and of $M^*|(L_0, L_1)$ are identical.

  Proof: The output of $\langle C, S^*\rangle|(L_0, L_1)$ 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 $\Pi^{(x,r,i)}_{M^*}$ and $\Pi^{(x,r,i)}_{CS^*}$ be the probability distributions of the first i tuples output by M^* and $\langle C, S^*\rangle$. We need to show that for any fixed random input r, $\Pi^{(x,r,t)}_{M^*} = \Pi^{(x,r,t)}_{CS^*}$. We prove this by induction. The base case, where $i = 0$, holds immediately. In the induction step we assume that $\Pi^{(x,r,i)}_{M^*} = \Pi^{(x,r,i)}_{CS^*} = T^{(i)}$. We need to prove that the $i+1$st tuples in $\Pi^{(x,r,i+1)}_{M^*}$, denoted by $\Pi^{(i+1)}_{M^*}$ and in $\Pi^{(x,r,i+1)}_{CS^*}$, denoted by $\Pi^{(i+1)}_{CS^*}$ have the same distribution. We show that $\Pi^{(i+1)}_{M^*}$ and $\Pi^{(i+1)}_{CS^*}$ are uniform over the set $V = \{(P_{2i}, P_{2i+1}, \pi_i, b)|b = S^*(x, r, T^{(i)}||P) \wedge ((P_{2i} = \pi_i RE(C_0), P_{2i+1} = \pi_i RE(C_1),\ if\ b = 0) \vee (P_{2i+1}[l] = RE(P_{2i}[l]), l = 1..b, l \neq j, P_{2i+1}[j] = yRE(P_{2i}[j]),\ if\ b = 1)\}$.

  For $\Pi^{(i+1)}_{CS^*}$, this is the case, by construction. If $\Pi^{(i+1)}_{M^*}$ 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. This completes the theorem proof. ■

We can now prove the following result:

Theorem 4: PROFIL$_R$ provides k-privacy.

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

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

Location Correctness: The user's location is verified in the $Spotter$ protocol. A malicious user not present at venue V, is unable to establish a connection with the device deployed at V, SPOTR$_V$. 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 $CheckIn$ protocol.

LCP Correctness: 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 2 shows this attack succeeds with probability $1/2^s$. Second, it attempts to prevent V from decrypting the counter sets after k users have run CheckIn. This can be done by preventing SPOTR$_V$

from reconstructing the private Benaloh key. Key shares are however signed by the provider, allowing SPOTR$_V$ to detect invalid shares.

CI-IND Satisfaction: To see that PROFIL$_R$ satisfies the CI-IND property, let $\mathcal{A}$ be an adversary that has an ϵ advantage in the CI-IND game. We assume a honest challenger, who does not run *Spotter* and *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.

No identifying information is sent by users during the *Spotter* and *CheckIn* procedures: the pseudonyms are *blindly* signed by S, all communication with S takes place over an anonymizer, and all communication with a venue is done using randomly chosen MAC and IP addresses. Thus, we can use $\mathcal{A}$ to build another adversary $\mathcal{B}$ that has the advantage ϵ either against (i) the blind signature protocol [16], or against the (ii) privacy property provided by the anonymizer.

Finally, we note that an adversary can use the *CheckIn* procedure to launch denial of service attacks against a venue, consuming its computation resources.

IV. SNAPSHOT LCP

We extend PROFIL$_R$ 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* PROFIL$_R$

We assume a user U co-located with k other users $U_1, .., U_k$. U needs to generate the LCP of their 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 $U_1, .., U_k$. Snapshot PROFIL$_R$ consists of three protocols, $\{Setup, LCPGen, PubStats\}$:

Setup$(U(r), U_1, .., U_k())$: U runs the following steps:

- Run the key generation function $KG(l)$ of the Benaloh cryptosystem (see Section II-D). Send the public key n and y to each user $U_1, .., U_k$.
- Engage in a multi-party secure function evaluation protocol [19] with $U_1, .., U_k$ 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 $R_1..R_k = R \ mod \ n$ and R_i is only known to U_i.
- Assign each of the k users a unique label between 1 and k. Let $U_1, .., U_k$ denote this order.
- Generate $C_0 = \{E(x_1, x_1', 0, 1), .., E(x_b, x_b', 0, b)\}$, where x_i, x_i', $i = 1..b$ are randomly chosen. Store C_0 indexed on dimension D.

Each of the k users engages in a 1-on-1 *LCPGen* with U to privately and correctly contribute her profile to U's LCP.

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

- Generate random values $(v_1, v_1'), .., (v_b, v_b')$. Let j be the index of the range where U_i fits on dimension D.
- Compute the new encrypted counter set C_i as: $C_i = \{RE(v_l, v_l', C_{i-1}[l])R_i \ mod \ n | l = 1..b, l \neq j\} \cup RE(v_j, v_j', C_{i-1}[j] + +)R_i \ mod \ n\}$ and send it to U.
- Engage in a ZK-CTR protocol to prove that $C_i \in \bar{C}_{i-1}$. 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 *LCPGen* with $U_1, .., U_k$, U's encrypted counter set is $C_k = \{E_j = E(u_j, u_j', c_j, j)R_1..R_k | j = 1..d\}$, where u_j and u_j' are the product of the obfuscation factors used by $U_1, .., U_k$ in their re-encryptions. The following protocol enables U to retrieve the snapshot LCP.

PubStats$(U(C_k))$**:** Compute $E_j K$, $\forall j = 1..d$, where $K = R^{-1} \ mod \ n$ ($R = R_1..R_k$), decrypt the outcome using the private key (p, q) and publish the resulting counter value. U verifies that the j-th decrypted record is of format (c_j, j) and that the sum of all counters equals k. If any verification fails, U drops the statistics - a cheater exists. Otherwise, the resulting counters denote the aggregate stats of $U_1, .., U_k$.

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, $i = 1..k$.

This protocol is a secure function evaluation - the participants learn their aggregated profiles, without learning the profiles of any participant in the process. We note however that existing SFE solutions cannot be used here: We need to ensure the input user profiles are correct, that is, each user increments a single counter.

V. APPLICATIONS

We now propose two PROFIL$_R$ applications.

A. *Public Safety*

Is a person likely to be safe in a specific public space, presently? The answer to this question is a function of the context of the space and of the person considered. In addition to location and time, the context is greatly influenced by the people present in that space. In previous work [20] we have proposed a personalized safety recommendation system, that leverages the history of locations visited by U to define his *safety index*. Specifically, we defined U to be safe within a context Ct, if U has a higher chance of crimes to occur around him, than the people in Ct.

We propose to use PROFIL$_R$ to build finer grained personalized safety recommendations, with privacy. PROFIL$_R$ divides the safety index interval ([0, 1]) into sub-intervals, and associates a counter with each. PROFIL$_R$ enables then a set of users to privately and correctly compute the distribution of their safety index values. Then, U is safe in a context Ct, if the number (or percentage) of users in Ct

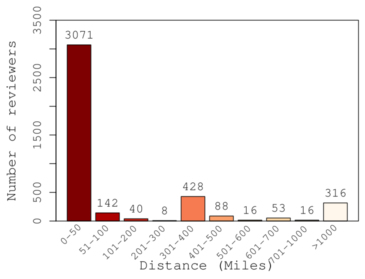

Fig. 2. Yelp venue stats: Distribution of the distance from one venue (“Ike’s Place”) to the home cities of its reviewers. 3000+ reviews were written by locals, but a large number of reviews were written by far-away visitors.

whose safety index values are smaller or equal to U’s safety index (are safer than U), exceeds a system wide threshold parameter.

B. Real-Time Yelp Venue Stats

In a second application, we rely on PROFIL$_R$ to enable venues to collect fine grained, real time statistics over the profiles of patrons with Yelp accounts. To motivate participation, PROFIL$_R$ prevents venues from inferring the identity and even the anonymous profiles of the currently present users.

Yelp is an excellent source of user profile information. Yelp users own accounts storing a wealth of public and personal information, including name, home city, friends, reviews written, photos uploaded, check-ins, “Elite” badges, etc. Knowing the real time distribution of current patron profile information, such as locals vs. non-locals, gender, the types of venues preferred, can help venues understand their customers. Furthermore, by studying the evolution in time of such information, e.g., using time series analysis, may enable venues to generate forecasts and better cater to their customers.

Fig. 2 illustrates this concept: it shows the distribution of the (great-circle) distance in miles from “Ike’s Place” venue in San Francisco, CA and the home cities of its (4000+) reviewers. More than 3000 reviews were left by locals, but far away customers also form a sizeable percentage.

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.

We have first measured the overhead of the $Setup$ operation. If d is the number of profile dimensions, N is the Benaloh modulus size and b the sub-range count of domain D, the computation overhead of Setup is $T_{Setup} = T_{keysig} + dbT_E + T_{TSS}$. T_{keysig} is the time to generate the signature key, T_E is the average time of Benaloh encryption and T_{TSS} is the time to initialize the TSS (i.e., random polynomial generation). The storage overhead of $Setup$ is $Store_{Setup} = dbN$.

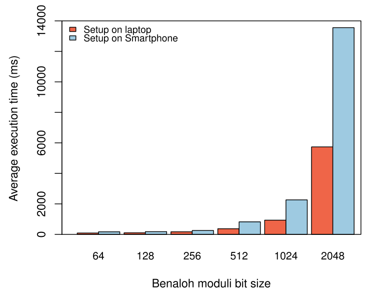

Fig. 3. *Setup* dependence on Benaloh modulus size. Note the significant increase to 13.5s for a 2048 bit modulus. This cost is however amortized over multiple check-in executions.

We set the b to be 10, Shamir’s TSS group size to 1024 bits and RSA’s modulus size to 1024 bits. Fig. 3 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.

The computation overhead of $CheckIn$ is $T_{CI} = bT_{RE} + T_{ZK}$, where T_{RE} is the Benaloh re-encryption cost and T_{ZK} is the overhead of the ZK-CTR protocol. The formula does not consider the cost of modular multiplication, random number generation and random permutation operations, that are neglibile compared to the other costs. Given s, the number of rounds of ZK-CTR, $T_{ZK} = 2sbT_{RE} + sbT_{RE} + \frac{s}{2}bT_{RE} = \frac{7}{2}sbT_{RE}$. The communication overhead is $T_{com\_CI} = bN + T_{com\_ZK}$. The communication cost of ZK-CTR, T_{com_ZK} is $s(2bN + \frac{1}{2}4bN + \frac{1}{2}2bN) = 5sbN$.

We now focus on the most resource consuming component, the ZK-CTR protocol. While the above formulas assume similar capabilities for the client and venue components, we now measure the client side running on the smartphone and the venue component executing on the laptop. Fig. 4 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. We have set $b = 10$. 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, Fig. 5 shows the overheads of these components as a function of the number of ZK-CTR rounds, when the Benaloh key size is 1024 bit and $b = 10$. For 30 rounds, when a cheating client’s probability of success is 2^{-30} (1 in a billion), 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. 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. Fig. 6 shows the dependency of the communication overhead (in KB) on b,

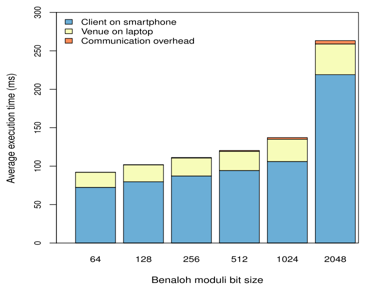

Fig. 4. The overhead imposed by ZK-CTR as a function of the Benaloh modulus size. Note the significant overhead increase for a 2048-bit modulus, of approximately 260ms per ZK-CTR round.

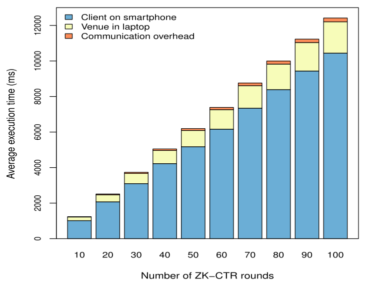

Fig. 5. The overhead of the ZK-CTR protocol as a function of the number of proof rounds. The linear increase in the number of rounds leads to a 12s overhead for 100 rounds. 100 rounds reduce however the probability of client cheating to an insignificant value, 2^{-100}.

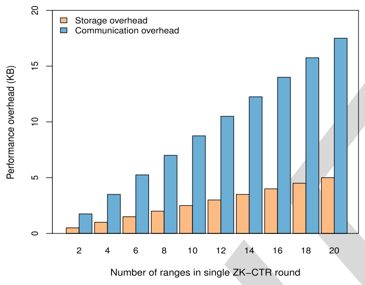

Fig. 6. Storage and communication overhead (in KB) as a function of b, the number of sub-intervals considered in the statistics computation. Even for $b = 20$, the storage overhead is only 5KB and the communication is 17KB.

when $N = 1024$. Even when $b = 20$, the communication overhead is around 17KB. Fig. 6 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

This article significantly extends a short paper [21], [22] with privacy and correctness definitions, an expanded version of PROFIL$_R$, a decentralized, snapshot PROFIL$_R$, detailed privacy and correctness proofs and applications.

Location cloaking. 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 [23], followed by a significant body of work [24], [25], [26]. 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 of participating users: user profiles are anonymous, only aggregates are available for inspection, and interactions with venues and the provider are indistinguishable.

l-diversity. Machanavajjhala et al. [27] have shown that k-anonymity for published user data, where each record is indistinguishable from at least $k-1$ other records (for sensitive attributes), is not sufficient to provide anonymity. To address this, they defined an l-diverse data block of tuples from various users, as one that contains at least l "well-represented" values for any sensitive attribute. We note that we do not collect individual (anonymized) user data. Instead, we build statistics over user data, that can be published only if k users contribute.

GSN privacy. Puttaswamy and Zhao [28] require users to store their information encrypted on the GSN provider. This includes 'friendship" and "transaction" proofs, cryptographically encrypted tokens encoding friend relations and messages. The proofs can only be decrypted by those who know the decryption keys. Transaction proofs are stored in "buckets" associated with approximate locations (e.g., blocks), enabling users to retrieve information pertinent to their current location. PROFIL$_R$ takes the next step, by enabling the aggregation of user data in a privacy preserving manner.

Mascetti et al. [29] propose solutions that hide user location information from the provider and enable users to control the information leaked to participating friends (e.g., co-location events), with a view to improve service precision, computation and communication costs. Freni et al. [30] argue that the inherent nature of geosocial networks makes it hard for users to gauge their privacy leaks. The proposed solution relies on a trusted third party to process posted locations according to user preferences, before publishing them on the GSN provider. Wernke et al. [31] use secret sharing and multiple, non-colluding service providers to devise secure solutions for the management of private user locations when none of the providers can be fully trusted. The position of a user is split into shares and each server stores one. A compromised server can only reveal erroneous user positions.

In contrast, PROFIL$_R$ provides the novel functionality of allowing the provider, venues and even users to privately compute LCPs over visitors or co-located users. PROFIL$_R$ does not require multiple, mutually untrusted servers, or trusted third parties.

Thompson et. al. [32] proposed a solution in which database storage providers compute aggregate queries without gaining knowledge of intermediate results; users can verify the results of their queries, relying only on their trust of the data owner. In addition to assuming a different environment, PROFIL$_R$ does not assume venue owners to be trustworthy. Toubiana et. al [33] 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.

Online social network privacy. Recent work on preserving the privacy of users from the online social network provider includes Cutillo et al. [34], who proposed Safebook, a distributed online social networks where insiders are protected from external observers through the inherent flow of information in the system. Tootoonchian et al. [35] proposed Lockr, a system for improving the privacy of social networks by using the concept of a social attestation, which is a credential proving a social relationship. Baden et al. [36] introduced Persona, a distributed social network with distributed account data storage. While PROFIL$_R$ builds on this work by requiring users to store their GSN information, its focus rests on protecting the privacy of users while *simultaneously* allowing venues to collect valuable statistics over visitors. This dual goal of PROFIL$_R$ differentiates this paper from previous work.

Sybil account detection. Our work relies on the assumption that participants cannot control a large number of fake, Sybil accounts. We briefly describe several relevant techniques for detecting social network Sybils. When given access to data collected by the social network provider, Wang et al. [37] proposed an approach that detects Sybil accounts based on their click stream behaviors (traces of click-through events in a browsing session). Molavi et al. [38] introduce a practical approach that focuses on the effects of Sybil accounts. They propose to defend against reviews from multiple identities of a single attacker, by associating weights with ratings and by introducing the concept of "relative ratings".

VIII. CONCLUSION

In this paper we have proposed PROFIL$_R$, a framework and mechanisms for privately and correctly building location-centric profiles. We have proved the ability of our solutions to satisfy the privacy and correctness requirements. We have introduced two applications for PROFIL$_R$. We have shown that PROFIL$_R$ is efficient, even when executed on resource constrained mobile devices.

REFERENCES

[1] Yelp, Inc., San Francisco, CA, USA. (2014, Feb. 28) [Online]. Available: http://www.yelp.com

[2] Foursquare, New York, NY, USA. (2014, Feb. 28) [Online]. Available: https://foursquare.com/

[3] B. Krishnamurthy and C. E. Wills, "On the leakage of personally identifiable information via online social networks," *Comput. Commun. Rev.*, vol. 40, no. 1, pp. 112–117, 2010.

[4] E. Steel and G. Fowler. (2010). *Facebook in Privacy Breach* [Online]. Available: http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html

[5] Foursquare Official Blog, New York, NY, USA. (2011). *On Foursquare, Cheating, and Claiming Mayorships from your Couch* [Online]. Available: http://goo.gl/F1Yn5

[6] (2012). *Raspberry Pi. An ARM GNU/Linux Box for $25. Take a Byte* [Online]. Available: http://www.raspberrypi.org/

[7] G. Coley, *Beagleboard System Reference Manual*. Dallas, TX, USA, BeagleBoard. Org., Dec. 2009.

[8] B. Carbunar and R. Potharaju, "You unlocked the Mt. Everest badge on foursquare! countering location fraud in geosocial networks," in *Proc. 9th IEEE Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Feb. 2012, pp. 182–190.

[9] J. Benaloh, "Dense probabilistic encryption," in *Proc. Workshop Sel. Areas Cryptograph.*, 1994, pp. 120–128.

[10] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proc. 14th Annu. ACM Symp. Theory Comput.*, New York, NY, USA, 1982, pp. 365–377.

[11] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[12] M. Abe, "Universally verifiable mix-net with verification work indendent of the number of mix-servers," in *Proc. EUROCRYPT*, 1998, pp. 437–447.

[13] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *Proc. USENIX Security Symp.*, 2004, pp. 303–320.

[14] (2013). *Orbot: Tor on Android* [Online]. Available: https://www.torproject.org/docs/android.html.en

[15] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[16] D. Chaum, "Blind signatures for untraceable payments," in *Proc. Adv. Cryptol. CRYPTO*, 1982, pp. 199–203.

[17] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, 1989.

[18] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems," *J. ACM*, vol. 38, no. 3, pp. 690–728, 1991.

[19] M. Jakobsson and A. Juels, "Mix and match: Secure function evaluation via ciphertexts," in *Proc. Adv. Cryptol. 6th Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2000, pp. 162–177.

[20] J. Ballesteros, B. Carbunar, M. Rahman, N. Rishe, and S. S. Iyengar, "Towards safe cities: A mobile and social networking approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 6, pp. 1–14, Nov. 2013.

[21] B. Carbunar, M. Rahman, J. Ballesteros, and N. Rishe, "Private location centric profiles for geosocial networks," in *Proc. 20th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (GIS)*, 2012, pp. 458–461.

[22] M. Rahman, J. Ballesteros, B. Carbunar, N. Rishe, and A. V. Vasilakos, "Toward preserving privacy and functionality in geosocial networks," in *Proc. 18th Int. Conf. Mobile Comput. Netw.*, 2013, pp. 1–3.

[23] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. MobiSys*, 2003, pp. 31–42.

[24] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. Work, J.-C. Herrera, *et al.*, "Virtual trip lines for distributed privacy-preserving traffic monitoring," in *Proc. ACM MobiSys*, 2008, pp. 15–28.

[25] F. G. Olumofin, P. K. Tysowski, I. Goldberg, and U. Hengartner, "Achieving efficient query privacy for location based services," in *Proc. Privacy Enhancing Technol.*, 2010, pp. 93–110.

[26] X. Pan, X. Meng, and J. Xu, "Distortion-based anonymity for continuous queries in location-based mobile services," in *Proc. GIS*, 2009, pp. 256–265.

[27] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, 2006, pp. 1–24.

[28] K. P. N. Puttaswamy and B. Y. Zhao, "Preserving privacy in location-based mobile social applications," in *Proc. 11th Workshop Mobile Comput. Syst. Appl.*, New York, NY, USA, 2010, pp. 1–6.

[29] S. Mascetti, D. Freni, C. Bettini, X. Sean Wang, and S. Jajodia, "Privacy in geo-social networks: Proximity notification with untrusted service providers and curious buddies," *VLDB J.*, vol. 20, no. 4, pp. 541–566, Aug. 2011.

[30] D. Freni, C. Ruiz Vicente, S. Mascetti, C. Bettini, and C. S. Jensen, "Preserving location and absence privacy in geo-social networks," in *Proc. 19th ACM CIKM*, New York, NY, USA, 2010, pp. 309–318.

[31] M. Wernke, F. Durr, and K. Rothermel, "PShare: Position sharing for location privacy based on multi-secret sharing," in *Proc. PerCom*, 2012, pp. 153–161.

[32] B. Thompson, S. Haber, W. G. Horne, T. Sander, and D. Yao, "Privacy-preserving computation and verification of aggregate queries on outsourced databases," in *Proc. 9th Int. Symp. Privacy Enhancing Technol.*, 2009, pp. 185–201.

[33] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas, "Adnostic: Privacy preserving targeted advertising," in *Proc. Network Distrib. Syst. Security (NDSS) Symp.*, 2010, pp. 1–3.

[34] A. Cutillo, R. Molva, and T. Strufe, "Safebook: Feasibility of transitive cooperation for privacy on a decentralized social network," in *Proc. IEEE WOWMOM*, Jun. 2009, pp. 1–6.

[35] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman, "Lockr: Better privacy for social networks," in *Proc. ACM CoNEXT*, 2009, pp. 1–12.

[36] R. Baden, N. Spring, and B. Bhattacharjee, "Identifying close friends on the internet," in *Proc. Hotnets*, 2009, pp. 1–6.

[37] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," in *Proc. USENIX Security*, 2013, pp. 1–15.

[38] A. M. Kakhki, C. Kliman-Silver, and A. Mislove, "Iolaus: Securing online content rating systems," in *Proc. 22nd Int. World Wide Web Conf. (WWW)*, Rio de Janeiro, Brazil, May 2013, pp. 1–5.

Authors' photographs and biographies not available at the time of publication.