ProfilR: Toward Preserving Privacy and Functionality in Geosocial Networks

Bogdan Carbunar∗, Mahmudur Rahman∗, Jaime Ballesteros∗, Naphtali Rishe∗, Athanasios V. Vasilakos†
∗School of Computing and Information Sciences, Florida International University, Miami, FL
†Dept. of Electrical and Computer Engineering, National Technical University of Athens (NTUA), Greece

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 ProfilR, a framework for constructing location centric profiles (LCPs), aggregates built over the profiles of users that have visited discrete locations (i.e., venues). ProfilR 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 co-located users. An Android implementation shows that ProfilR is efficient: the end-to-end overhead is small even under strong privacy and correctness assurances.

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 [2] and Foursquare [?] 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 [?] and even sell [?] 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 ProfilR, a framework that allows the construction of LCPs based on the profiles of present users, while ensuring the privacy and correctness of participants. Informally, we define privacy as the inability of venues and the GSN provider to accurately learn user information, including even anonymized location trace profiles. 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 [?], 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 ProfilR, that relieves the GSN provider from a costly involvement in venue specific activities. To achieve this, ProfilR 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 ProfilR satisfies the introduced correctness and privacy properties.

Second, we propose a completely decentralized ProfilR extension, built around the notion of snapshot LCPs. The distributed ProfilR 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 ProfilR 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 ProfilR, a framework for computing LCPs. Devise both a venue centric and a decentralized solution. Prove that ProfilR satisfies the proposed privacy and correctness properties.
• Provide two applications for ProfilR: (i) privacy preserving, personalized public safety recommendations and (ii) privately building real time statistics over the profiles of venue patrons with Yelp accounts.
• Evaluate ProfilR through an Android implementation. Show that ProfilR is efficient even when deployed on
previous generation smartphones.

The paper is organized as follows. Section ?? describes the system and adversary model and defines the problem. Section ?? introduces PROFIL\textsubscript{R} and proves its privacy and correctness. Section ?? introduces the notion of snapshot LCPS and presents a distributed, real-time variant of PROFIL\textsubscript{R}. Section ?? describes two PROFIL\textsubscript{R} applications. Section ?? evaluates the performance of the proposed constructs. Section ?? describes related work and Section ?? concludes.

II. MODEL AND BACKGROUND

We consider a core functionality that is supported by the most influential geosocial network (GSN) providers, Yelp [?] and Foursquare [?]. 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 [?], a BeagleBoard [?] or any Android smartphone). This equipment can be installed and used for other purposes as well, including detecting fake user check-ins [?] preventing fake badges and incorrect rewards, and validating social network (e.g., Yelp [?]) reviews. Venue deployed equipment provides a necessary ingredient: ground truth information from remote locations.

A. Location Centric Profiles

Each user has a profile \( P_{U} = \{p_{U1}, p_{U2}, ..., p_{Ud}\} \), consisting of values on \( d \) dimensions (e.g., age, gender, home city, etc). Each dimension has a range, or a set of possible values. Given a set of users \( U \) at location \( L \), the location centric profile at \( L \), denoted by \( LCP(L) \) is the set \( \{LCP_1, LCP_2, ..., LCP_d\} \), where \( LCP_i \) denotes the aggregate statistics over the \( i \)-th dimension of profiles of users from \( U \).

In the following, we focus on a single profile dimension, \( D \). We assume \( D \) takes values over a range \( R \) that can be discretized into a finite set of sub-intervals (e.g., set of continuous disjoint intervals or discrete values). Then, given an integer \( b \), chosen to be dimension specific, we divide \( R \) into \( b \) intervals/sets, \( R_1, ..., R_b \). For instance, gender maps naturally to discrete values (\( b = 2 \)), while age can be divided into disjoint sub-intervals, with a higher \( b \) value.

We define the aggregate statistics \( S \) for dimension \( D \) of \( LCP(L) \) to consist of \( b \) counters \( c_1, ..., c_b \); \( c_i \) records the number of users from \( U \) whose profile value on dimension \( D \) falls within range \( R_i \), \( i = 1..b \).

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 Figure ???. 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 \( Prot(P_i(args_1), ..., P_n(args_n)) \) 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 \( A \) 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 \( A \) on behalf of \( i \) users. Let \( C_i \) denote the resulting counter set. For each \( j = 1..b, A \) outputs \( \epsilon_j^i \), its guess of the value of the \( j \)-th counter of \( C_i \). The advantage of \( A \), \( Adv(A) = |Pr[C_i[j] = \epsilon_j^i] - 1/(i + 1)| \), defined for each \( j = 1..b \), is negligible.

Location Correctness: Let \( A \) denote an adversary that controls the GSN provider and any number of users. Let \( \mathcal{C} \) be a challenger that controls a venue \( V \). \( A \) running as a user...
counter’s value or the encryption key and (ii) to re-encrypt all counters afterward. If \( C'_V \notin C_V \), the CheckIn completes successfully with only negligible probability.

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

**C. Attacker Model**

We assume venue owners are malicious and will attempt to learn private information from 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 [?]1, an extension of the Goldwasser-Micali [?]. 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 \( l((p-1)/l) \) and \( \text{gcd}(l, (p-1)/l) = 1 \) and \( \text{gcd}(l, q-1) = 1 \). Let \( n = pq \). Select \( y \in \mathbb{Z}_n^* \), such that \( y^{(p-1)(q-1)/l} \mod n \neq 1 \). \( y \) and \( n \) are the public key and \( p \) and \( q \) are the private key.

- \( E(u, m) \): Encrypt message \( m \in \mathbb{Z}_n^* \), 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 output \( m = 0 \). Otherwise, for \( i = 1 \ldots l \), compute \( s_i = y^{-z} \mod n \). If \( s_i = 1 \), return \( m = i \).

Benaloh’s cryptosystem is additively homomorphic:
\[
E(u_1, m_1)E(u_2, m_2) = E(u_1 u_2, m_1 + m_2).
\]
We further define the re-encryption function \( RE(v, E(u, m)) \) to be \( y^m u^l v^l = E(wv, m) \). 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[?], [?], [?] 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 [?], an Android implementation of Tor [?].

**Location Verification.** We use one of the protocols proposed in [?] 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 \( \Delta T \) and a fresh random value \( R \). \( U \)’s device generates a key 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) [?] 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 \( R_0 \), \( b \in_R \{0, 1\} \), 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**

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 Figure ??).

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 Figure ??, 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. SOTR\textsubscript{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, SOTR\textsubscript{V} runs PubStats to reconstuct the private key, decrypt all encrypted counters and publish the tally. The communication during PubStats takes place over an authenticated channel (see Figure ??).

A. The Solution

Let C\textsubscript{i} denote the set of encrypted counters at V, following the i-th user run of CheckIn. C\textsubscript{i} = \{C\textsubscript{i}[1], ..., C\textsubscript{i}[b]\}, where C\textsubscript{i}[j] denotes the encrypted counter corresponding to R\textsubscript{j}, the j-th sub-range of R. We write C\textsubscript{i}[j] = E(u\textsubscript{j}, u\textsubscript{j}', c\textsubscript{j}, j) = [E(u\textsubscript{j}), E(u\textsubscript{j}', j)], where u\textsubscript{j} and u\textsubscript{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\textsubscript{j}, v\textsubscript{j}', E(u\textsubscript{j}, u\textsubscript{j}', c\textsubscript{j}, j)) denote the re-encryption of the j-th record with two random values v\textsubscript{j} and v\textsubscript{j}';

\begin{align*}
RE(v\textsubscript{j}, v\textsubscript{j}', E(u\textsubscript{j}, u\textsubscript{j}', c\textsubscript{j}, j)) &= \left[RE(v\textsubscript{j}, E(u\textsubscript{j}, c\textsubscript{j})), \right. \\
RE(v\textsubscript{j}', E(u\textsubscript{j}', j))] &= E(u\textsubscript{j}, v\textsubscript{j}, c\textsubscript{j}, E(u\textsubscript{j}', j)]. Let C\textsubscript{i}[j] + + = E(u\textsubscript{j}, u\textsubscript{j}', c\textsubscript{j} + 1, j) denote the encryption of the incremented j-th counter. Note that incrementing the counter can be done without decrypting C\textsubscript{i}[j] or knowing the current counter’s value: C\textsubscript{i}[j] + + = [E(u\textsubscript{j}, c\textsubscript{j}y, E(u\textsubscript{j}', j)] = [y^{c\textsubscript{j}+1}u\textsubscript{j}', E(u\textsubscript{j}', j)] = E(u\textsubscript{j} + 1, E(u\textsubscript{j}', j)].

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

\textbf{Setup}(V, S(k)): The provider S runs the key generation function KG(l) of the Benaloh cryptosystem (see Section ??). Let p and q be the private key and n and y the public key. S sends the public key to SOTR\textsubscript{V}. SOTR\textsubscript{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, SOTR\textsubscript{V} performs the following steps:

- Initialize counters c\textsubscript{1}, ..., c\textsubscript{b} to 0.
- Generate C\textsubscript{0} = \{E(x\textsubscript{i}, x\textsubscript{i}', c\textsubscript{i}, 1) = E(x\textsubscript{i}, x\textsubscript{i}', c\textsubscript{i}, b)\}, where x\textsubscript{i}, x\textsubscript{i}', i = 1..b are randomly chosen values. Store C\textsubscript{0} indexed on dimension D.
- Initialize the share set S\textsubscript{key} = \{0\}.
- Generate system wide parameters k and m > k and initialize the (k, m) TSS.

\textbf{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. SOTR\textsubscript{V} uses one of the location verification procedures proposed in [2] to verify U’s presence at L and T (see Section ??).

Let U be the i-th user checking-in at V. If the verification succeeds and i ≤ k, S uses the (k, m) TSS to compute a share of p (Benaloh secret key, factor of the modulus n). Let p\textsubscript{i} be the share of p. S sends the (signed) share p\textsubscript{i} to U. If i > k, S calls Setup to generate new parameters for V.

\textbf{CheckIn}(U(p\textsubscript{i}, n, V), V(n, y, C\textsubscript{i-1}, S\textsubscript{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\textsubscript{i-1} is the current set of encrypted counters. SOTR\textsubscript{V} sends C\textsubscript{i-1} to U. Let v, U’s value on dimension D, be within R’s j-th sub-range, i.e., v ∈ R\textsubscript{j}. U runs the following steps:

- Generate b pairs of random values \{(v\textsubscript{1}, v\textsubscript{1}'), ..., (v\textsubscript{b}, v\textsubscript{b}')\}.
- Compute the new encrypted counter set C\textsubscript{i}, where the order of the counters in C\textsubscript{i} is identical to C\textsubscript{i-1}: C\textsubscript{i} = \{RE(v\textsubscript{1}), v\textsubscript{1}', C\textsubscript{i-1}[l]) | l = 1..b, i ≠ j \} ∪ \{RE(v\textsubscript{1}, v\textsubscript{1}', C\textsubscript{i-1}[j] + +). Let C\textsubscript{i} and the signed (by S) share p\textsubscript{i} of p to V.

If SOTR\textsubscript{V} successfully verifies the signature of S on the share p\textsubscript{i}, U and SOTR\textsubscript{V} engage in a zero knowledge protocol ZK-CTR (see Section ??). ZK-CTR allows U to prove that C\textsubscript{i} is a correct re-encryption of C\textsubscript{i-1}: only one counter of C\textsubscript{i-1} has been incremented. If the proof verifies, SOTR\textsubscript{V} replaces C\textsubscript{i-1} with C\textsubscript{i} and adds the share p\textsubscript{i} to the set S\textsubscript{key}. Otherwise, SOTR\textsubscript{V} drops C\textsubscript{i} and rolls back to C\textsubscript{i-1}.

\textbf{PubStats}(V(C\textsubscript{k}, Sh, V), S(p, q)): SOTR\textsubscript{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\textsubscript{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\textsubscript{i} being a correct re-encryption of the set C\textsubscript{i-1}, i.e., a single counter has been incremented. Let ZK-CTR(i) denote the protocol run for sets C\textsubscript{i-1} and C\textsubscript{i}. U and SOTR\textsubscript{V} run the following steps s times:

- U generates random values (t\textsubscript{1}, t\textsubscript{1}'), ..., (t\textsubscript{b}, t\textsubscript{b}'), and random permutation π, then sends to SOTR\textsubscript{V} the proof set P\textsubscript{1} = π\{RE(t\textsubscript{1}, t\textsubscript{1}', C\textsubscript{i-1}[l]) | l = 1..b\}.
- U generates random values (w\textsubscript{1}, w\textsubscript{1}'), ..., (w\textsubscript{b}, w\textsubscript{b}'). It sends to SOTR\textsubscript{V} the proof set P\textsubscript{1} = π\{RE(w\textsubscript{1}, w\textsubscript{1}', C\textsubscript{i-1}[l]) | l = 1..b\}.
- SOTR\textsubscript{V} generates a random bit a and sends it to U.
- If a = 0, U reveals random values (t\textsubscript{1}, t\textsubscript{1}'), ..., (t\textsubscript{b}, t\textsubscript{b}'), and (w\textsubscript{1}, w\textsubscript{1}'), ..., (w\textsubscript{b}, w\textsubscript{b}'). SOTR\textsubscript{V} verifies that for each l = 1..b, RE(t\textsubscript{1}, t\textsubscript{1}', C\textsubscript{i-1}[l]) occurs in P\textsubscript{1} exactly once, and that for each l = 1..b, RE(w\textsubscript{1}, w\textsubscript{1}', C\textsubscript{i-1}[l]) occurs in P\textsubscript{1} exactly once.
- If a = 1, U reveals α\textsubscript{l} = w\textsubscript{l}t\textsubscript{l} and α\textsubscript{l}' = w\textsubscript{l}t\textsubscript{l}' for all l = 1..b along with α, the position in P\textsubscript{1}, and P\textsubscript{1} of the incremented counter. SOTR\textsubscript{V} verifies that for all l = 1..b, l ≠ j, RE(α\textsubscript{l}, α\textsubscript{l}', P\textsubscript{1-1}[l]) = P\textsubscript{1} and RE(α\textsubscript{j}, α\textsubscript{j}', P\textsubscript{1-1}[j]) = P\textsubscript{1}.
- If any verification fails, SOTR\textsubscript{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 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 ??) 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 [?] 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 system. The privacy breach mentioned above is due to the fact that now S can correlate CheckIns 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 Ĉ denote the set of re-encryptions of records of C, where only one record has its counter incremented. To show that ZK-CTR is a ZK proof of C_i ∈ Ĉ_i-1, we need to prove completeness, soundness and zero-knowledge.

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

Proof: If C_i ∈ Ĉ_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 ∈ P_{i-1}.

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

Proof: We need to prove that if C_i /∈ Ĉ_i-1, U cannot convince S unless with negligible probability. For simplicity, we assume C_i /∈ Ĉ_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 to come up 0. Then, U builds P_{i-1}[j] = E(u_{j}, u_j', v_j', c_j, j) and P_i[j] = E(v_{j}, v_j', w_j, c_j', j'). If however a = 1, U has to produce a value α_j such that RE(α_j, E(u_{j}, c_j)) = E(v_j', c_j') or RE(α_j, E(u_{j}, c_j + 1)) = E(v_j', c_j'). In the first case, this means v_j' = v_j + c_j mod n. Without knowing n’s factorization, U cannot compute l’s inverse modulo φ(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 Z_n^*, 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 ≠ 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 [7], [8]. 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 (C, S^*) (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 T_{S^*} and a fixed sequence of random bits on its random tape, R{T^*}. The input x consists of L_0 = C_0 and L_1 = C_1. The content of the input communication tape for S^*, C_{T^*} will consist of tuples (P_{2i}, P_{2i+1}, π_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 R{T^*} and the communication-record tape C{T^*}. 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 ∈ {0, 1}. If a = 0, M^* picks a random permutation π_i, generates t_i, t_i', l = 1..b randomly and computes P_{2i} = π_i{RE(t_i, t_i', 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} = π_i{RE(w_i, w_i', 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 a_i, a_i' randomly, l = 1..b. It then generates a random j ∈ 1..b and computes P_{2i+1} such that for all l = 1..b, l ≠ j, RE(a_i, a_i', P_{2i}[l]) = P_{2i+1}[l] and for the j-th position, RE(a_i, a_i', P_{2i}[j]) = P_{2i+1}[j].

- **Step 2:** M^* sets b = S^*(x, r; P_{0}, P_{1}, π_0, ..., P_{2i-2}, P_{2i-1}, π_{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}, π_j), j = 1..i − 1 and proof P_{2i}, P_{2i+1} on its communication tape C{T^*}. We have the following three cases.

  (Case 1). a = b = 0. M^* can produce t_i, t_i', w_i, w_i', l = 1..b and π_i to prove that P_{2i} = π_i{RE(t_i, t_i', C_{i-1}[l]), l = 1..b} and P_{2i+1} = π_i{RE(w_i, w_i', C_i[l]), l = 1..b}. M^* sets b_i to b; appends the tuple (P_{2i}, P_{2i+1}, π_i, b_i) to C{T^*} and proceeds to the next round (i+1).

  (Case 2). a = b = 1. M^* can produce a_i, a_i', l = 1..b, and index j such that RE(a_i, a_i', P_{2i}[l]) = P_{2i+1}[l], l = 1..b, l ≠ j and RE(a_i, a_i', P_{2i}[j]) = P_{2i+1}[j]. M^* sets
Theorem 4: \( \text{PROFIL-R} \) provides k-privacy.

Proof: (Sketch) Following the definition from Section ?, let us assume that the adversary \( A \) has access to an encrypted counter set \( C_t \) generated after \( C \) has run \textit{Spotter} followed by \textit{CheckIn} on behalf of \( i < k \) different users. The records of set \( C_t \) are encrypted and \( A \) has \( i \) shares of the private key. For any \( j = 1..h \), let \( e_j' \) be \( A \)'s guess of the value of the \( j \)-th counter in \( C_t \). If \( |P_j| = |C_j| = e_j' - 1/(k + 1) = \epsilon \) 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 the \((k, m)\) 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 \( d \in \{0, 1\} \) 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 \textit{Spotter} and \textit{CheckIn} (acting as challenger) with \( A. \) \( B \) re-encrypts all counters from \( A \), except the \( j \)-th one, which it replaces with \( E(u, M_j) \). \( B \) 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 \textit{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 \((k, m)\) 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 \( n = p_0q \). Then, acting as \( i \) different users, \( U_j, j = 1..i \) \( B \) runs \textit{Spotter} with \( S \) (which it also controls) to obtain \( S \)'s signature on \( sh_j \). For each of the \( i \) users, \( B \) runs \textit{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 \textit{PROFIL-R}.

Location correctness: The user’s location is verified in the \textit{Spotter} protocol. A malicious user not present at venue \( V \), is unable to establish a connection with the device deployed at \( V. \) \textit{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 \textit{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 ?? shows this attack succeeds with probability \( 1/2^\ast. \) Second, it attempts to prevent \( V \) from decrypting the counter sets after \( k \) users have run \textit{CheckIn}. This can be done by preventing \textit{SPOTR}_V from reconstructing the private Benaloh key. Key shares are however signed by the provider, allowing \textit{SPOTR}_V to detect invalid shares.

CI-IND Satisfaction: To see that \textit{PROFIL-R} satisfies the CI-IND property, let \( A \) be an adversary that has an \( \epsilon \) advantage in the CI-IND game. We assume a honest challenger, who does not run \textit{Spotter} and \textit{CheckIn} twice for the same (user, epoch) pair. Otherwise, the use of the signed pseudonyms provides an advantage to \( A. \) Note that if pseudonyms are not used, this requirement is not necessary.

No identifying information is sent by users during the \textit{Spotter} and \textit{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 \( A \) to build another adversary \( B \) that has the advantage \( \epsilon \) either against (i) the blind signature protocol [7], 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\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 set up ad hoc networks. Devices establish local connections with neighboring devices and privately compute the instantaneous aggregate LCP of their profiles.

A. Snapshot PROFIL\textsubscript{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\textsubscript{R} consists of three protocols, \{Setup, LCPGen, PubStats\}:

\textbf{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 ??). Send the public key \( n \) and \( y \) to each user \( U_1, ..., U_k \).
- Engage in a multi-party secure function evaluation protocol [?] 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 \( C_0 = \{ E(x_1, x'_1, 0, 1), ..., E(x_k, x'_k, 0, b) \} \), 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 compute the LCP of \( U \)’s LCP.

\textbf{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: \( C_i = \{ E(v_1, v'_1, C_{i-1}[l]) | R_i \mod n | l = 1..b, l \neq j \} \cup \{ E(v_j, v'_j, C_{i-1}[j] + 1) | R_i \mod n \} \) and send it to \( U \).
- Engage in a ZK-CTR protocol to prove that \( C_i \in \hat{C}_{i-1} \). The only modification to the ZK-CTR protocol is that all re-encrypted values are also multiplied with \( R_i \mod n \) and sent to \( U \).

\textbf{PubStats}(\( U(C_i()) \)): 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 \( U_1, ..., U_k \).

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\textsubscript{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 [?] 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\textsubscript{R} to build finer grained personalized safety recommendations, with privacy. PROFIL\textsubscript{R} divides the safety index interval \([(0, 1)]\) into sub-intervals, and associates a counter with each. PROFIL\textsubscript{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 \( Ct \) 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\textsubscript{R} to enable venues to collect fine grained, real time statistics over the profiles of patrons with Yelp accounts. To motivate participation, PROFIL\textsubscript{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.
More than 3000 reviews were left by locals, but far-away customers also form a sizeable percentage.

Figure 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 written by locals, but a large number of reviews were written by far-away visitors.

Figure 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.

Figure 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.

Figure 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, $2^{-100}$, when the Benaloh modulus size ranges from 64 to 2048 bits.

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. Figure 7 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

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 $T_{Set\text{up}} = T_{keysig} + dbT_E + T_{TSS}$. $T_{keysig}$ 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 $\text{store}_{Setup} = dbN$.

We set the $b$ to be 10, Shamir’s TSS group size to 1024 bits and RSA’s modulus size to 1024 bits. Figure 7 shows the Setup overhead on the smartphone and laptop platforms.
ROFIL with privacy and correctness definitions, an expanded version of the (single round) communication overhead, overhead (at a venue). The storage overhead is only a fraction of the number of sub-intervals considered in the statistics computation. Even for $b = 20$, the communication overhead is only 5KB and the communication is 17KB.

The dependency of the communication overhead (in KB) on $b$, when $N = 1024$. Even when $b = 20$, the communication overhead is around 17KB. Figure 2 shows also the storage overhead (at a venue). The storage overhead is only a fraction of the (single round) communication overhead, $2bN$. For a single dimension, with 20 sub-ranges, the overhead is 5KB.

![Performance overhead (KB) vs. Number of ranges in single ZK-CTR round](image)

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.

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 135ms. The communication overhead is exhibited 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 client component is 29ms and the communication component is 106ms. Furthermore, Figure 2 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 note that $P_{ZK}$ -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 [?] 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. [?] 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. [?] 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. [?] 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. [?] 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 [?] 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 private location-centric targeted ads, its main goal is different - to compute location (venue) 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. [?], 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. [?] 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. [?] introduced Persona,
a distributed social network with distributed account data storage. While PROFIL\textsubscript{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\textsubscript{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. [?] 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. [?] 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. CONCLUSIONS**

In this paper we have proposed PROFIL\textsubscript{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\textsubscript{R}. We have shown that PROFIL\textsubscript{R} is efficient, even when executed on resource constrained mobile devices.

**REFERENCES**

Bogdan Carbunar is an assistant professor in the School of Computing and Information Sciences at FIU. Previously, he held various researcher positions within the Applied Research Center at Motorola. His research interests include distributed systems, security and applied cryptography. He holds a Ph.D. in Computer Science from Purdue University.

Mahmudur Rahman is a Ph.D candidate in the School of Computing and Information Sciences at FIU, working under supervision of Dr. Bogdan Carbunar. He received his M.S degree in C.S from FIU in 2012 and Bachelor’s degree in C.S.E from Bangladesh University of Engineering and Technology. He spent 3 years in industry before joining FIU. His research interests are in security and privacy with applications in online and geosocial networks, wireless networks, distributed computing systems and mobile applications. He is particularly interested in studying the tradeoffs between privacy and usability that are achievable in OSNs and strives to provide privacy aware efficient and secure solutions in that context.

Jaime Ballesteros is a PhD candidate in the School of Computing and Information Sciences at FIU. He received his Bachelor’s degree in Computer Science from Universidad Javeriana in Colombia. His research interests are in large scale data management and data analysis on geographical datasets. In particular, he is exploring algorithms for database joins under composite spatial and textual fuzzy constraints and their applications to geographical data analysis and geosocial networks.

Naphtali Rishe is the Author of 3 books on database design and geography; Editor of 5 books on database management and high performance computing; Inventor of 4 U.S. patents on database querying, semantic database performance, Internet data extraction, and computer medicine; Author of 300 papers in journals and proceedings on databases, software engineering, Geographic Information Systems, Internet, and life sciences; Awardee of over $45 million in research grants by Government and Industry, including NASA, NSF, IBM, DoD, USGS; Architect of major industrial projects – both prior to his academic career, and as a consultant since; Founder and Director of the High Performance Database Research Center at FIU (HPDRC); Director of the NSF Center for Research Excellence in Science and Technology at FIU (CREST) and of the NSF International FIU-FAU-Dubna Industry-University Cooperative Research Center for Advanced Knowledge Enablement (IUCRC); Mentor of 70 postdocs, PhDs and MS; the inaugural FIU Outstanding University Professor, Rishe’s TerraFly project has been extensively covered by worldwide press, including the New York Times, USA Today, NPR, Science and Nature journals, and FOX TV News.

Athanasios V. Vasilakos is currently Professor at the University of Western Macedonia, Greece and visiting professor at the National Technical University of Athens, Greece. He has served or is serving as an Editor for many technical journals, such as IEEE TNSM, IEEE TSMC-PART B, IEEE TC, IEEE TITB, ACM TAAS, and IEEE ISAC Special Issues in May 2009, and January and March 2011. He is Chairman of the Council of Computing of the European Alliances for Innovation.