ISSN: 2321-7782 (Online)
Volume 2, Issue 11, November 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com
Special Issue: International Conference on Advanced Research Methodology Held by The International Association for Scientific Research & Engineering Technology, India

VCPROFILE: A Novel Framework for Privacy Preserving in Geosocial Networks

Monika Chilluru[1]
M.Tech Student
Department of Computer Science and Engineering,
Siddharth Institute of Engineering and Technology,

B. Ravindra Naick[2]
Assistant Professor,
Department of Computer Science and Engineering
Siddharth Institute of Engineering and Technology

P. Nirupama[3]
Professor & Head
Department of Computer Science and Engineering,
Siddharth Institute of Engineering and Technology

Abstract: ***Online social networks have become a significant source of personal information. Profit is the main participation incentive for social network providers. Its reliance on user profiles 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) further collect fine grained location information, through check-ins performed by users at visited venues. 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 propose to take first steps toward addressing the conflict between profit and privacy in geosocial networks. We introduce VCPROFILE a novel framework location centric profiles (LCPs). LCPs are statistics built from the profiles of users that have visited a certain location or a set of co-located users. LCP 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. The implementation shows that VCProfile is efficient; the end-to-end overhead is small even under strong privacy and correctness assurances.

Keywords: Privacy Preserving, Geosocial netoworks,

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.

We introduce VCProfile, 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 checkins [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 VCPROFILE, that relieves the GSN provider from a costly involvement in venue specific activities. To achieve this, VCPROFILE 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 VCPROFILE satisfies the introduced correctness and privacy properties. Second, we propose a completely decentralized VCPROFILE extension, built around the notion of *snapshot* LCPs. The distributed VCPROFILE 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 VCPROFILE 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 VCPROFILE, a framework for computing LCPs. Devise both a venue centric and a decentralized solution. Prove that VCPROFILE satisfies the proposed privacy and correctness properties.

» Provide two applications for VCPROFILE:privacy preserving, personalized public safety recommendations and privately building real time statistics over the profiles of venue patrons with Yelp accounts.

» Evaluate VCPROFILE through an Android implementation. Show that VCPROFILE is efficient even when deployed on previous generation smartphones.

The paper is organized as follows. Section II describes related work Section III describes the system and adversary model and defines the problem. Section IV introduces VCPROFILE and proves its privacy and correctness. Section V introduces the notion of snapshot LCPs and presents a distributed, real-time variant of VCPROFILE. Section VI describes two VCPROFILE applications and Section VII concludes.

II. Related Work

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 [11], followed by a significant body of work [12], [13], [14]. We note that VCPROFILE 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. [15] 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 [16] 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. VCPROFILE takes the next step, by enabling the aggregation of user data in a privacy preserving manner.

Mascetti et al. [17] 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. [18] 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. [19] 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, VCPROFILE provides the novel functionality of allowing the provider, venues and even users to privately compute LCPs over visitors or co-located users. VCPROFILE does not require multiple, mutually untrusted servers, or trusted third parties.

Thompson et. al. [20] 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, VCPROFILE does not assume venue owners to be trustworthy. Toubiana et.al [21] proposed Adnostic, a privacy preserving ad targeting architecture. Users have a profile that allows the private matching of relevant ads. While VCPROFILE 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. [22], 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. [23] 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 introduced Persona, a distributed social network with distributed account data storage.While VCPROFILE 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 VCPROFILE 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. [24] 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. [25] 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".

III. 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 $PU = (pU1, pU2, .., pUd\}$ 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 μ at location L, the *location centric profile* at L, denoted by $LCP(L)$ is the set $\{LCP1, LCP2, .., LCPd\}$ where $LCPi$ denotes the aggregate statistics over the i –th dimension of profiles of users from μ . 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, $R1, .., Rb$. 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 $c1, .., cb$; ci records the number of users from μ whose profile value on dimension D falls within range Ri , $i = 1..b$.

b) Private LCP Requirements

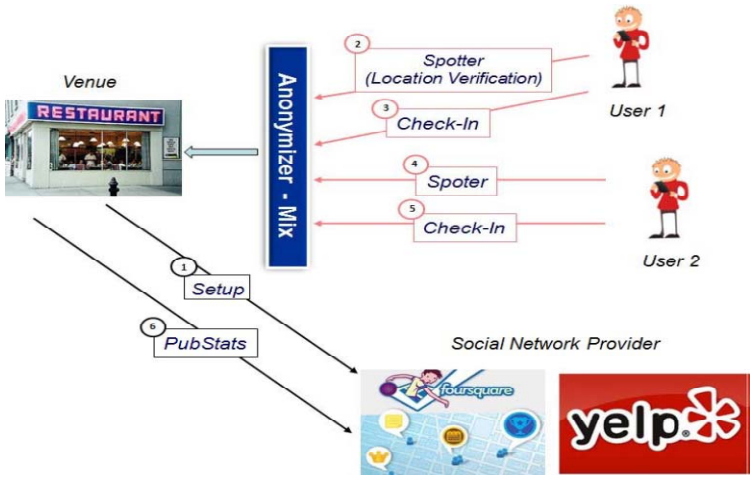

Fig. 1. Solution architecture (k = 2). The red arrows denote anonymous communication channels, whereas black arrows indicate authenticated (and secure) communication channels.

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, $PP(k) = \{Setup, Spotter, Check\ In, PubStats\}$, see Fig. 1. *Setup*is run by each venue where user statistics are collected, to generate parameters for user check-ins. To perform a checkin, 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, *Check In* 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 *Ri* , the counter *ci* is incremented by 1.

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

LCP Correctness: Let *A* denote an adversary that controls the GSN provider and any number of users. Let *C* be a challenger that controls a venue *V*. Let *CV* denote the set of counters at *V* before *A* runs *Check In* at *V* and let *C_ V* be the set of counters afterward. If *C_ V* /∈.*CV* , the *Check In* 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 *q* bits, *b*1, .., *bq* , and sends them to *C*. For each bit *bi* , *i* = 1..*q*, *C* runs *Spotter* followed by *Check In* on behalf of user *Ubi*. At the end of this step, *C* generates a random bit *b* and runs *Spotter* followed by *Check In* on behalf of *Ub* at a venue not used before. *A* outputs a bit *b_*, its guess of *b*. The advantage of *A*, *Adv(A)* = |*Pr*[*b_*=*b*] 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

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 *_T* 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 *Rb*, *b* ∈*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.

Homomorphic Cryptosystems. We use the Benaloh cryptosystem [9], an extension of the Goldwasser-Micali [10].

IV. VCPROFILE

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 *Check In* 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 *Check In*, 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 *Check In* 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 *Ci* denote the set of encrypted counters at *V*, following the *i* -th user run of *Check In*. *Ci* = {*Ci* [1], ..,*Ci* [*b*]}, where *Ci* [*j*] denotes the encrypted counter corresponding to *Rj*, the *j* -th sub-range of *R*. We write *Ci* [*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(uj,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 *Ci* [*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 *Ci* [*j*] or knowing the current counter's value: *Ci* [*j*] + + = [*E(u j , c j)y, E(u_j , j)*] = [*yc j*+1*urj , E(u_j , j)*] = [*E(u j , c j* + 1*), E(u_j, j)*].

In the following we use the above definitions to introduce VCPROFILE. VCPROFILE instantiates *PP(k)*, where *k* is the privacy parameter. The notation *P(A(paramsA), B(paramsB))* 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, .., *cb* to 0.

» Generate *C*0 = {*E(x*1, *x*_1, *c*1, 1*), .., E(xb, x_b, cb, b)*}, where *xi , x_i , i* = 1..*b* are randomly chosen values. Store *C*0 indexed on dimension *D*.

» Initialize the share set *Skey* = ∅.

» 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 *Check In* protocols. SPOTR*V* uses one of the location verification procedures proposed in [8] to verify *U*'s presence at *L* and *T*. 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 *pi* be the share of *p*. *S* sends the (signed) share *pi* to *U*. If *i* > *k*, *S* calls *Setup* to generate new parameters for *V*.

CheckIn(U(*pi*, n, V), V(n, y, *Ci*−1, *Skey*)): 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, *Ci*−1 is the current set of encrypted counters. SPOTR*V* sends *Ci*−1 to *U*. Let *v*, *U*'s value on dimension *D*, be within *R*'s *j* -th subrange, i.e., $v \in Rj$. *U* runs the following steps:

» Generate *b* pairs of random values {*(v*1, *v*_1*), .., (vb, v_b)*}. Compute the new encrypted counter set *Ci*, where the order of the counters in *Ci* is identical to *Ci*−1:*Ci* = {*RE(vl, v_l* ,*Ci*−1[*l*])|*l* = 1..*b, l* _= *j*} ∪ *RE(v j , v_j* ,*Ci*−1[*j*] ++)}.

» Send *Ci* and the signed (by *S*) share *pi* of *p* to *V*.

If SPOTR*V* successfully verifies the signature of *S* on the share *pi* , *U* and SPOTR*V* engage in a zero knowledge protocol ZK-CTR. ZK-CTR allows *U* to prove that *Ci* is a correct re-encryption of *Ci*−1: only one counter of *Ci*−1has been incremented. If the proof verifies, SPOTR*V* replaces *Ci*−1 with *Ci* and adds the share *pi* to the set *Skey*. Otherwise, SPOTR*V* drops *Ci* and rolls back to *Ci*−1.

PubStats(V(*Ck*,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 *Ck*, 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 *Ci* being a correct re-encryption of the set *Ci*−1, i.e., a single counter has been incremented. Let ZK-CTR(i) denote the protocol run for sets *Ci*−1 and *Ci* . *U* and SPOTR*V* run the following steps *s* times:

» *U* generates random values *(t*1, *t*_1*), .., (tb, t_b)* and random permutation π, then sends to SPOTR*V* the proof set *Pi*−1 = π{*RE(tl , t_l*,*Ci*−1[*l*]), *l* = 1..*b*}.

» *U* generates random values *(w*1,*w*_1*), .., (wb,w_b)*. It sends to SPOTR*V* the proof set *Pi* = π{*RE(wl,w_l* ,*Ci* [*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*), .., (tb, t_b)*and *(w*1,*w*_1*), .., (wb,w_b)*. SPOTR*V* verifies that for each *l* = 1..*b*, *RE(tl , t_l* ,*Ci*1[*l*]) occurs in *Pi*−1 exactly once,and that for each *l* = 1..*b*, *RE(wl,w_l* ,*Ci* [*l*]) occurs in *Pi* exactly once.

» If *a* = 1, *U* reveals *ol* = *vlwl t*−1*l* and *o_l*= *v_lw_l t_*−1*l* ,for all *l* = 1..*b* along with *j* , the position in *Pi*−1 and *Pi* of the incremented counter. SPOTR*V* verifies that for all *l* = 1..*b, l* _= *j* , *RE(ol , o_l , Pi*−1[*l*]) = *Pi* [*l*] and *RE(o j , o_j , Pi*−1[*j*]*y)* = *Pi* [*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 VCPROFILE 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 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, *RU,e*. *S* does not sign more than one value for *U* for any epoch. In runs of *Spotter* and *Check In* during epoch *e*, *U* uses *RU,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 *Check In* 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 *Check In*s executed using the same *RU,e*. However, *S* does not know the real user identity behind *RU,e* – due to the use of blind signatures.

V. SNAPSHOT LCP

We extend VCPROFILE 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 VCProfile

We assume a user U co-located with k other users U1, ..,Uk . 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 collocation with U1, ..,Uk . Snapshot VCPROFILE consists of three protocols, {Setup, LCPGen, PubStats}:

Setup(U(r),U1, ..,Uk ()): 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, ..,*Uk* .

» Engage in a multi-party secure function evaluation protocol [19] with *U*1, ..,*Uk* to generate shares of a public value *R* < *n*. At the end of the protocol, each user *Ui* has a share *Ri* , such that *R*1..*Rk* = *R mod n* and *Ri* is only known to *Ui* .

» Assign each of the *k* users a unique label between 1 and *k*. Let *U*1, ..,*Uk* denote this order.

» Generate *C*0 = {*E*(*x*1, *x*_1, 0, 1), .., *E*(*xb, x*_*b*, 0, *b*)}, where *xi* , *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*(*Ci*−1),*Ui* ()): Let *Ci*−1 be the encrypted counters after *U*1, ..,*Ui*−1 have completed the protocol with *U*. *U* sends *Ci*−1 to *Ui* . *Ui* runs the following:

» Generate random values *(v*1*, v_*1*), .., (vb, v_b)*. Let *j* be the index of the range where *Ui* fits on dimension *D*.

» Compute the new encrypted counter set *Ci* as: *Ci* = {*RE(vl, v_l* ,*Ci*−1[*l*])*Ri mod n*|*l* = 1..*b, l* _= *j*} ∪ *RE(v j , v_j* ,*Ci*−1[*j*] ++)*Ri mod n*} and send it to *U*.

» Engage in a ZK-CTR protocol to prove that *Ci* ∈ .*C i*−1. The only modification to the ZK-CTR protocol is that all re-encrypted values are also multiplied with *Ri mod n*, *Ui* 's share of the public value *R*. If the proof verifies, *U* replaces *Ci*−1 with *Ci* .

After completing *LCPGen* with *U*1*, ..,Uk* , *U*'s encrypted counter set is *Ck* = {*E j* = *E(u j , u_j , c j , j)R*1..*Rk* | *j* = 1..*d*}, where *u j* and *u_j* are the product of the obfuscation factors used by *U*1*, ..,Uk* in their re-encryptions. The following protocol enables *U* to retrieve the snapshot LCP.

PubStats(*U(Ck***)):** Compute *E j K*, ∀ *j* = 1..*d*, where *K* = *R*−1 *mod n* (*R* = *R*1..*Rk*), 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*, ..,Uk* . Even though *U* has the private key allowing it to decrypt any Benaloh ciphertext, the use of the secret *Ri* values prevents it from learning the profile of *Ui* , *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.

VI. APPLICATIONS

We now propose two VCPROFILE application.

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 VCPROFILE to build finer grained personalized safety recommendations, with privacy. VCPROFILE divides the safety index interval ([0*,* 1]) into sub-intervals, and associates a counter with each. VCPROFILE 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 VCPROFILE to enable venues to collect fine grained, real time statistics over the profiles of patrons with Yelp accounts. To motivate participation, VCPROFILE 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.

VII. CONCLUSION

In this paper, we propose to take first steps toward addressing the conflict between profit and privacy in geosocial networks. We introduce VCPROFILE a novel framework location centric profiles (LCPs). LCPs are statistics built from the profiles of users that have visited a certain location or a set of co-located users. LCP 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. The implementation shows that VCProfile is efficient; the end-to-end overhead is small even under strong privacy and correctness assurances.

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 personallyidentifiable information via online social networks," Comput. Commun.Rev., vol. 40, no. 1, pp. 112–117, 2010.
4. Foursquare Official Blog, New York, NY, USA. (2011). On Foursquare, Cheating, and Claiming Mayorships from your Couch [Online]. Available:http://goo.gl/F1Yn5
5. (2012). Raspberry Pi. An ARM GNU/Linux Box for $25. Take a Byte [Online]. Available: http://www.raspberrypi.org/
6. G. Coley, Beagleboard System Reference Manual. Dallas, TX, USA,BeagleBoard. Org., Dec. 2009.
7. 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.
8. J. Benaloh, "Dense probabilistic encryption," in Proc. Workshop Sel. Areas Cryptograph., 1994, pp. 120–128.
9. 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.
10. D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Commun. ACM, vol. 24, no. 2, pp. 84–90, 1981.
11. M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in Proc. MobiSys, 2003, pp. 31–42.
12. 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.
13. 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.
14. 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.
15. 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.
16. K. P. N. Puttaswamy and B. Y. Zhao, "Preserving privacy in locationbased mobile social applications," in Proc. 11th Workshop Mobile Comput. Syst. Appl., New York, NY, USA, 2010, pp. 1–6.
17. 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
18. 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.
19. 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.
20. B. Thompson, S. Haber, W. G. Horne, T. Sander, and D. Yao, "Privacypreserving computation and verification of aggregate queries on outsourced databases," in Proc. 9th Int. Symp. Privacy Enhancing Technol., 2009, pp. 185–201.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.

