Private Geosocial Networking

Bogdan Carbunar
School of Computing and Information Sciences
Florida International University
Miami, FL
carbunar@cs.fiu.edu

Radu Sion
Computer Science Department
Stony Brook University
Stony Brook, NY
sion@cs.stonybrook.edu

ABSTRACT

Location based social or *g*eosocial networks (GSNs) have recently emerged as a natural combination of location based services with online social networks: users register their location and activities, share it with friends and achieve special status (e.g., "mayorship" badges) based on aggregate location predicates. Boasting millions of users and tens of daily check-ins, such services pose significant privacy threats: user location information may be tracked and leaked to third parties. Conversely, a solution enabling location privacy may provide cheating capabilities to users wanting to claim special location status. In this paper we introduce new mechanisms that allow users to (inter)act privately in today's geosocial networks while simultaneously ensuring honest behaviors. We show that our solutions are efficient both on the provider and the client side.

1. INTRODUCTION

Location based services (LBS) offer information and entertainment services to mobile users, that rely on the geographical position of their mobile devices. A recently introduced but popular example, is the geosocial network (GSN) – social networks centered on the geographical position of their users. Services such as Foursquare [1], SCVNGR [2], Gowalla [3] or Groundmap [4] allow users to register or "check-in" their location, share it with their friends, leave recommendations and collect prize "badges". Badges can be acquired by checking-in at certain locations, in ways conforming to a pre-defined pattern, simultaneously with other users, or the largest number ("mayor" badge).

An important problem, that can hinder wider scale adoption, is compromised location privacy. Service providers learn the places visited by each user, the times and the sequence of visits as well as user preferences (e.g., places visited more often). The implications are significant as service providers may use this information in ways that the users never intended when they signed-up (e.g., having their location information shared with third parties [5, 6]).

While compromised privacy may seem a sufficient reason to avoid the use of such services, here we argue this is not necessary. Instead, we propose a framework where users themselves store and manage their location information. The provider's (oblivious) participation serves solely the goal of ensuring user correctness. This enables users to privately and securely check in and acquire special location based status, e.g., in the form of badges. Badges are defined as aggregate predicates of locations. Solutions can then be devised to support a variety of such predicates, including (i) registering a pre-defined number of times at a location or set of locations, (ii) registering the most number of times (out of all the users) at a location and (iii) simultaneously registering with k other users at a location. Given the recent surge of location privacy scandals and the associated liabilities [7], we believe that implementing such solutions is also in the service provider's best interest.

To this end, the problem has two main facets. On one side, clients need strong privacy guarantees: The service provider should not learn user profile information, including (i) linking users to (location,time) pairs, (ii) linking users to any location, even if they achieve special status at that location and (iii) building user profiles – linking multiple locations where the same user has registered. On the other side, when awarding location-related badges the service provider needs assurances of client correctness. Otherwise, since special status often comes with financial and social perks, clients have incentives to report fake locations [8], copy and share special status tokens, or check-in more frequently than allowed.

In this work we introduce three privacy mechanisms to the aggregate location predicate problem. In *GeoBadge*, a user can privately prove k check-ins at one site or a pre-defined set of sites, where k is a predefined parameter. *GeoM* extends *GeoBadge* with provably time-constrained check-ins as well as arbitrary values for k. Finally, *MPBadge* extends *GeoBadge* with the notion of simultaneous, co-located check-ins from multiple users. The complexity of *MPBadge* lies in the seeming contradiction between the ability of multiple clients to anonymously check-in at the same location and the ability of rogue users to launch Sybil attacks [9].

We have implemented and evaluated the performance of the *GeoBadge* and *GeoM* protocols on Motorola Android smartphones as well as a laptop hardware. Experimental results are extremely positive. A single laptop allows the provider to support hundreds of check-ins per second, while a smartphone can build strongly secure aggregate location and correctness proofs in just a few seconds.

2. RELATED WORK

Location Cloaking: Location and temporal cloaking techniques, or introducing small errors in location reports in order to provide 1-out-of-k anonymity have been initially proposed in [10], followed by a significant body of work [11, 12, 13, 14]. These techniques are vulnerable to intersection attacks: the address of a user that frequently reports a residential address may be identified by computing the intersection of the cloaked reports.

Location Verification: Saroiu and Wolman [15] introduced the location proof concept – a piece of data that certifies a receiver to a geographical location. The solution relies on special access points (APs), that are able to issue such signed proofs. Luo and Hengartner [16] extend this concept with client privacy, achieved with the price of requiring three independent trusted entities. Note that both solutions rely on the existence of specialized APs or celltowers, that modify their beacons and are willing to participate and sign arbitrary information. To address the central management problems, Zhu and Cao [17] proposed the APPLAUS system, where co-located, Bluetooth enabled devices compute privacy preserving location proofs.

Proximity Alerts: Zhong et al. [18] have proposed three protocols that privately alert participants of nearby friends. Location privacy here means that users of the service can learn a friend's location only if the friend is nearby. Manweiler et al. [19] propose several cloaking techniques for private server-based location/time matching of peers. Narayanan et al. [20] proposed several other solutions for the same problem, introducing the use of location tags as a means to provide location verification.

Summary: Existing work has focused on (i) hiding user location from LBS providers and other parties and on (ii) enabling users to prove claimed locations. In this paper we consider the next step, of anonymizing location aggregates defined by geosocial networks.

3. MODEL

The System: The geosocial network (GSN) consists of a provider, S, hosting the system and serving a number of subscribers. To use the provider's services, a client application needs to be downloaded and installed. Subscribers can then register and receive initial service credentials, including an unique user id; let Id_A denote the id of user A. In the following we use the terms *user* and *subscriber* to refer to users of the service and the term *client* to denote the software provided by the service and installed by users on their devices.

The provider supports a given set of locations, defined in terms of discrete points-of-interests (POIs) or sites: restaurants, bars, movie theaters, etc. We refer to such POIs as sites or venues. Users can check-in through their clients at specific sites: given the device's GPS location, the client presents the user with a list of matching, proximity sites. The client then selects the site, for which the check-in is then executed.

The time is divided into epochs. For instance, Foursquare [1] supports one day long epochs. Users are restricted to a single check-in per site per epoch.

A full-fledged privacy solution is composed of a set of protocols $Geo = \{Setup, RegisterSite, Subscribe, CheckIn, StatVerif\}$. The Setup function generates the system wide parameters, including keys. RegisterSite is executed by a client to register a new site with the system. Subscribe is executed once by any client C that wants to register with the service. CheckIn is run by a subscribed client that wants to report its location at a certain time. StatVerif is a protocol that enables the client to achieve special status/badge for a given site. We consider three types of special status tokens: (i) the location badge, issued when the client runs check-in at the same site during k different epochs, (ii) the multiplayer badge, when s users run check-in simultaneously for the same site and (iii) the mayor badge, issued when the client has the largest number of check-in runs, at most one per epoch, in the past m epochs at a given venue.

Server Concerns. The provider S is honest, yet curious. S is interested in collecting tuples of the format (Id, P, T), where Id is a user id, P is a site and T is a time value. To this end, it may collude with existing clients and generate Sybil clients to track users of interest. Finally, the provider has no interest in colluding with users to issue badges without merit. To achieve privacy, intuitively, the provider should learn nothing about *Geo* clients. First, this includes the sites at which users run the *CheckIn* function, how many times and when they run *CheckIn* (in total and for any site). We note that this necessarily includes also hiding correlations between sites where a given client has run *CheckIn*.

Client Concerns. The client is assumed to be malicious. Malicious clients can be outsiders that are able to corrupt existing devices or may be insiders - subscribers, users that have installed the client. Malicious clients can try to cheat on their location (claim to be in a place where they are not [8]), attempt to prove a status they do not have, or disseminate credentials received from the server to other clients. The latter case includes any information received from the server, certifying presence at a specific location.

4. GEO-BADGE

We now introduce *GeoBadge*, a private protocol that allows users to prove having visited the same location k times. In a nutshell, *GeoBadge* works as follows: each subscribed client contacts the provider over the anonymizer Mix, authenticates anonymously, proves its current location and obtains a blindly signed, single use nonce and a share of a secret associated with the current site. When k shares have been acquired (after k check-ins at the same site) the client is able to reconstruct the secret - which is the proof required for the badge of the site. The single use nonces prevent users from distributing received shares (or proofs).

During Setup, S chooses a large prime p and generates a random key K. S publishes p and keeps K secret. During a RegisterSite call, the client that registers a new site is called the *o*wner of the site. S generates a secret M_P randomly and uses a threshold secret sharing solution to compute shares of M_P. S publishes the number of shares required to receive a badge at the site, along with the verification value $V_P = H(M_P H_K(P) \ mod \ p)$. In order to subscribe, a client runs Subscribe over an anonymizer with S, in order to obtain tokens that allow it later to authenticate anonymously with the server (see Boneh and Franklin [21]). The reason for using Mix is to hide C's location from S.

During a CheckIn, the client anonymously proves to S that it is a subscriber (see Boneh and Franklin [21]). It then uses techniques detailed in Section 2 (e.g., [15, 16, 17]) to prove its location to S. If the location is certified, S generates a share of the secret associated with the check-in site and sends it to the client. The client collects shares and when it detects having reached a badge status, it initiates a run of StatVerif. Specifically, the client aggregates its shares to reveal the secret M_P of the site P and sends it to S. Note that to prevent clients from sharing and re-using secrets, during the CheckIn process the client and server run a blind signature protocol: the client obtains a signed random value from S, to which S does not have access. During StatVerif, the client needs to provide also k values signed by S, along with M_P. Since S keeps a record of seen signed values, clients cannot "double-spend" them.

The use of anonymizers, of shares that are aggregated into the secret associated with the site, along with blind signatures, prevent the server from learning the identity of the client or of identifying and linking the check-ins that lead to the badge. Moreover, the clients cannot achieve badge status illegally. New shares cannot be derived by clients from existing ones and users cannot run check-ins at sites where they are not located.

5. GEO-M

Using the Foursquare terminology, the user that has run $CheckIn$ the most number of times, at a site S, within the past m epochs, becomes the mayor of the place. In this section we propose $GeoM$, a solution that allows users to achieve this status with privacy, while allowing anyone to verify this fact. $GeoM$ extends $GeoBadge$ with several features. First, it allows clients to prove any number of check-ins, not just a pre-defined value k. Second, the check-ins are time constrained: clients have to prove that all check-ins have occurred in the past m epochs. Finally, client issued proofs can be published by the provider to be verified by any third party, without the danger of being copied and re-used by other clients.

$GeoM$ achieves these features by requiring the service provider to issue only one token for each site during any epoch. When a user has accumulated k tokens for a site, it proves to the provider that it has k out of the m tokens given in the past m epochs for that site. The proof is in zero knowledge (ZK) and if it verifies is published by the server.

During Setup, the server generates two large safe primes p and q which it keeps secret and the composite $n = pq$ that is made public. In addition to its functionality from GeoBadge, RegisterSite requires S to initialize a random number generator for each new site. Then, during each epoch, S generates a random token, keeps it secret, but publishes its square modulo n, which are quadratic residues. Whenever a user runs CheckIn (following similar steps to Geo-Badge) and succeeds in authenticating and verifying its location, S sends it the square root of the published value during the current epoch (effectively the random token generated for that epoch). When the client accumulates enough tokens to become mayor (more tokens that anyone else), the client initiates the StatVerif procedure. During StatVerif, the client cannot send the accumulated tokens as that would leak the epochs when its check-ins occurred. Instead, the client builds zero-knowledge proofs of knowledge of the tokens - of square roots of published quadratic residues.

The zero-knowledge proofs enable the client to prove to the server with high probability, knowledge of enough CheckIn run outputs for the desired site. The server however does not learn anything else, for instance the times when the check-ins occurred.

6. MULTI-PLAYER: MP-BADGE

The multi-player badge is issued when a user presents proof of co-location and interaction with $k-1$ other users at a site P. k is a parameter that may depend on the site P. We now present $MPBadge$, an extension of $GeoBadge$ that provides this functionality with privacy. $MPBadge$ relies on threshold signatures, where each client is able to provide a signature share and k unique signature shares generated at the same site in the same epoch can be combined to produce a signed co-location proof. An additional difficulty here lies in the ability of an anonymous user to cheat: run $CheckIn$ multiple times in the same epoch, obtain k signature shares and generate by itself the co-location proof.

We solve this issue by allowing a user to run $CheckIn$ only once per site per epoch. For this, we require each user to obtain a blind signature from S, for each supported site, once per epoch. When the client runs CheckIn with S, besides authenticating anonymously and proving its location, it sends the blind signature corresponding to the check-in site. The client cannot obtain more than a blind signature per site and S scans for and penalizes duplicate uses. If the verifications succeed, S sends the client a share of a secret generated for the site during the current epoch.

After performing the CheckIn operation, the client needs to identify co-located clients (at least k-1 of them). This is performed in MP-CheckIn procedure, where each client uses its Wi-Fi in ad hoc mode, set to a default ssid, to identify other clients and initiate contact. When a co-located client is identified, the client shares its share of the secret revealed by S during CheckIn, as well as its value that was blindly signed by S for the site and the current epoch. When at least k clients run this step, each client is able to recover the secret of the site from the shares and send the secret, along with the blindly signed accumulated values to S - during StatVerif. S verifies that the secret revealed is correct and that the exact set of k revealed blind signatures has not been used before more than k-1 times. S records the set of k blind signatures and allows it to be used only k times. Subsequent uses of the tokens are allowed, as long as the newly revealed set contains at least one fresh blind signature.

The use of the blindly signed shares prevents a client from generating multiple signatures for the same site and epoch. It however does not prevent sybil attacks, where the attacker controls multiple client subscriptions and devices.

7. EVALUATION

In this section we study the efficiency of our solutions from the standpoint of both service provider (server) and client. To this end we have implemented $GeoBadge$ and $GeoM$ n Android and Java. We have executed the protocol client side on Motorola Milestone smartphones, with an ARM Cortex A8 600 MHz CPU and 256 MB RAM, running Android 2.1. The server side was run on HP Compaq 8510w laptops with an Intel Core 2 Duo T7500 Processor of 2.2GHz and 4MB RAM. All the results shown in the following are computed as an average over at least 10 independent runs.

In the first experiment we study the performance of GeoBadge in terms of k, the required number of CheckIn runs. We set

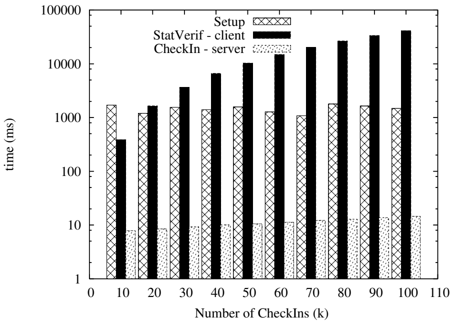

Figure 1: *GeoBadge* dependence on check-in count.

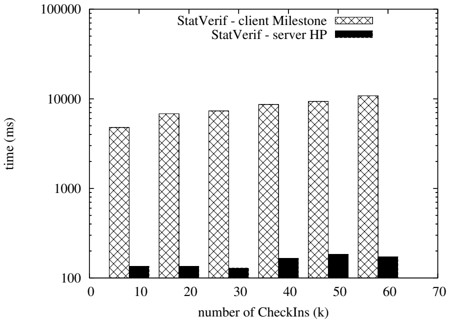

Figure 2: *GeoM* dependence on k, the number of check-ins.

the modulus size to 1024 bits. Figure 1 shows our results. The Setup cost does not depend on k. It does however depend on the modulus size: the cost of generating a prime. The client side StatVerif cost exhibits a quadratic dependency on k, as the reconstruction formula has k factors and each Lagrange coefficient has k components. For k=100, this cost is almost 42s. As expected, the CheckIn cost exhibits a linear dependency on k, but is small: even for polynomials of degree 99, the server can run 70 CheckIns per second.

We now evaluate $GeoM$. Figure 2 shows the performance of $StatVerif$ (client and server side) in ms, as a function of k, the number of mayorship check-ins. N, the modulus bit size is set to 1024, m, the number of past epochs considered is set to 60 and s, the number of proof sets in the ZK proofs is set to 40. The y axis is shown in logarithmic scale. The server side exhibits a small linear increases with k, but is only 170ms when $k = m = 60$ (one check-in in each of the previous 60 epochs). The client side is slower, with up to 10s required ($k = m = 60$) on the smartphone but only 0.9s when executed on the laptop.

8. CONCLUSIONS

In this paper we study privacy issues in achieving aggregate location predicates in GSNs. We introduce several new privacy and correctness properties and propose solutions that privately and securely build a variety of aggregate location predicates. Smartphone implementations prove the solutions to be practical.

9. REFERENCES

[1] Foursquare. `https://foursquare.com/`.

[2] SCVNGR. `http://www.scvngr.com/`.

[3] Gowalla. `http://gowalla.com/`.

[4] Groundmap. `http://www.groundmap.com/`.

[5] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. In *WOSN*, pages 7–12, 2009.

[6] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. *Computer Communication Review*, 40(1):112–117, 2010.

[7] Josh Lowensohn. Apple sued over location tracking in iOS. Cnet News. `http://news.cnet.com/8301-27076_3-20057245-248.html/`, 2011.

[8] Gpscheat! `http://www.gpscheat.com/`.

[9] John R. Douceur. The Sybil Attack. In *IPTPS*, pages 251–260, 2002.

[10] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of MobiSys*, 2003.

[11] Baik Hoh, Marco Gruteser, Ryan Herring, Jeff Ban, Dan Work, Juan-Carlos Herrera, Re Bayen, Murali Annavaram, and Quinn Jacobson. Virtual Trip Lines for Distributed Privacy-Preserving Traffic Monitoring. In *Proceedings of ACM MobiSys*, 2008.

[12] Femi G. Olumofin, Piotr K. Tysowski, Ian Goldberg, and Urs Hengartner. Achieving Efficient Query Privacy for Location Based Services. In *Privacy Enhancing Technologies*, pages 93–110, 2010.

[13] Xiao Pan, Xiaofeng Meng, and Jianliang Xu. Distortion-based anonymity for continuous queries in location-based mobile services. In *GIS*, pages 256–265, 2009.

[14] Gabriel Ghinita, Maria Luisa Damiani, Claudio Silvestri, and Elisa Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *GIS*, pages 246–255, 2009.

[15] Stefan Saroiu and Alec Wolman. Enabling New Mobile Applications with Location Proofs. In *Proceedings of HotMobile*, 2009.

[16] W. Luo and U. Hengartner. VeriPlace: A Privacy-Aware Location Proof Architecture. In *Proceedings of ACM SIGSPATIAL GIS*, 2010.

[17] Z. Zhu and G. Cao. APPLAUS: A Privacy-Preserving Location Proof Updating System for Location-based Services. In *Proceedings of IEEE INFOCOM*, 2011.

[18] Ge Zhong, Ian Goldberg, and Urs Hengartner. Louis, Lester and Pierre: Three Protocols for Location Privacy. In *Proceedings of PETS*, 2007.

[19] Justin Manweiler, Ryan Scudellari, Zachary Cancio, and Landon P. Cox. We saw each other on the subway: secure, anonymous proximity-based missed connections. In *Proceedings of HotMobile*, 2009.

[20] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *Proceedings of NDSS*, 2011.

[21] Dan Boneh and Matt Franklin. Anonymous Authentication With Subset Queries (Extended Abstract). In *in Proceedings of CCS*, 1999.