Private Badges for Geosocial Networks Used In Real Time Application

R.Krithiga[1], V.Muthamilselvi[2]
[1]Student, [2]Assistant professor
Valliammai Engineering College,Chennai

***Abstract* - Geosocial networks (GSNs) extend classic online social networks with the concept of location. Users can report their presence at venues through "check-ins" and, when certain check-in sequences are satisfied, users acquire special status in the form of "badges". We first show that this innovative functionality is popular in Foursquare, a prominent GSN. Furthermore, we address the apparent tension between privacy and correctness, where users are unable to prove having satisfied badge conditions without revealing the corresponding time and location of their check-in sequences. To this end, we propose several privacy preserving protocols that enable users to prove having satisfied the conditions of several badge types. Specifically, we introduce (i) Geo-Badge and T-Badge, solutions for acquiring location badges, (ii) Freq-Badge, for mayor ship badges, (iii) e-Badge, for proving various expertise levels.**

***Index Terms* - Geosocial networks, privacy, security.**

I. INTRODUCTION

LOCATION Based Services (LBS) provide users with information and entertainment applications centered on their geographical position. A recently introduced but popular LBS are Geosocial Networks (GSNs), social net- works centered on the locations of users and businesses. GSNs such as Foursquare [1] and Yelp [2] allow users to register or "check-in" their location, share it with their friends, leave recommendations and collect prize "badges". Badges are acquired by checking-in at certain locations (i.e., venues), following a required pattern. Besides keeping track of the locations of their friends, users rely on GSNs to receive promotional deals, coupons and personalized recommendations. For GSN providers however, the main source of revenue is location-based ad targeting. Boasting millions of users [3] and tens of mil- lions of location check-ins per day [4], GSNs can provide personalized, location dependent ads. The more user information they are able to collect, the more accurate are their predictions. Thus, the price of participation for users is compromised privacy, in particular, 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., the frequency distribution of their visits) [5], [6]. The service providers may use this information in ways the users never suspected when they signed-up (e.g., having their location shared with third parties [7], [8]). Opting out of GSN services seems to be a rational way to avoid compromised privacy (allowing stalking, theft [9]). In this paper however, we show that such radical measures may not be necessary. To this end, we introduce a frame- work that enables users to privately acquire GSN badges. In this framework. Users are responsible for storing and managing their location information, and the provider's (oblivious) participation serves solely the goal of ensuring user correctness.

The challenge consists of providing solutions that balance three requirements. On one dimension, clients need strong privacy guarantees. The service provider should not learn user profiles 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 even (iii) building pseudonymous user profiles – linking multiple locations where the same "unknown" user has checked- in. On the second dimension, the service provider needs assurances of client correctness when awarding location- related badges. Otherwise, since special status often comes with financial and social perks, privacy would protect users that perpetrate fraudulent behaviors such as, reporting fake locations [11], duplicating and sharing special status tokens, or checking-in more frequently than allowed. On a third dimension, the provider needs to be able to collect certain user information. Being denied access to all user information discourages participation. The use of client pseudonyms to provide client privacy during check-ins and special status requests is vulnerable to profiles based deanonymization attacks [12], [13].Constructed pseudonymous profiles can be joined with residential and employment datasets to reveal profiles owner identities.

II. MODEL

In the System We consider a geosocial network provider, S, which we model after the most popular in existence to date, Foursquare [1]. Each subscriber (or user) has an account with S. Subscribers are assumed to have mobile devices equipped with a GPS receiver and a Wi-Fi interface (present on most Smartphone). To use the provider's services, a client application needs to be downloaded and installed. Subscribers can register and receive initial service credentials, including a unique user id; let IdA 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. Besides users, the geosocial network also supports a set of venues, which are businesses with a geographic location. Let V denote the set of all the venues registered with the system. Users report their location, through check-ins at venues of interest, share it with friends (e.g., imported from Facebook or discovered and invited on Foursquare) and are awarded points and "badges". A user with more check-in days at a venue than

anyone else in the past 60 days becomes the "Mayor" of the venue. Foursquare has partnered with a long list of venues (bars, cafes, restaurants, etc) to reward the Mayor with freebies and specials. Foursquare imposes a discrete division of time, in terms of epochs. A user can check-in at one venue at most once per epoch. This strategy has made Foursquare quite popular, with a constantly growing user base, which we currently estimate at over 20 million users.

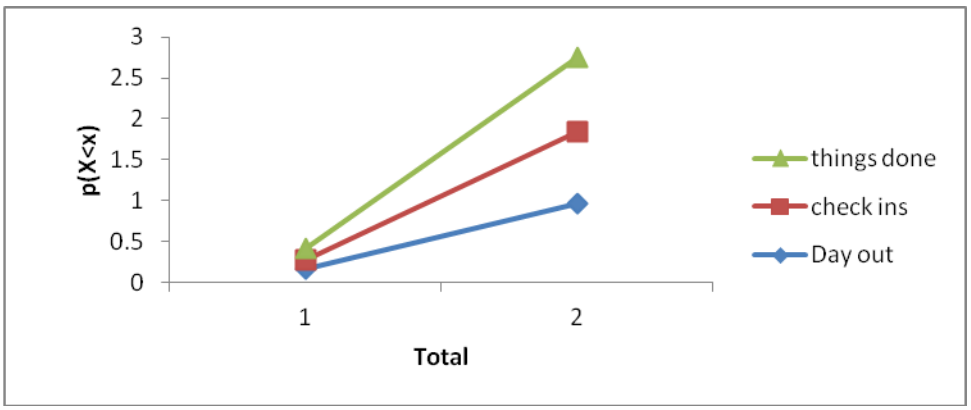
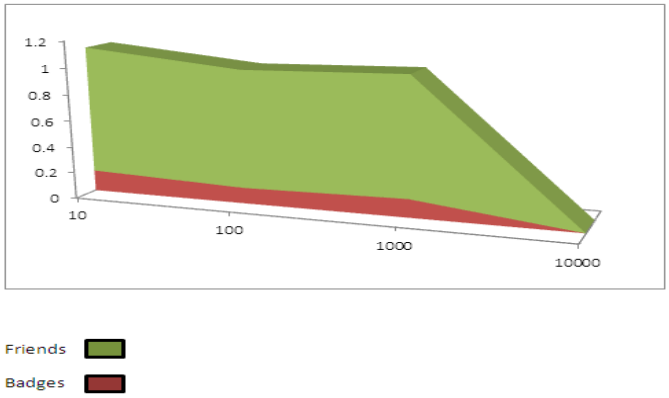


Fig.1. Foursquare stats. (a) CDF of days out, check-ins and things done by users.



(b) Badge and friends evaluation.

Foursquare Data : In order to understand the need for our solutions, we have collected profiles from 781,239 randomly selected Foursquare users. For every user, we have gathered the user profile including the total number of friends, the total number of check-ins, the total number of days the user was out (days the user was actively performing check-ins) and the total number of things done (e.g., reviews left for a venue). Our first question was how active are Foursquare users. Fig. 1(a) shows the CDF of the number of check-ins, days out (days the user was actively performing check-ins) and things done (e.g., reviews left for a venue) by users. Note that 45% of the collected users have between 80 and 950 check-ins, for between 50 and 300 days of activity (at this time Foursquare is 2 years and a half old). This shows that many foursquare users are very active. Our second question regards the popularity of badges in geosocial networks. Fig. 1(b) shows the cumulative distribution function (CDF) of the number of badges earned by users as well as their friends. Note that 45% of the users (between the median and the 95th percentile) have between 10 and 50 badges and between 20 and 95 friends. This, coupled with the large numbers of reported check-ins, leads us to conclude that foursquare is a system worthy to evaluate our protocols. To corroborate the check-in data in a location-aware fashion, we used a Foursquare feature that allows users to query the list of venues at a location using (latitude, longitude) pairs. Specifically, we started with a seed latitude and longitude (in our case, 40.000, -73.000, representing New York City). We then generated 5000 random coordinates around this coordinate pairs. For each newly generated coordinate pair, we queried Foursquare to collect all the venues near that location. Fig. 2(a) shows the scatter plot of check-ins vs. users in one of the most active locations in our dataset, the city of Babylon in Long Island, NY. Each point on the plot denotes a venue, the x axis shows the total number of check-ins recorded at the venue and the y axis shows the total number of users that have performed the check- ins. Note that a few venues record 1000-5000 check-ins, from more than 500 users. Most venues however range from a few tens to a few hundred check-ins and users. Finally, Fig. 2(b) shows the evolution between August 2010 and February 2011 of the number of check-ins per day for two randomly selected venues. The number of check-ins range between 3 to almost 70 per day. Our conclusions are that Four square users are actively checking-in and venues record many daily check-ins. This data rich environment can be a goldmine for rogue GSN providers. Moreover, the number of recorded check-ins suggests that badges and mayor ship are likely to become objects of contention. Thus, devising private and secure "badging" protocols becomes a problem of primary importance for GSNs.

Geo: A Framework for Private GSNs A full-fledged private GSN solution is composed of a set of protocols Geo = {Setup, RegisterVenue, Subscribe, CheckIn, StatVerify, ProveBadge}, described in the following. We use the notation Prot(P1(args1),..,Pn(argsn)) to denote protocol Prot run between participants P1,..,Pn, each with its own arguments. Setup(S()):

Executed initially (only once) by the service provider S. The server produces public information pubS and private information privS. The server publishes pubS. RegisterVenue(O(V),S(privS)): Executed by the owner O to register a new venue V with the provider. Subscribe(C(),S(pubS,privS)): Executed once by any client C that wants to register with the service. If the subscription fails, the server returns -1. Otherwise, the client receives a unique id and the server's public information pubS. CheckIn(C(Id,V,T,pubS),S(privS)): Executed by a sub- scribed client with identifier Id, to report location V at time T to the provider S. S verifies the correctness of V and T and returns -1 in case of failure. Otherwise, the client is issued a special token proving its presence at V during T.

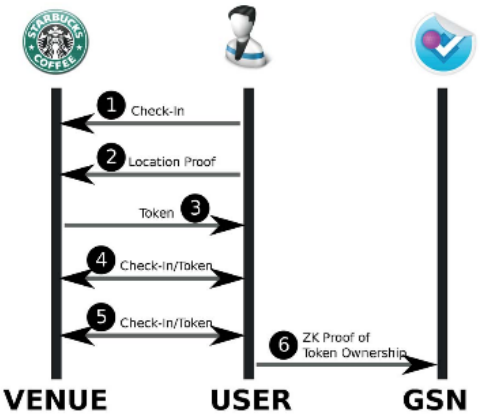


Fig.3. High level overview of a private badge protocol.

Stat Verify(C(Id,V,k,Tk,pubS),S(privS)): After accumulating sufficient tokens, the client runs Stat Verify with the server, for a specific venue V, providing its entire set of tokens, Tk. If the tokens prove that special status has indeed been achieved, the server issues a special status token (or badge), BV, to the client. We support several badge types, introduced by Foursquare.

III. VARIOUS BADGES

- Location Badge (Geo-Badge/T-Badge). Geo-Badge is issued after the client runs CheckIn during k different epochs at a venue V. T-Badge is issued after the client runs CheckIn at k different venues. Geo-Badge and T- Badge model Foursquare badges such as "Newbie", "Local", "Adventurer", "Explorer" and "Superstar"
- Expert Badge (e-Badge). e-Badges support several levels of expertise. To achieve level 1 of expertise, the client needs to run CheckIn at k different, select locations, with a common background. A user having expertise level L for an e-Badge can reach level L+1 after performing k more check-ins at similar (but different) locations. k is a system parameter. This models several expertise badges from Foursquare, where the rules are the same for all the areas of expertise: A user achieves level 1 for checking in at five unique places. From there, every level up is five more unique places.
- Mayor ship (Freq-Badge). Issued when the client has performed the largest number of CheckIns, at most one per epoch, in the past m epochs at a given venue. m is a system parameter. Freq-Badge models Foursquare "mayor" badges.
- Multi-Player Badge (MPBadge). Issued when the client runs CheckIn simultaneously with s other users at the same location. s is a system parameter. The MPBadge models foursquare badges.
- ProveBadge(C1(pubS,V,BV),C2(pubS,V),S(privS,V)): This protocol enables client C1 to prove ownership of a badge BV for a venue V to another client C2. In order to preserve the privacy of C1, following the ProveBadge execution, C2 should not learn additional information about C1 and should not be able to prove ownership of the badge to another client.

IV. PRIVACY AND CORRECTNESS PROPERTIES

Server Side We consider a provider S that follows the protocols correctly. This implies for instance that the provider will not hand out incorrect information to users. However, we assume that S is interested in collecting tuples of the for- mat (Id,V,T), where Id is a user id, V is a venue and T is a time value. In order to achieve this goal, S may collude with venues and existing clients and generate Sybil clients to track users of interest. The provider however does not collude with users to issue badges without merit. We do not consider physical attacks, such as, the server physically tracking individual users. Intuitively, to achieve privacy, the provider should learn nothing about Geo clients, including the venues and times at which a user runs the Check-In function, as well as her total and per-venue Check-In counts. We note that this necessarily includes also hiding correlations between venues where a given client has run Check-In. We formalize this intuition using games run between an adversary A and a challenger C. A controls the service provider, the set of venues and any number of clients, thus controls the initial parameter generation functionality (e.g., the Setup function). Shares public parameters with C. C controls two clients C0 and C1. C initially runs the Subscribe function with A for the two clients and obtains their unique identifiers. In a first CheckIn-Indistinguishability game, we model the adversary's inability to distinguish between clients during Check-In executions, even when the adversary controls an initial trace of Check-In executions. We use the notation Cb(args) or Cci to denote either client C0 or client C1 (according to the value of the bit b or ci), using input values args.

CheckIn Indistinguishability(CI-IND): A generates pub- lic information pubA (and corresponding private information priv A), generates l bits c1,..,cl, and l + 1 venue ids V1,..,Vl,Vl+1, Vi ∈ V, i=1..l+1, and sends them to C. For each i = 1..l, C needs to run Check-In on behalf of client Cci, at venueVi. C verifies that the time between two consecutive requests for the same client is sufficient to enable the client to travel the distance between the corresponding venues. If this condition is not satisfied, C ignores the request. Otherwise, it exe- cutes Check-In(Cci(Idci,Vi,Ti,pubA),A(privA)). After processing the l requests, C makes sure that the distance between both C0 and C1's last check-ins to venue Vl+1 can be physically traversed between the time of their last check-ins and the current time. If the verification fails, C stops the game. Otherwise, C generates a bit b ∈{ 0,1} and runs Check-In(Cb(Idb,Vl+1,Tl+1,pubA,A(priv A)).A outputs a bit b. A solution provides CI-IND if the advantage of A in the CI-IND game, Adv(A) =|Pr[b=b]−1/2|, is negligible. CI-IND Intuition. The above definition models the claim of an adversary of being able to distinguish the client executing a Check-In protocol. For this, the challenger allows the adversary to request it to perform a number of Check-In operations on behalf of C0 and C1, two clients controlled by the challenger. The adversary also specifies the location where the check-in is to take place. Then, the challenger chooses privately one of the two clients and performs a Check-In on its behalf, at a venue chosen by the adversary. The adversary wins if it is able to guess the client that has performed the check-in, with probability significantly higher than 1/2. We note that the challenger verifies the feasibility of the check-ins: the fact that the adversary is not trying to win the game by making it impossible for a client to succeed in a check-in at a location. In a second, StatVerify-Indistinguishability game, the adversary (e.g., service provider) should be unable to distinguish between clients running StatVerify, even if the adversary is able to trace client Check-In executions.

StatVerify Indistinguishability (SV-IND): A generates public information pub A and sends it to C but keeps the private information privA secret. The game has two steps. In the first step, A generates k = 2s new bits c1,..,ck such that s of them are 0 and s of them are 1. A also generates k venue ids, V1,..,Vk, Vi ∈ V, i=1..k. A sends c1,..,ck and V1,..,Vk to C. For each i=1..k, C runs Check-In(Cci(Idci,Vi,T,pubA),A(privA)), only if the time between the previous Check-In of client Cci and Ti is sufficient to enable Cci to travel the distance between the venue of the previous Check-In and Vi. At the end of this step, C verifies that C0 and C1 have performed the same number of check-ins at any venue V1,..,Vk. If this verification does not succeed, C stops the game. In the second step, A sends to C a venue id V ∈ V, such that the distance between the venue of the last Check-In of client Cj (j=0,1) and V can be physically traversed from the time of that Check-In to the current time. C generates a bit b∈{ 0,1} and runs StatVerify(Cb(Idb,V,T,pubA),A(privA). A outputs a bit b. A solution is said to provide SV-IND if the advantage of A, Adv(A) =|Pr[b=b]−1/2|, is negligible. SV-IND Intuition. The SV-IND game models the inability of A, that controls the entire system with the exception of two clients C0 and C1, controlled by C, to guess the identity of the client (C0 or C1) performing a Stat-Verify operation. For this, in an initial step, A is allowed to request C to perform Check-In operations and specify the identity of the client and the venue where the check-in is to be per- formed. At the end of this step, C verifies that the two clients are equivalent: they have the same (badge) status at all the venues requested by A. A secretly chooses one of the clients and executes StatVerify on its behalf for one of the venues chosen by A. A wins if it is able to guess the identity of the client with probability significantly larger than 1/2. The following property models the ability of the server to collect venue-based statistics: Provider Usability. The service provider can count the Check-In executions for any venue as well as list the issued badges and mayor ships.

Client Side: The client is assumed to be malicious. Malicious clients can be outsiders that are able to corrupt existing devices or may be insiders, i.e., 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 [11]), 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. Our solutions are not designed to handle private venues, venues that uniquely identify the user performing a check- in there (e.g., the user's home). In the following game, k is a system parameter that denotes the number of check-ins a user needs to perform in order to acquire special status (a badge). Status Safety. The challenger C controls the service provider and the adversary A controls any number of clients. The challenger runs first the Setup protocol and provides A with its public parameters. A runs Subscribe any number of times to generate clients. A then runs Check-In with C for any number of venues, but at most k − 1 times for any venue. A runs StatVerify with C. The advantage of A is defined to be Adv(A) = Pr[StatVerify(C(paramsC),S(privS)) = 1]. We say that a solution is status safe if Adv(A) is negligible. Token Non-distributability. No client or coalition thereof can use the same set of tokens more than once. Token-Epoch Immutability. No client or coalition thereof can obtain more than one token per site per epoch.

***GEO-BADGE*:** Geo-Badge is a private protocol that allows users to prove having visited the same location k times (see Fig. 3 for a high level diagram). The set of supported k values is predefined, e.g., k=1 for "Newbie", k = 10 for "Adventurer", k = 25 for "Explorer", etc, and is known by all client applications. At the end of the section we show how to adapt this solution to support T-Badges. Geo-Badge 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 venue. When k shares have been acquired, the client is able to reconstruct the secret, which is the proof required for the badge. The single use nonces prevent users from distributing received shares (or proofs). Geo-Badge extends Geo and provides the skeleton on which we build the subsequent solutions. For instance, the anonymous authentication and location verification functions are only described for Geo-Badge and inherited by Freq-Badge and MP-Badge. Each client maintains a set Tk, storing all the tokens accumulated during Check-In runs. When the client accumulates enough tokens in Tk to achieve special status, it runs StatVerify, aggregating the tokens in Tk. In the following we instantiate each protocol, executed between a client C and the GSN provider S. Setup(S()): Executed once in the beginning, by S. S generates a large prime modulus p that will be used to compute secret shares and publishes p. S generates a random key K, that will be used for authentication purposes. K is kept secret

by S. For each badge that requires k check-ins, S generates two large primes pk and qk such that qk|(pk −1). LetGqk be the unique subgroup ofZ ∗ pk of order qk. Letgk be a genera- tor of Gqk. S generates a fresh, random geo-badge GBk and computes the commitment value CMTk = gGBk k ∈ Gqk. For each supported badge, S publishes pk, qk, gk and CMTk, but keeps secret GBk.

RegisterVenue(O(V),S(privS)): The owner O that registers venue V, sends toS its public key. For each new venue V, for which the service provider offers badges (after k Check-In runs) S generates a secret MV randomly. S uses a threshold secret sharing solution to compute shares of MV, by gener- ating a polynomial Pol of degree k−1 whose free coefficient is MV: Pol(x) =MV+c1x+c2x2+...+ck−1xk−1. S keeps Pol's coefficients secret but publishes the degree k and the verification value VerV =H(HMACK(V)MV mod p). A client that reconstructs VerV, has proof of having achieved the special status (Geo-Badge). S stores Pol's coefficients for V, along with the public key of V's owner. Subscribe(C(),S(pubS,privS)): The communication in this step is performed over Mix, to hide C's location from S. C runs the setup stage of the Anonymous Authentication protocol to obtain tokens that allow it later to authenticate anonymously with the server.

CheckIn(C(Id,V,T,pubS),S(privS)): Let (current) time T be during epoch e. The following actions are performed by a client C and the service provider S:

- Anonymous Authentication: C runs the anonymous authentication procedure to prove to S that it is a subscriber. This step is performed over Mix.
- Location Verification: C runs a location verification protocol to prove presence at V.
- Token Generation: C generates a fresh random value R and sends the blinded R to S, asObf(R) (obfus- cated for instance using a modular multiplication, see Chaum's work [36] on blind signatures). S computes xe = H(e) mod p and ye = Pol(xe) mod p. S sends to C the tuple (xe,ce,SigS(Obf(R))), where ce = HMACK(V)ye mod p and the last field denotes the blindly signed R. C "unblinds" the signed nonce (see [36]), obtains se =SigS(R) and stores (xe,ce,se) into the set Tk. StatVerify(C(Id,V,k,Tk,pubS),S(privS,k)): LetTk = {(x1,c1,SigS(R1)), .., (xk,ck,SigS(Rk))}. Letlj(x) = m=1..k,m=j x−xm xj−xm mod p be the Lagrange coefficients.

The following steps are executed, over Mix:
• C computes SS = ——————————————j=1..kcjlj(0). C verifies that H(SS) = VerV. If the verification fails, C outputs -1 and stops. Otherwise, it sends SS, along with the set of signed nonces, (SigS(R1),..,SigS(Rk)) and the venue V to S.
• S verifies that (i) the k random values are indeed signed by it, (ii) that R1,..,Rk are unique and have not been used before and (iii) that H(SS) =VerV. If either verification fails, S outputs -1. Otherwise, S stores the values R1,..,Rk, then sends the badge GBk (see Setup) toC (over Mix).
Prove Badge(C1(pubS,GBk,V,pk,qk,gk), C2(pubS,V,pk,qk,gk), S(privS,V,CMTk): C2 retrieves CMTk from S. C1 and C2 engage in a zero knowledge protocol where C1 proves knowledge of the GBk, the discrete logarithm of CMTk, for instance, using Schnorr's solution .

***TOURING BADGE (T-BADGE)*:** The "adventurer" badge is unlocked when the user checks- in at k different locations. Geo-Badge can be easily modified to support this functionality: the provider assigns one share (one point of the polynomial Pol) to each participating venue. The free coefficient of Pol is the secret which unlocks the badge. Whenever a user checks-in at one venue, it receives the share associated with the venue. After visiting k venues, the user has k shares and can reconstruct the secret and unlock the badge. Note that multiple check-INS at the same venue will retrieve the same share, thus forcing the client to visit k different venues. We note that multiple users could collude and combine their shares to obtain an "adventurer" badge, while none of them in isolation satisfies the condition. However, users may lack incentives for this attack: only one of the participants would receive the badge while the others waste their shares.

FREQ-BADGE: Using the Foursquare terminology, the user that has run Check-In the most number of times, at a venue SV, with in the past m epochs, becomes the mayor of the place. Let MrV denote the number of check-ins (at V) performed by the current mayor of V. We introduce Freq-Badge = {Setup, Register Venue, Maintain Venue, Maintain Badge, Subscribe, Check-In, StatVerify, ProveBadge}, a solution that extends Geo with two protocols: Maintain Venue and Maintain Badge. FreqBadge allows clients to prove having performed any number of check-ins, not just a predefined value. The check-ins are time constrained: clients have to prove that all check-ins have occurred in the past m epochs. Furthermore, client issued proofs can be published by the provider to be verified by any third party, without the risk of being copied and re-used by other clients.

Overview Freq-Badge achieves these features in the following way: In the Maintain Venue protocol, the service provider generates exactly one fresh token per epoch, for each supported venue V. When a client runs Check-In at V, it receives V's token for the current epoch. The client stores the tokens accumulated for V in the set TkV. At any time, for any venue V, the provider publishes and makes available upon request to any client, two values, (i) MrV, the number of tokens that the mayor of V has proved to have accumulated in the past m epochs, and (ii) CMTV, a badge commitment value whose true nature we will reveal later. If, during a Check-In run, a client's number of tokens, |TkV|, exceeds the current MrV, StatVerify is invoked. The provider maintains a queue of StatVerify requests: each new request is placed at the end of the queue and each request is processed in the order in which it was received. StatVerify succeeds only if the client is able to prove to the provider that it knows at least MrV +1 out of them tokens given in the past m epochs for that venue. The proof is in zero knowledge. If the proof succeeds, it is

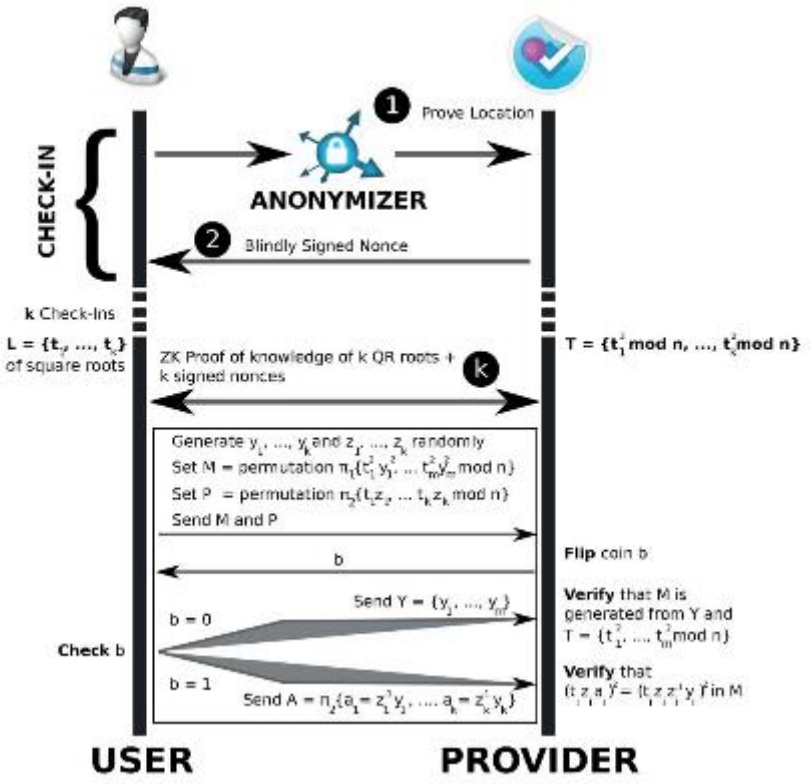


Fig.4. Diagram of Freq-Badge

published by the provider, along with an increased MrV value, reflecting the new mayor's number of tokens. The provider then issues a private Freq-Badge badge to the client, and publishes CMTV, a commitment value for this badge. If multiple clients initiate the Stat Verify protocol simultaneously, with the same number of tokens, only the first becomes the mayor: after the completion of the first client's Stat Verify protocol, the MrV value is incremented. The second client's StatVerify will not succeed, since its number of tokens does not exceed (but only equals) the new MrV value.

The Solution We now describe each protocol of Freq-Badge,The server generates two large safe primes p and q and the composite n = pq. LetN denote n's bit length. S publishes n and keeps p and q secret. RegisterVenue(O(V),S(privS,MrV)): For a newly registered venue V, S generates a new random seed rV and uses it to initialize a pseudo-random number generator GV. S also generates two large primes pV and qV such that qV|(pV−1). Let GqV be the unique subgroup of Z ∗ pV of order qV. LetgV be a generator of GqV. S publishes pV, qV and gV. S also sets MrV to 0: the venue has no mayor yet.

V. EVALUATION

We have implemented Geo-Badge, Freq-Badge and MP Badge in Android and Java and have tested the client side on the Nexus One smart phone and the server side on a 16 quad core server featuring Intel(R) Xeon(R) CPU X7350 @ 2.93GHz and 128GB RAM. We have stress-tested the server side by sequentially sending multiple client requests. All the results shown in the following are computed as an average over at least 10 independent runs.

Geo-Badge: We study the most compute-intensive functions of Geo-Badge: Setup, the GSN provider side of Check-In, the client and provider sides of StatVerify. We investigate first the dependence on the modulus bit size. The Setup cost, a one time cost for the GSN provider, ranges from 277ms for 512 bit keys to 16.49s for 2048 bit keys. Fig. 6(a) shows the performance of the remaining three components in milliseconds (ms) using a logarithmic y scale. The x axis is the modulus size, ranging from 512 to 2048 bits. The value of k, the number of Check-In runs required to acquire the badge is set to 50. On a single core, the CheckIn cost is 13ms even for a 2048 bit modulus size. The cost of the provider side of StatVerify is almost constant for different key bit sizes, also around 13ms – on an Open SSL sample, the cost of performing one signature.

FreqBadge: In the next experiment we studied FreqBadge. We have first tested key bit sizes ranging from 512 to 2048. A one time occurrence for the GSN provider, the Setup cost ranges from 227ms to 1.5s and is negligible. Fig. 7(a) shows the performance of Check-In (server side) and StatVerify (client and server side) in ms, as a function of the key bit size. The y axis shows the time in ms, in logarithmic scale. s, the number of proof rounds is set to 40, m, the number of past epochs is set to 60 and k, the number of CheckIn runs is set to 30. The client side StatVerify, executed on the Nexus One platform , requires between 1.7s to 7.5s. Since the provider is the bottleneck, the sensitive operations are Check-In and the provider side of StatVerify. These operations are fast: Requiring one table lookup and a signature generation, Check-In takes 4.8ms. On a 16 quadcore server, the provider can support more than 13,000 check-ins per second more than 1.1 billion ops per day. The provider side of StatVerify is less compute intensive than the client side: it ranges from 36ms to 309ms (for 2048 bit keys).

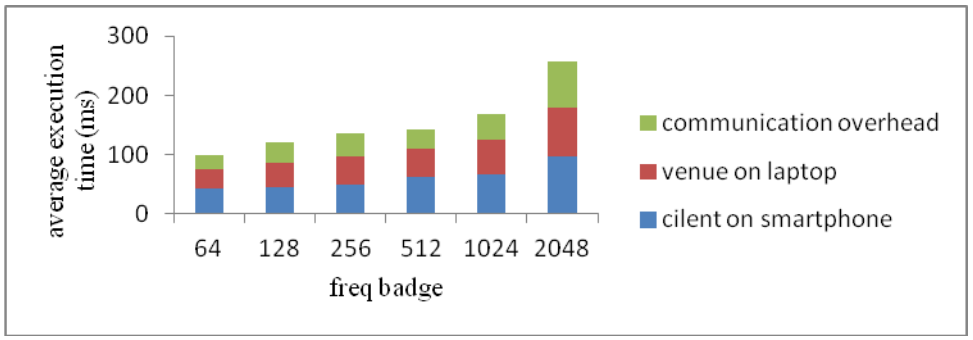


Fig.5. Average execution of freq-badge

.

VI. CONCLUSIONS

In this paper we have studied privacy issues concerning popular geosocial network features, check-ins and badges. We have proposed several private protocols including (i) Geo-Badge and T-Badge, for acquiring location badges, (ii) Freq-Badge, for mayor ship badges and . Furthermore, we have devised e-Badge, a novel protocol that allows users to privately build expertise badges. We showed that Geo-Badge, Freq-Badge and MP Badge are efficient. The provider can support thousands of Check-Ins and hundreds of Stat Verifies per second. A Smartphone can build badges in a few seconds.

REFERENCES

[1] Foursquare. https://foursquare.com/.
[2] Yelp. http://www.yelp.com.
[3] Lauren Indvik .Foursquare Surpasses 3 Million User Registrations. http: //mashable.com/2010/08/29/foursquare-3-million- users/.
[4] Jolie O'Dell. Foursquare Day Sets Record with 3M+ Checkins. http: //mashable.com/2011/04/20/foursquare-day-2/.
[5] ChloeAlbanesius.Applelocation,privacyissuepromptshouseinquiry. PCMag.http://www.pcmag.com/article2/0,2817,2365619,00 .asp.
[6] Jennifer Valentino-Devries. Google defends way it gets phone data. Wall Street Journal. http://online.wsj.com/article/ SB10001424052748703387904576279451001593760.html, 2011.
[7] Balachander Krishnamurthy and Craig E. Wills. On the leakage of personally identifiable information via online social networks. In WOSN, pages 7–12, 2009.
[8] 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.
[9] Please rob me. raising awareness about over-sharing. www.pleaserobme. com.
[10] Josh Lowensohn. Apple sued over location tracking in iOS. Cnet News. http://news.cnet.com/8301-27076 3-20057245-248.html/, 2011.
[11] Gpscheat! http://www.gpscheat.com/.
[12] John Krumm. Inference attacks on location tracks. In Pervasive, 2007.
[13] Philippe Golle and Kurt Partridge. On the anonymity of home/work location pairs. In Pervasive, 2009.
[14] John R. Douceur. The Sybil Attack. In IPTPS, pages 251–260, 2002.
[15] Bogdan Carbunar, Radu Sion, Rahul Potharaju, and Moussa Ehsan. The shy mayor: Private badges in geosocial networks. In Proceedings of the 10th International Conference Applied Cryptography and Network Security (ACNS), pages 436–454, 2012.
[16] Marco Gruteser and Dirk Grunwald. Anonymous usage of location- based services through spatial and temporal cloaking. In Proceedings of MobiSys, 2003.
[17] 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.
[18] Femi G. Olumofin, Piotr K. Tysowski, Ian Goldberg, and Urs Hengart- ner. Achieving Efficient Query Privacy for Location Based Services. In Privacy Enhancing Technologies, pages 93–110, 2010.
[19] Xiao Pan, Xiaofeng Meng, and Jianliang Xu. Distortion-based anonymity for continuous queries in location-based mobile services. In GIS, pages 256–265, 2009.
[20] 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.
[21] David Eppstein, Michael T. Goodrich, and Roberto Tamassia. Privacy- preserving data-oblivious geometric algorithms for geographic data. In GIS, pages 13–22, 2010.