# Privacy-preserving Social Recommendations in Geosocial Networks

Bisheng Liu and Urs Hengartner
Cheriton School of Computer Science
University of Waterloo
{bisheng.liu, urs.hengartner}@ uwaterloo.ca

***Abstract*—Geosocial networks like Foursquare have enabled people to conveniently share their whereabouts with their friends online, such as sharing check-ins at visited venues. This information could be utilized by recommender systems to improve the recommendation accuracy, known as *social recommendations*. However, incorporating social context into recommender systems introduces new privacy threats to users. We design a framework to achieve the benefits of social recommendations while *preserving the privacy of social relations* and *considering the business interests of the service provider (SP)*. Namely, we propose that each user manages social relations locally and participates in computing social recommendations without revealing social relations to the SP and without the SP revealing proprietary information to a user. In addition, we identify two classes of inference attacks where the SP may infer the existence of social relations by a) observing the arrival times and the originating addresses of messages and b) monitoring users' individual check-in histories. Furthermore, we propose using *private check-ins* and *delayed check-ins* to defend against such attacks. Finally, we conduct a comprehensive performance evaluation over large-scale real-world datasets. The results suggest that the proposed privacy-preserving framework is feasible on a smart phone and only slightly affects the overall performance of recommender systems.**

## I. INTRODUCTION

Traditional recommender systems [1] utilize users' previous behaviors, as well as the information gathered from item profiles and user profiles, to compute recommendations. Recently, researchers have discovered that recommender systems can exploit users' *social context* to improve the recommendation accuracy, known as *social recommendations* [21–23, 31]. For example, when it comes to the choice of restaurants, a user would probably prefer recommendations from friends rather than recommendations from strangers. Furthermore, the recent explosive growth of social networking services has enabled people to conveniently share information with friends online. For example, *geosocial networks* allow users to check in at visited venues at any given time and share the check-in information with their friends. Such check-in information could be conveniently utilized to provide point-of-interest (POI) recommendations for users. For instance, Foursquare [32], one of the most popular geosocial network service providers, now provides users with personalized POI recommendations based on where their friends have been to.

Because recommender systems may utilize sensitive information from users to produce recommendations, users might be unwilling to adopt such systems due to concerns over privacy. Privacy for traditional recommender systems has been well researched in the literature [2, 5, 7, 8, 17, 20, 25]. Most of the proposed privacy-preserving approaches aim to hide individual user records from the recommender system while still being able to provide users with accurate recommendations.

The focus of this paper is the new privacy threats introduced by social recommendations. We consider *the privacy of social relations* to be one of the most fundamental privacy requirements for a user. A recent study [14] shows that many people have an interest in hiding their friends set (approximately 52.6% of 1.4 million NYC Facebook users in the study hid their friends set in June 2011). For instance, certain sensitive friendships might uncover some hidden information of a user (e.g., friendships that might expose a user's sexual orientation [19]). In particular, privacy of social relations is critical in regions where geosocial network services are operated by unscrupulous service providers or closely monitored by suppressive governments [28]. For the abovementioned reasons, we assume that a user would prefer to keep her social relations secret both from strangers and from the service provider (SP). While designing a privacy-preserving recommender system, we should also keep in mind that a SP might only be willing to adopt or switch to the new system if adopting the new system would not be detrimental to the SP's business interests. For example, the SP would probably prefer to keep certain key elements of the recommender system hidden from other parties, such as the ratings of each user at different venues. In this paper, we address the following question: in a geosocial network, can we design a privacy-preserving recommender system that provides users with the benefits of social recommendations, while *a) keeping users' social relations private from the SP* and *b) preserving the SP's business interests*?

The contributions of this paper are as follows: *a)* our work is the first, to our best knowledge, to propose a privacy-preserving framework for social recommendations in geosocial networks; *b)* we take the SP's interests into consideration and propose that the SP encrypts each user's ratings at visited venues with cryptosystems that support homomorphic properties; *c)* we identify two types of *inference attacks* and propose using *private check-ins* and *delayed check-ins* to defend against these attacks; *d)* we evaluate the feasibility of our approach using large-scale datasets collected from a real-world geosocial network.

The remainder of this paper is organized as follows: In Section II, we present the related work. In Section III, we introduce some background material. In Section IV, we present in detail the proposed privacy-preserving framework. In Section V, we describe inference attacks and the defense mechanisms. We present the experimental results in Section VI. Finally we conclude our work in Section VII.

## II. Related Work

Recommender systems utilizing the information shared in social networks have recently attracted tremendous attention from both academia [21–23, 31] and industry [32]. Konstas et al. [21] have proposed a recommendation algorithm that performs a Random Walk with Restarts on a graph established among users, items and tags. Ma et al. [22, 23] have incorporated users' social friendships into model-based collaborative recommender systems. All these approaches aim to recommend traditional items, such as movies; the effectiveness of these approaches in geosocial networks has not yet been evaluated. Ye et al. [31] have proposed a unified POI recommendation framework in geosocial networks, which fuses user preferences for a POI with social influence and geographical influence. Our work adopts a similar unified POI collaborative recommender system.

Some of the earliest privacy-preserving approaches for recommender systems come from Canny [7, 8], who has proposed using homomorphic encryption and a peer-to-peer protocol to provide privacy for several model-based collaborative recommender systems. Similar ideas have been explored [17, 25]. Polat and Du [27] have proposed that customers use a randomized perturbation technique to disturb their private data before sending the data to the SP. Similar ideas have also been explored [5, 20]. Aïmeur et al. [2] have proposed using a semi-trusted third party to distill encoded sensitive customer information. Previous work aims to keep individual user records hidden from the SP or other users. Our research differs from these approaches because our research aims to keep users' social relations secret from the SP, while still providing users with the benefits of social recommendations.

Machanavajjhala et al. [24] have studied social recommendations solely based on a social graph. Their results suggest accurate recommendations inevitably leak information about the existence of edges between certain nodes. Our work differs from theirs because *a)* we are trying to hide the social relations from the SP and *b)* social recommendations in our framework are integrated with other recommender systems.

The closest references to our research can be found in the literature of privacy for social relations in social networks. Anderson et al. [3] have proposed an architecture for social networking that protects users' social information from both the operator and other network users. Domingo-Ferrer et al. [15] have proposed a scheme to prevent the resource owner in social networks from learning the relationships and trust levels between the users who collaborate in the resource access. Tootoonchian et al. [30] have proposed a cryptographic framework, *Lockr*, to separate social networking content from all other online social network functions. Backes et al. [4] have proposed a similar approach to achieve access control while keeping social relations hidden from the SP. We exploit a similar idea as *relation authentication* [4] to implement access control.

Eagle et al. [16] have explored the possibility of inferring friendships between users by analyzing high resolution user traces collected from mobile phones. Crandall et al. [10] have used datasets collected from the Flickr website to investigate the extent to which social relations between people can be inferred from co-occurrences in time and space. Cranshaw et al. [11] have proposed a novel set of location-based features and adopted a supervised machine learning framework to predict the friendship between two users by analyzing their location trails. A similar idea has been explored by Scellato et al. [29]. In our research, the SP does not have the ground truth information about friendships. Therefore, the SP is unable to exploit the supervised machine learning method to detect friendships.

## III. Background

### A. Social Recommender Systems

Most proposed social recommender systems include two components: *social influences*, which define how much influence each friend has on a user; *recommendation algorithms*, which define how to compute recommendations with friends' social influences and their ratings of each item/POI as inputs.

The definition of social influence varies in different systems. In this paper, we adopt the definition proposed by Ye et al. [31], which is based on the fraction of common friends and the fraction of common venues between a user and each friend. Let $SI_{i,k}$ be the social influence of user $k$ on user $i$, where user $k$ is a friend of user $i$. Let $F_i$ and $L_i$ denote the friends set and the visited venue set of user $i$, respectively. Let $F_k$ and $L_k$ denote the friends set and the visited venue set of user $k$, respectively. The social influence of user $k$ on user $i$ is defined as follows [31]:

$$SI_{i,k} = \eta \cdot \frac{|F_i \cap F_k|}{|F_i \cup F_k|} + (1-\eta) \cdot \frac{|L_i \cap L_k|}{|L_i \cup L_k|} \quad (1)$$

where $\eta$ denotes a tuning parameter with the value between 0 and 1.

The social context is typically incorporated into other recommender systems. Ye et al. [31] propose a linear fusion framework to integrate the results from multiple recommender systems, including social recommendations. We adopt their approach as the basis for our research, because their approach has been proved to be effective for POI recommendations in geosocial networks. Assuming that user $i$ has never visited venue $a$ before, the social recommendation algorithm by Ye et al. proposes estimating the rating of user $i$ at venue $a$ (defined as the *social rating* of user $i$ at venue $a$, $SR_{i,a}$) as follows:

$$SR_{i,a} = \frac{\sum_{k \in F_i} SI_{i,k} R_{k,a}}{\sum_{k \in F_i} SI_{i,k}} \quad (2)$$

where $F_i$ denotes the friends set of user $i$, $SI_{i,k}$ denotes the social influence of friend $k$ on user $i$ and $R_{k,a}$ denotes the rating of user

$k$ at venue $a$. If we merge the denominator of (2) into $SI_{i,k}$ in the numerator, then (2) can be rewritten as follows:

$$SR_{i,a} = \sum_{k \in F_i} SI'_{i,k} R_{k,a} \quad (3)$$

where $SI'_{i,k}$ in (3) represents the normalized social influence with the value between 0 and 1. Finally, for a given user $i$, her score at venue $a$ is defined as a normalized value of $SR_{i,a}$:

$$S^f_{i,a} = \frac{SR_{i,a}}{\max_a \{SR_{i,a}\}} \quad (4)$$

where $\max_a\{SR_{i,a}\}$ is a normalization term that represents the highest social rating for user $i$ at all venues.

The unified fusion framework [31], which integrates the results from multiple recommender systems, is represented as follows:

$$S_{i,a} = \sum_{k=1}^{n} \alpha_k S^k_{i,a}, \text{where} \sum_{k=1}^{n} \alpha_k = 1 \quad (5)$$

where $S_{i,a}$ denotes the final score of user $i$ at venue $a$, $S^k_{i,a}$ denotes the score of user $i$ at venue $a$ computed by recommender system $k$, and the weighting parameter $\alpha_k$ denotes the relative importance of recommender system $k$ compared to other recommender systems. The framework returns the top-$N$ highest ranked venues for each user as recommendations.

### B. Homomorphic Encryption

Homomorphic encryption is a form of encryption where performing a certain algebraic operation on the plaintext is equivalent to performing certain (possibly other) algebraic operations on the ciphertext. We describe the addition operation of an additive cryptosystem.

Let *Enc*() denote the encryption function. We can compute the encrypted value of the sum of two plaintext messages $m$ and $n$ as follows:

$$Enc(m+n) = Enc(m) \cdot Enc(n) \quad (6)$$

We can compute the encrypted value of the product $m$ and $k$ (where $k$ is a positive integer) as the modular exponentiation of the cipertext of $m$ with the plain text value of $k$ as the exponent, which can be denoted as follows:

$$Enc(m \cdot k) = (Enc(m))^k \quad (7)$$

As we will show in Section IV-D, in our framework, users need to compute (7) when $k$ is a floating point number. To perform real arithmetic over a finite field, we represent floating point numbers as fixed point numbers by multiplying them by a fixed base.

### C. System Model

A **legitimate user** in our model can create new venues in the system. **Venues** in our system can be either *private* (such as *Alice*'s home) or *public* (such as a local club). The verification of the identity of the user as the venue owner can be performed offline. We assume that the information of a **private venue** (e.g., GPS coordinates, venue name) is only available to the venue owner, the owner's friends and the SP.

**Friends** in our system are defined as users who have reached mutual consent to form a friendship; that is, the friendship in our system is bidirectional. When a user visits a venue, she can choose to check in at that venue by publishing the check-in information to the system. The check-in includes the identity of the user, the time when the check-in takes place, the venue information, and possibly some other data. Users may perform the check-in using a mobile device (e.g., a smart phone or a laptop). We assume that only a user's friends are allowed to access her check-ins. We also assume that the SP learns all checkin information (for reasons explained in Section III-D), but does not learn each user's friends set. A **stranger** to a given user is defined as a collection of non-registered visitors and registered users of the system who are not the user's friends.

**The service provider** (SP) in our system maintains a *user database* managing user identities and user profiles, a *venue database* managing venue profiles, a *check-in database* keeping all users' check-ins, and an *Access Control List* (ACL) for each user. The SP also manages a *user-venue rating matrix*. Each entry in the matrix represents the rating of a given user at a given venue, which is computed by the SP. Entries in the user-venue rating matrix are empty if the corresponding user has never checked in at the corresponding venue. For a given user, the SP aims at computing the scores at those venues with empty entries using (5) and ranking those venues accordingly. **Competitors of the SP** are defined as other geosocial network SPs who provide similar check-in services and POI recommendation services as the SP.

We overview **the recommendation process** in our system as follows: for each check-in, the SP is responsible for computing and updating the rating of the user at the venue using a certain *proprietary* algorithm with possibly the user's check-in frequencies, explicit feedbacks and/or some other observations over a long period as inputs; periodically, each user locally computes social recommendations *without* learning friends' ratings at venues and *without* revealing the social relations to the SP, and then submits the results to the SP (more details in Section IV); upon receiving a POI recommendation request, the SP runs multiple recommender systems and integrates the results including the social recommendations, and returns the top-$N$ highest ranked venues to the requesting user .

### D. Threat Model

In our threat model, we consider the friendship between users in a geosocial network to be sensitive, and thereby the friendship between two users should be kept private from the SP and strangers. We consider the SP to be honest but curious; that is, the SP follows our protocol, but is curious about learning the friends set of a user. We consider this assumption to be reasonable in Online Social Networks. The SP has a valuable reputation to maintain and any suspicion of malicious behaviors could potentially lead to a significant loss of users [12].

The business interests of a geosocial network SP usually lie in those collected check-ins. It would be very unlikely for the SP to adopt or switch to a privacy-preserving recommender

system if the new system could harm the SP's business interests. Therefore, in order to motivate the SP, we choose not to hide the contents of check-ins from the SP (except "private check-ins", which we will further describe in detail in Section V). In addition, we assume that the SP knows the real identity of each user in the system (e.g., Foursquare explicitly declares in the "Terms of Use" that users *must* provide Foursquare with accurate, truthful, and complete registration information). We do acknowledge that anonymity is an important privacy requirement for users on the Internet. However, location data gathered from check-ins could allow the SP to re-identify a user [6]. In our model, users can choose to use pseudonyms in order to keep anonymity from strangers. However, we do not consider user identities to be private from the SP.

In our model, we assume a user has the option of hiding the friends set from her friends. In case that a user decides to hide the friends set from her friend, we consider the friend to be honest (with that user) but also curious; that is, the friend follows our protocol when interacting with the user but is curious about learning the friends set of the user as well. Furthermore, we consider it is possible for the SP to infer the friendship between two users, if the SP discovers that the two users share a number of common friends.

We assume that users adopt a mix network (e.g., Mixmaster [26]) to send messages to the SP so that the SP cannot link multiple messages to the same sender. We also assume that the communication between users and the SP is secure so that other parties cannot learn anything by eavesdropping. In addition, as previously mentioned in the system model, the information of a private venue is only available to the SP, the venue owner and the owner's friends. Therefore, strangers to a legitimate user might be interested to learn such information.

Last but not least, from the SP's perspective, a competing SP might be interested in learning some key information of the system adopted by the SP, such as the ratings (computed by the *SP*) of users at each venue. A competing SP may try to learn such information either by colluding with users or by becoming a user himself.

## IV. Privacy-preserving Recommendations

In this section, we first give an overview of our framework. Then, we describe the components of the framework in detail.

### A. Overview

The core idea behind our framework is that we keep the sensitive social networking data hidden from the SP without affecting the performance of POI recommendations by the SP. We now briefly describe the components comprising the framework. We assume that *Bob* and *Alice* are both legitimate users in the system. Without loss of generality, we describe the framework from the perspective of *Bob*, who will be performing the following tasks:

**Managing friendship**: Upon receiving *Alice*'s friend request, *Bob* can choose to accept/reject the request. Once *Bob* and *Alice* have mutually agreed to form a friendship, they exchange the proof of the existence of the friendship, which we will discuss more in Section IV-B.

**Checking in at venues**: When visiting venue $a$, *Bob* can perform a check-in and publish the check-in to the system. Only *Bob*'s friends are allowed to access the check-in. We denote the check-in record as $check\_in(Bob, (a, L_a), t)$, where $L_a$ represents the GPS coordinates of venue $a$ and $t$ denotes the time when *Bob* performs the check-in.

**Accessing friends' check-ins**: If *Bob* is interested to learn his friend *Alice*'s whereabouts, *Bob* can anonymously request *Alice*'s recent check-ins from the SP. *Bob* proves the existence of the friendship with *Alice* to the SP by presenting the proof of the friendship without revealing his identity, which we will elaborate more in Section IV-C.

**Computing recommendations**: *Bob* participates in the computation of social recommendations for himself, by anonymously requesting his friends' ratings at venues from the SP, interacting with friends and computing the *social influence* (as defined in (1)) of each friend, calculating the social rating at each venue using (3) with the help of *homomorphic encryption*, and finally submitting the social recommendations to the SP. Finally, the SP integrates the results from multiple recommender systems using (5) and returns the recommendations to *Bob*.

### B. Managing Friendship

We assume that when a new user registers in the system, she generates a pair of public and private keys and sends a copy of the public key to the SP.

We now describe the process of forming the friendship between *Alice* and *Bob*. First, *Alice* and *Bob* reach mutual consent to form a friendship by exchanging request messages encrypted with each other's public key using an independent contact channel that does not involve the SP, such as a third party email account. Then, *Bob* sends *Alice* a secret friendship key $Fkey_{bob}$, which is shared only among *Bob* and his friends. Upon receiving $Fkey_{bob}$, *Alice* computes $H(Fkey_{bob})$ where $H$ is the SHA-256 hash function, and stores the result as the proof of the friendship with *Bob*, denoted as $FriendshipTag_{bob}$. *Bob* also computes $FriendshipTag_{bob}$ and sends it to the SP for future authentication. *Alice* performs the same procedure to set up the friendship with *Bob*. We will describe the usage of $Fkey_{bob}$ in Section V.

We now briefly describe the process of ending the friendship between *Alice* and *Bob*. *Bob* deletes *Alice* from his friends set. In addition, *Bob* chooses a new friendship key and broadcasts it to all remaining friends. Then, *Bob* computes the new corresponding $FriendshipTag_{bob}$ and sends it to the SP. Because *Alice* does not know the new friendship key, she is unable to compute the new $FriendshipTag_{bob}$. Again, *Alice* performs the same procedure.

### C. Accessing Friends' Check-ins

In our system, only a user's friends are allowed to access her check-ins. The SP maintains an *Access Control List* (ACL) for each user, which defines who is allowed to access the user's check-ins. We exploit a similar idea from Backes et al. [4] to implement a friendship-based ACL.

We briefly describe the authentication process. Upon setting up the friendship with *Bob*, *Alice* can prove to the SP that she is indeed a friend of *Bob* by presenting $FriendshipTag_{bob}$ to the SP. The SP accepts/rejects *Alice*'s claim by checking $FriendshipTag_{bob}$ with the copy received from *Bob*. Notice that, $FriendshipTag_{bob}$ does not contain the identity of *Alice* and therefore *Alice* can use the tag to anonymously request *Bob*'s check-ins from the SP. We assume a secure anonymous channel (such as a mix network [26]) where each user can anonymously communicate with the SP, so that outside attackers cannot launch impersonation, phishing, interception, or replay attacks.

This approach slightly differs from the approach proposed by Backes et al. [4], where the authors propose that a user signs the (non-secret) friendship tag before sending it to his/her friends and uses the signature as the proof of the friendship. A friend of the user uses *zero-knowledge proofs* to prove knowledge of the signature without providing the signature to the SP. They argue that the SP might store the signature of the friendship tag and later reuse it in other scenarios. This approach could be used to authenticate the friendship in our framework; however, in our system, we assume the SP to be honest but curious, which implies the SP will not impersonate the user with other SPs. Therefore, we argue that it is unnecessary for a user to sign the friendship tag or to hide the signature from the SP in our application scenario. In addition, *zero-knowledge proofs* introduce extra computation and communication overhead.

### D. Computing Recommendations

As previously described in the system model, the SP maintains a *user-venue rating matrix M*. The entry in the *i*-th row and the *a*-th column of *M* represents user *i*'s rating computed by the SP at venue *a*.

We describe how to compute social ratings for user *i* using (3) without revealing the social relations to the SP. We require that user *i* locally maintains a table $table\_friends_i$. Each row/record in $table\_friends_i$ corresponds to a friend of user *i*. Let user *k* be a friend of user *i*. The record in $table\_friends_i$ corresponding to user *k* consists of the public key of user *k,* the *social influence* of user *k* on user *i*, a list of venues that user *k* has previously visited and the rating of user *k* at each visited venue. Note that, the ratings of user *k* at visited venues are computed by the SP using a certain proprietary algorithm. We believe that the SP prefers to keep the ratings private from other parties, especially from competing service providers, because the performance of the whole recommender system highly depends on whether or not these ratings truly reflect users' preferences and interests.

For a given user *i*, we divide the computation of recommendations for the user into four steps.

#### 1) Step one: obtaining friends' ratings.

User *i* anonymously requests her friends' ratings from the SP using the same method as requesting the friends' check-ins, as described in Section IV-C. Notice that, if user *i* requests the ratings of multiple friends using a single message, the SP can infer that these users requested by user *i* share a common friend. As described in the threat model, users who share a number of common friends may very well be friends with each other. To prevent such correlations, we require that user *i* uses separate messages to request the ratings of each friend. As described in our threat model, we assume that the user adopts a mix network to send messages to the SP so that the SP cannot link two separate messages originating from the same user. Prior to sending the ratings to the requesting user, the SP encrypts the ratings using a homomorphic additive encryption scheme with his public key to protect the business interests. Upon receiving the encrypted ratings, the user updates the corresponding records for each friend in $table\_friends_i$.

#### 2) Step two: computing social influences.

We first describe how to compute the social influence of each friend using (1) if user *i* chooses to hide the friends set from her friends. Let user *k* be a friend of user *i*. In order to compute the cardinality of the intersection set between $F_i$ and $F_k$ ($F_i$ and $F_k$ denote the friends set of user *i* and user *k*, respectively) while keeping $F_i$ hidden from user *k*, user *i* initiates a *Private Set Intersection Cardinality* (PSI-CA) protocol [13], which can determine the cardinality of the intersection set without revealing the actual set. Upon the completion of the protocol, user *i* learns the value of $|F_k|$, $|F_i \cap F_k|$ and thereby the value of $|F_i \cup F_k|$, while user *k* only learns the value of $|F_i|$. On the other hand, if user *i* is comfortable with sharing the friends set with her friend *k*, user *i* can directly exchange the whole friends set with user *k* and then calculate the value of $|F_k|$, $|F_i \cap F_k|$ and $|F_i \cup F_k|$. Assuming a user's friend is honest with the user, user *i* can just send the computed $|F_i \cap F_k|$ and $|F_i \cup F_k|$ to user *k* so user *k* does not need to repeat the computation. In addition, user *i* can locally compute the intersection and the union of $L_i$ and $L_k$ ($L_i$ and $L_k$ denote the visited venue set of user *i* and user *k*, respectively), because user *i* has already stored the list of venues visited by user *k* after completing Step one. Finally, user *i* stores/updates the social influence of user *k* in $table\_friends_i$.

#### 3) Step three: computing and updating social ratings.

Upon completing Steps one and two, user *i* can compute the social ratings using (3). Even though the ratings of friends at venues are encrypted with the SP's public key, user *i* can compute the ciphertext of the social ratings encrypted with the SP's public key with the help of homomorphic encryption. User *i* calculates the ciphertext of the social rating at venue *j* using the following equation, which is derived from (3) using (6) and (7):

$$Enc(SR_{i,a}) = \prod_{k \in F_i} (Enc(R_{k,a}))^{SI'_{i,k}} \quad (8)$$

Because the normalized social influence of friend *k* on user *i* $SI'_{i,k}$ is a floating point number with the value between 0 and 1, we represent $SI_{i,k}$ as a fixed point number by multiplying it by a fixed base. User *i* sends the encrypted social ratings back to the SP. Finally, the SP decrypts the social ratings using his private key, calculates the scores using (4), and updates the records corresponding to user *i* for future usage. Note that, there is no motivation for user *i* to report falsified social ratings because it would only affect the social recommendation accuracy for herself.

*4) Step four: generating POI recommendations.*

Upon receiving the POI recommendation request, the SP runs multiple recommender systems simultaneously, integrates the results from each recommender system including the results from the social recommender system, and returns the top-$N$ highest ranked venues based on the final scores.

Note that, Steps one, two and three can be carried out by user $i$ periodically, regardless of when Step four actually takes place. We believe that the overall social recommendations from the friends are relatively stable over a period of time. Therefore, in order to reduce the computation and communication overhead, user $i$ can set the frequency of performing Steps one, two and three to be relatively low, e.g., once every week.

## V. Inference Attacks and Defense Mechanisms

Using our proposed framework, a user is able to anonymously access friends' check-ins, thereby preventing the SP from learning the existence of a friendship from a single request message. However, the SP might be able to infer the friendship by observing and correlating multiple messages or check-ins. In this section, we introduce two types of *inference attacks*: *check-in-inference attacks* where the SP infers the friendship by observing users' check-in histories; *message-inference attacks* where the SP infers the friendship by observing the arrival times and the originating addresses of messages sent by users. Furthermore, we propose several mechanisms to mitigate the threat of inference attacks.

### A. Check-in Inference Attacks

Assuming that *Alice* and *Bob* are friends with each other, we identify three types of *check-in-inference attacks* where the SP might be able to infer the friendship between *Alice* and *Bob* solely based on their check-in histories.

*a)* ***Private-venue related attacks.*** *Bob checks in at private venues owned by Alice, such as Alice's home. The private property of the venue implies that users who check in at the venue are most likely to be friends with the venue owner. Therefore, the SP could infer that Bob is a friend of Alice.*

To defend against such attacks, we first propose a special type of check-ins, *private check-ins*, where the user encrypts the venue information contained in the check-ins using her friendship key, which is shared only among the user and her friends (see Section IV-B). Therefore, only the friends of the user are able to decrypt the venue information. We denote the private check-in record of *Bob* at venue $a$ as $check\_in(Bob, E((a, L_a), Fkey_{bob}), t)$, where $E((a, L_a), Fkey_{bob})$ represents the encryption of the information of venue $a$ using an authenticated symmetric-key encryption function with *Bob*'s friendship key $Fkey_{bob}$. In addition, we require that $E()$ is a probabilistic encryption scheme so that the SP is unable to correlate multiple private check-ins that take place at the same venue.

Now we demonstrate how to use *private check-ins* to defend against the abovementioned attack. We propose that *Bob* always performs private check-ins at private venues. Let the check-in record be $check\_in(Bob, E(("Alice's\ home", L_{alice}), Fkey_{bob}), t)$, where $L_{alice}$ represents the GPS coordinates of *Alice*'s home. Consequently, the checked-in venue is hidden from the SP and therefore *Alice*'s identity is hidden from the SP; *Bob*'s friends can still decrypt the venue information because they know $Fkey_{bob}$. However, now a user who is a friend of *Bob* but is not a friend of *Alice* can learn the GPS coordinates of *Alice*'s home, which violates the privacy requirement of the private venue that only the SP, the venue owner and her friends are able to learn the venue information (as described in Section III-C). We propose that *Bob* first encrypts the venue information using *Alice*'s friendship key $Fkey_{alice}$ and then further encrypts it using his own friendship key $Fkey_{bob}$. The final check-in record is $check\_in(Bob, E(E(("Alice's\ home", L_{alice}), Fkey_{alice}), Fkey_{bob}), t)$. Now only a common friend of *Alice* and *Bob* can learn the information about *Alice*'s home and the fact that *Bob* checked in at *Alice*'s home at time $t$.

*b)* ***Attacks based on co-occurrences.*** *Alice and Bob hang out together a lot; therefore they checked in at quite a few venues together. By monitoring the number of co-occurrences between Alice and Bob's check-ins over a long period of time, the SP might be able to infer the friendship between them.*

We assume that the SP considers two users to be friends if the total number of co-occurrences between them exceeds a certain *detection threshold*. To mitigate the threat of such attacks, *Bob* can delay publishing the check-ins till *Alice* has left. Delayed check-ins may cause a problem for popular commercial applications such as real-time coupons for users in stores. Alternatively, we propose that prior to checking in, *Bob* first checks if *Alice* has already checked in at the same venue earlier by anonymously requesting *Alice*'s latest check-ins from the SP. If *Alice* has already checked in and the number of co-occurrences would exceed the detection threshold if *Bob* submits a normal check-in, *Bob* performs a private check-in. If *Bob* does not know the exact value of the detection threshold set by the SP, *Bob* can play conservatively by assuming the threshold to be a value that he is confident to be lower than the actual threshold chosen by the SP.

*c)* ***Attacks based on features of shared venues.*** *Alice and Bob share a number of comon venues (even if they did not visit those venues together). The SP might be able to infer the friendship by exploiting certain features of the shared venues.*

Similar techniques have been exploited to design a link prediction system for online location-based social networks [11, 29]. Scellato et al. [29] use a supervised machine learning method to predict new friendships based on place features, such as the fraction of common places between two users. In our framework, because the SP does not have the ground truth information about friendships, the SP can instead turn to a threshold based algorithm to detect the friendship using the same place features. Among those place features proposed by Scellato et al., the minimum location entropy across all shared venues and the similarity between users' check-in behaviors are reported to be two of the most important indicators of potential friendship. We briefly describe these two features for the convenience of future discussion.

Let $U$ denote the user set and $V$ denote the venue set. Let $C_a$ be the total number of check-ins of all users at venue $a$ and let $P_{i,a}$ be the check-in probability of user $i$ at venue $a$, which is

defined as $C_{i,a}/C_a$ where $C_{i,a}$ represents the number of check-ins of user $i$ at venue $a$. The location entropy of venue $a$, $H(a)$, is defined as follows [11, 29]:

$$H(a) = -\sum_{i \in U} P_{i,a} \log(P_{i,a}) \quad (9)$$

The results suggest that two users are more likely to become friends if they are observed together at locations of low entropy than at locations of high entropy such as a shopping mall [11, 29]. We require that the SP periodically computes and publishes the location entropy for each venue. We also assume that the SP would not lie about the value of the location entropy, since we assume the SP to be honest-but-curious.

In this paper, we adopt cosine similarity to measure the similarity between users' check-ins. The check-in similarity between users $i$ and $j$, denoted as $Sim_{i,j}$, is computed as follows:

$$Sim_{i,j} = \frac{\sum_{a \in V} C_{i,a} \cdot C_{j,a}}{\sqrt{\sum_{a \in V} C_{i,a}^2} \cdot \sqrt{\sum_{a \in V} C_{j,a}^2}} \quad (10)$$

To defend against such attacks, *Bob* constantly monitors those abovementioned features between *Alice* and him. If the value is approaching the detection threshold, *Bob* can perform private check-ins at venues which *Alice* has previously visited. Similarly, if *Bob* does not know the exact value of the threshold, *Bob* could conservatively assume the threshold to be a lower value.

**The impact of private check-ins.** Private check-ins can affect the performance of collaborative recommender systems, because the SP is unable to correlate the encrypted venue to any other venues stored in the system. However, considering the huge amount of check-ins stored in the SP's database, we believe that a relatively small number of private check-ins would not affect the overall performance of the system significantly. We evaluate in detail the impact in Section VI-C.

### B. Message Inference Attacks

Depending on the information leaked to the SP, messages sent by users can be grouped into three general classes: a) *identity-based* messages from which the SP can learn the identity of the sender of the message, e.g., check-in messages; b) *friendship-based* messages from which the SP can learn that the sender of the message is a friend of a certain user, e.g., messages requesting to access friends' check-ins; c) messages that do not reveal any information to the SP, e.g., messages requesting to access contents that are public to all users.

The SP can launch *message-inference attacks* by a) linking one identity-based message and one friendship-based message to the same sender, e.g., a check-in message sent by *Alice* earlier and a request message for accessing a user's check-ins originating from the same IP address as the first message (which indicates that *Alice* and the user being requested are friends with each other); b) linking two friendship-based messages to the same sender, e.g., two separate messages originating from the same IP address request accessing two users' check-ins (which indicates that the two users being requested share a common friend).

The SP can launch such attacks by observing and analyzing the arrival times and the originating addresses of the received messages. In order to mitigate such threats, we propose that users adopt a mix network (e.g., Mixmaster [26]) to send messages to the SP so that the SP cannot link multiple messages to the same sender.

## VI. Evaluation

In this section, we conduct the following experiments to evaluate the feasibility of our proposed framework: *a)* the cryptographic overhead on a smart phone; *b)* the threat of check-in-inference attacks; *c)* the impact of private check-ins.

### A. Datasets

We choose Gowalla as the data source for our experiments. Gowalla was a geosocial networking website where users shared their locations by checking in at venues (Gowalla was acquired by Facebook in December 2011). The friendship in Gowalla was bidirectional; that is, both parties needed to accept each other's friend request to form the friendship. The datasets used in our experiments were collected by Cho et al. [9]. We use three months of datasets between May 2010 and July 2010, which consist of the friendship network of Gowalla users and the time and location information of all check-ins made by users during the three month period. The whole datasets contain a total of 57,071 active users and 616,429 venues. The total number of check-ins is 1,818,714. We define the rating of a user at a given venue as the total amount of check-ins the user has performed at the venue. We obtain the user-venue rating matrix after summarizing the total check-ins, which has a density of $3.54\times10^{-5}$ (which is consistent with datasets crawled from other large-scale geosocial networks, e.g., $4.24\times10^{-5}$ for Foursquare datasets and $2.72\times10^{-4}$ for Whrrl datasets [31]). Each user has an average of 9.67 friends and each user has checked in at 21.84 venues on average.

### B. Cryptographic Overhead

In the framework, the most computation-intensive cryptographic operations are *Private Set Intersection Cardinality* to calculate social influences (only required if a user chooses to hide the friends set from the friends) and *homomorphic encryption* to compute social recommendations. We conduct the following experiments on a Google Nexus One phone featuring a 1-GHz Snapdragon CPU and 512 MB of RAM. For each experiment, we carry out 5,000 runs and average the results.

We adopt the cryptographic protocol proposed by De Cristofaro et al. [13] for Private Set Intersection Cardinality. The computation complexity is linear in the sizes of the two sets. Let *Alice* have $v$ friends and *Bob* have $w$ friends. Assuming that it is *Alice* who requests computing the set intersection cardinality from *Bob*, *Alice* needs to compute $2(v+1)$ exponentiations with short exponents and $v$ modular multiplications while *Bob* needs to compute $(v + w)$ modular exponentiations with short exponents and $w$ modular multiplications. Under the practical setting recommended in [13], the length of the exponent is 224

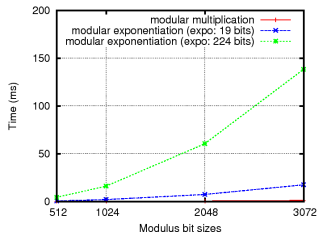



Figure 1. Varying modulus length

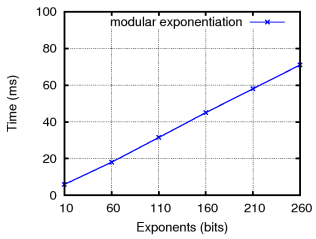



Figure 2. Varying exponent length

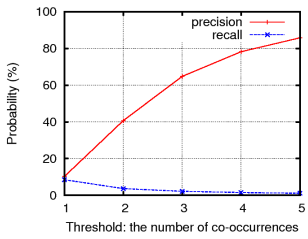



Figure 3. Co-occurrences

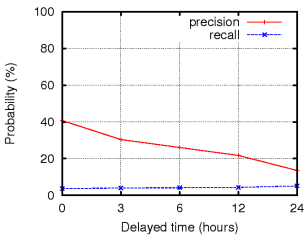



Figure 4. Delayed time $\Delta T$

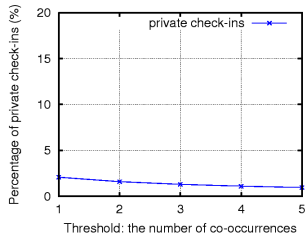



Figure 5. Private check-ins.

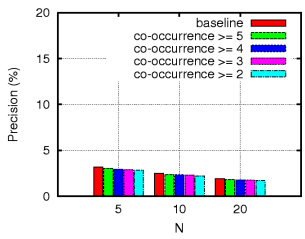



Figure 6. Neighborhood-based CF (precision)

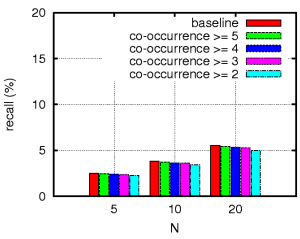



Figure 7. Neighborhood-based CF (recall).

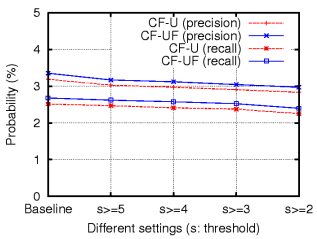



Figure 8. United CF system

bits. Furthermore, assuming a user's friend is honest with the user, *Alice* can just send the computed set intersection cardinality to *Bob* so *Bob* does not need to repeat the computation.

We adopt the Paillier cryptosystem as the homomorphic encryption scheme because of its additive homomorphic property. Assuming that the sum of the number of venues visited by a user's friends is $N_v$, the user needs to compute $N_v$ modular exponentiations and less than $N_v$ modular multiplications. In addition, in order to perform real arithmetic over a finite field, we represent floating point numbers as fixed point numbers by multiplying them by a fixed base, denoted as *base*. Therefore, the exponent used in (8) is the product of $SI'_{i,k}$ and *base*, denoted as *expo*. For evaluation purposes, we fix the value of $SI'_{i,k}$ to be 0.5 (recall that $SI'_{i,k}$ represents the normalized social influence with the value between 0 and 1) and fix the *base* to be $10^6$. Therefore, the length of *expo* is shorter than 19 bits.

In the first experiment, we vary the modulus length and evaluate the computation time of a single modular multiplication and a single modular exponentiation on the smart phone. We first set the length of *expo* to be 224 bits (in accordance with the case of computing social influences) and then run the same experiment with the length of *expo* as 19 bits (in accordance with the case of computing social recommendations). The results are presented in Fig. 1. In the second experiment, we fix the modulus length to be 2048 bits, vary the length of *expo*, and evaluate the computation time of a single modular exponentiation. The results are presented in Fig. 2.

Fig. 1 shows that on a smart phone, with a 2048-bit modulus, it takes approximately 0.54 ms to compute a modular multiplication, 7.54 ms to compute a modular exponentiation with the length of *expo* being 19 bits, and 60.63 ms to compute a modular exponentiation with the length of *expo* being 224 bits. Fig. 2 shows that the computation time of a modular exponentiation increases as the length of *expo* grows. For an average user in the tested datasets, it takes approximately 11.99 s to compute social influences of all friends (if the user wants to hide the friends set from her friends) and 1.71 s to compute social ratings at all venues visited by friends. Note that, users are only required to perform such cryptographic operations at low frequencies (e.g., only once a week), as mentioned in Section IV-D. Therefore, it is computationally feasible to perform the required cryptographic operations on a smart phone.

### C. *Check-in-inference Attacks Based on Co-occurrences*

**Threat of the attack.** We evaluate the threat of check-in-inference attacks based on co-occurrences, as described in Section V-b. The SP considers two users to be friends if the total number of co-occurrences between them exceeds a certain threshold. We compare the results with ground truth data. We use two metrics to study the threat of the attacks: *a) precision*: the fraction of identified friendships that are correct; *b) recall*: the fraction of real friendships that are correctly identified.

We consider two check-ins to be a *co-occurrence* if the venues contained in the check-ins are the same and the time difference between two check-ins is smaller than $T$. In this experiment, we choose $T$ to be 3 hours. We vary the threshold of co-occurrences from 1 to 5 and evaluate the precision and recall of the attacks. The results depicted in Fig. 3 indicate that as the threshold increases, the precision of the attack increases, while the recall decreases. The empirical results suggest two co-occurrences to be a practical choice of the threshold; the precision is over 40% and the recall is approximately 3.67%. While the recall is relatively low, the detection algorithm achieves relatively high precision, which suggests that if two friends have checked in at more than two venues together, the SP can infer the friendship between them at a rather high probability.

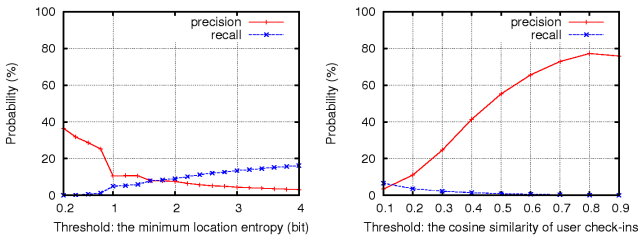
**Delayed check-ins.** If a user learns that his/her friend has just checked in at the same venue, the user can choose to delay performing the check-in by a period of $\Delta T$. If the friendship was detectable by the SP, to guarantee that the friendship is still detectable, the SP has to extend the observation window from $T$ to $T + \Delta T$ accordingly. We fix the threshold to be two co-occurrences, vary the value of $\Delta T$, and evaluate the performance of the attacks. The results are depicted in Fig. 4. As expected, the precision of the attack drops as we increase the value of $\Delta T$. However, the results indicate that the precision of the attacks remains over 13% even when $T$ is 24 hours, suggesting that delayed check-ins could only mitigate the threat, as the SP

is still able to detect the friendship at a relatively lower precision.

**Impact of the private check-in.** For each check-in in the datasets, we examine whether or not it needs to be encrypted to avoid exceeding the detection threshold for inference attacks based on co-occurrences. We calculate the fraction of private check-ins among all check-ins under different thresholds. Fig. 5 shows that there is only a small fraction of private check-ins among all check-ins. In particular, a total of 1.60% check-ins are required to be encrypted with the threshold being two.

In the following experiments, we investigate the impact of private check-ins on the performance of collaborative recommender systems. For simplicity's sake, we adopt a classic neighborhood-based collaborative recommender system [18, 31]. We adopt methods and metrics similar to Ye et al. [31] to evaluate the performance of the recommender system. For each individual user in the datasets, we randomly mark off $x$% ($x$ = 10, 30 (default value), 50) of all venues visited by the user. The aim of the recommender system is to recover the marked off user-venue pairs for each targeted user. The recommender system computes a ranking score for each venue and returns the top-$N$ ($N$ = 5 (default value), 10, 20) highest ranked venues for each user as recommendations. For a given user, if a recommended venue happens to be a previously marked off venue for the user, we consider the venue to be a recovered venue. We use two metrics to evaluate the performance of the recommender system: *a) precision*: the ratio of recovered venues to the $N$ recommended venues; *b) recall*: the ratio of recovered venues to the set of marked off venues.

We treat the performance of the recommender system when there are no private check-ins as our *baseline*. We compare the baseline with the performance of the recommender system when there are private check-ins under different thresholds of co-occurrences. For each experiment, we conduct 10 runs and average the results. We conduct the same experiments when $N$ = 5, 10, 20. The results are presented in Fig. 6 (precision) and Fig. 7 (recall). We omit plotting the error bar because the standard deviation is below one thousandth. The performance of the recommender system is consistent with the reported results in [31] (the reported precision in [31] is below 0.03 for the Foursquare datasets with $4.24\times10^{-5}$ density of user-venue rating matrix). Note that, the effectiveness of recommender systems for sparse datasets is typically rather low due to limited information contained in the datasets, which is expected to increase as more user check-ins are logged in the system. In our experiments, we focus on observing the relative performance difference of the recommender system with and without private check-ins instead of their absolute performance measures. Fig. 6 and Fig. 7 indicate that the introduction of private check-ins does not have a significant impact on the performance of the recommender system. There is an average of 0.35% loss in precision and 0.26% loss in recall when the detection threshold is two. In addition, as the number of total check-ins continues to grow, we expect the impact of private check-ins to be even less, because the remaining unencrypted check-ins are most likely enough to characterize user similarities. We also conduct the same experiments by varying the percentage of marked off venues, which exhibits similar results (figures omitted).




(a) Minimum location entropy (b) Check-in similarity

Figure 9. Attack based on features of shared venues.

In addition, we evaluate the performance of the unified collaborative framework using (5), which integrates the results of the neighborhood-based collaborative recommender system and the social recommender system (using the optimal setting recommended in [31]). We compare the performance of the unified collaborative framework (denoted as CF-UF) with the neighborhood-based collaborative recommender system (denoted as CF-U) under different thresholds of co-occurrences. Fig. 8 shows that incorporating social context indeed improves the performance and private check-ins do not affect the overall performance of the recommender system significantly.

### *D. Check-in-inference Attacks Based on Features of Shared Venues*

We evaluate the threat of check-in-inference attacks based on monitoring the minimum location entropy across all shared venues by two users and the similarity of user check-in behaviors. We use the same metrics, precision and recall, to evaluate the performance of the attacks.

In the first experiment, we assume that the SP considers two users to be friends if the minimum location entropy across their shared venues is smaller than a certain detection threshold. We then compare the results with ground truth data. Fig. 9(a) depicts the performance of the attacks as we increase the threshold of location entropy. The overall precision and recall are much lower than those from the attacks based on co-occurrences. Notice that, there is a steep drop in precision and increase in recall when the threshold of entropy is one bit. One possible explanation is that there are a total of 13.91% venues that have one bit of entropy and have been visited by only two users in the datasets.

In the second experiment, we assume that the SP considers two users to be friends if the cosine similarity of two users' check-in behaviors exceeds a certain threshold. Similarly, we vary the threshold of similarity weight and evaluate the performance of the attacks. We only consider users having at least five check-ins, because the similarity weights derived from users with too few check-ins may be misleading. The results are depicted in Fig. 9(b). When the threshold of similarity weight is 0.4, the achieved precision is approximately 41.46%, which is close to the performance of the detection algorithm based on co-occurrences when the threshold of co-occurrences is two. However, the corresponding recall is only 1.44%, which is approximately 40% of the recall achieved by the detection algorithm based on co-occurrences. Notice that, as we increase the threshold from 0.8 to 0.9, the precision of the algorithm

decreases, suggesting that most friends in the datasets have a check-in similarity weight less than 0.9.

The overall results suggest that inference attacks based on features of the shared venues between users pose a smaller threat than inference attacks based on co-occurrences. We treat the performance of the attacks based on co-occurrences when the threshold of co-occurrences is two as our *baseline*. In order to achieve the same level of precision as the baseline (which is approximately 40.71%), for the attacks based on minimum location entropy, we set the threshold of location entropy to be 0.2 bits and evaluate the fraction of private check-ins. A total of less than 0.4% (much lower than the fraction of private check-ins in the baseline, which is approximately 1.6%) of all check-ins needs to be encrypted, suggesting that the impact of the introduced private check-ins on the performance of the user-based collaborative recommender system is almost neglectable. Similarly, for the attacks based on users' check-in similarity, in order to achieve the same level of precision as the baseline, we set the threshold of similarity weight to be 0.4. Only 1.3% of all check-ins needs to be encrypted, suggesting that the introduced private check-ins only slightly affect the performance of the recommender system as well.

Overall, the empirical results in this section suggest that the proposed inference attacks indeed compromise users' friendship privacy. Users are most vulnerable against inference attacks based on co-occurrences. Delayed check-ins can mitigate the threat of inference attacks based on co-occurrences, but the attacker is still able to infer the friendship at a lower success probability. On the other hand, private check-ins are effective against all proposed inference attacks at the cost of slight reduction in recommendation accuracy.

## VII. CONCLUSIONS

In this paper, we propose a privacy-preserving framework to achieve social recommendations while keeping the social relations private from the SP. We propose that the SP encrypts the user ratings to protect his own business interests. We study two types of inference attacks, propose several mechanisms to defend against such attacks and conduct a thorough evaluation using large-scale datasets. The empirical results show the feasibility of our proposed framework.

## ACKNOWLEDGMENT

We thank the anonymous reviewers for their helpful comments. This work is supported by a Google Focused Research Award, the Ontario Research Fund, and the Natural Sciences and Engineering Research Council of Canada.

## REFERENCES

[1] Adomavicius, G. and Tuzhilin, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE TKDE*. 17, 6 (2005).

[2] Aïmeur, E., Brassard, G., Fernandez, J.M. and Onana, F.S.M. Alambic: a privacy-preserving recommender system for electronic commerce. *IJIS*. 7, 5 (2008).

[3] Anderson, J., Diaz, C., Bonneau, J. and Stajano, F. Privacy-enabling social networking over untrusted networks. *WOSN* 2009.

[4] Backes, M., Maffei, M. and Pecina, K. A security API for distributed social networks. *NDSS* 2011.

[5] Berkovsky, S., Eytani, Y., Kuflik, T. and Ricci, F. Enhancing privacy and preserving accuracy of a distributed collaborative filtering. *RECSYS* 2007.

[6] Bettini, C., Wang, X. and Jajodia, S. Protecting privacy against location-based personal identification. *Secure Data Management*. (2005).

[7] Canny, J. Collaborative filtering with privacy. *S & P* 2002.

[8] Canny, J. Collaborative filtering with privacy via factor analysis. *SIGIR* 2002.

[9] Cho, E., Myers, S.A. and Leskovec, J. Friendship and mobility: user movement in location-based social networks. *SIGKDD* 2011.

[10] Crandall, D.J., Backstrom, L., Cosley, D., Suri, S., Huttenlocher, D. and Kleinberg, J. Inferring social ties from geographic coincidences. *PNAS*. 107, 52 (Dec. 2010).

[11] Cranshaw, J., Toch, E., Hong, J., Kittur, A. and Sadeh, N. Bridging the gap between physical location and online social networks. *UBICOMP* 2010.

[12] De Cristofaro, E., Soriente, C., Tsudik, G. and Williams, A. Hummingbird: privacy at the time of twitter. *S & P* 2012.

[13] De Cristofaro, E. and Tsudik, G. *Fast and private computation of set intersection cardinality*. *CANS* 2012.

[14] Dey, R., Jelveh, Z. and Ross, K. Facebook users have become much more private: a large-scale study. *INFOCOM* 2012.

[15] Domingo-Ferrer, J., Viejo, A., Sebé, F. and González-Nicolás, Ú. Privacy homomorphisms for social networks with private relationships. *Computer Networks*. 52, 15 (2008).

[16] Eagle, N., Pentland, A. (Sandy) and Lazer, D. Inferring friendship network structure by using mobile phone data. *PNAS*. 106, 36 (2009).

[17] Han, S., Ng, W.K. and Yu, P.S. Privacy-preserving singular value decomposition. *ICDE* 2009.

[18] Herlocker, J.L., Konstan, J.A., Borchers, A. and Riedl, J. An algorithmic framework for performing collaborative filtering. *SIGIR* 1999.

[19] Jernigan, C. and Mistree, B.F.T. Gaydar: Facebook friendships expose sexual orientation. *First Monday*. 14, 10 (2009).

[20] Kaleli, C. and Polat, H. P2P collaborative filtering with privacy. *Turk J Elec Eng & Comp Sci*. 8, 1 (2010).

[21] Konstas, I., Stathopoulos, V. and Jose, J.M. On social networks and collaborative recommendation. *SIGIR* 2009.

[22] Ma, H., King, I. and Lyu, M.R. Learning to recommend with social trust ensemble. *SIGIR* 2009.

[23] Ma, H., Zhou, D., Liu, C., Lyu, M.R. and King, I. Recommender systems with social regularization. *WSDM* 2011.

[24] Machanavajjhala, A., Korolova, A. and Atish Das Sarma. Personalized social recommendations: accurate or private. *VLDB Endow.* 4, 7 (2011).

[25] Miller, B.N., Konstan, J.A. and Riedl, J. PocketLens: toward a personal recommender system. *ACM Trans. Inf. Syst.* 22, 3 (2004), 437–476.

[26] Möller, U., Cottrell, L., Palfrader, P. and Sassaman, L. 2003. Mixmaster protocol—version 2. *Draft* (2003).

[27] Polat, H. and Du, W. Privacy-preserving collaborative filtering. *IJEC*. 9, 4 (2003).

[28] Reeves, S. Internet is double-edged sword in Arab revolts. http://middle-east-online.com/english/?id=46109

[29] Scellato, S., Noulas, A. and Mascolo, C. Exploiting place features in link prediction on location-based social networks. *SIGKDD* 2011.

[30] Tootoonchian, A., Saroiu, S., Ganjali, Y. and Wolman, A. Lockr: better privacy for social networks. *CONEXT* 2009.

[31] Ye, M., Yin, P., Lee, W.-C. and Lee, D.-L. Exploiting geographical influence for collaborative point-of-interest recommendation. *SIGIR* 2011.

[32] Foursquare Explore. http://www.foursquare.com/explore