A robust reputation-based location-privacy recommender system using opportunistic networks

Yuchen Zhao
University of St Andrews
yz39@st-andrews.ac.uk

Juan Ye
University of St Andrews
jy31@st-andrews.ac.uk

Tristan Henderson
University of St Andrews
tnhh@st-andrews.ac.uk

ABSTRACT

Location-sharing services have grown in use commensurately with the increasing popularity of smart phones. As location data can be sensitive, it is important to preserve people's privacy while using such services, and so location-privacy recommender systems have been proposed to help people configure their privacy settings. These recommenders collect and store people's data in a centralised system, but these themselves can introduce new privacy threats and concerns.

In this paper, we propose a decentralised location-privacy recommender system based on opportunistic networks. We evaluate our system using real-world location-privacy traces, and introduce a reputation scheme based on encounter frequencies to mitigate the potential effects of shilling attacks by malicious users. Experimental results show that, after receiving adequate data, our decentralised recommender system's performance is close to the performance of traditional centralised recommender systems (3% difference in accuracy and 1% difference in leaks). Meanwhile, our reputation scheme significantly mitigates the effect of malicious users' input (from 55% to 8% success) and makes it increasingly expensive to conduct such attacks.

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Security

Keywords

location-based services, privacy, recommender systems, opportunistic networks, security, shilling attack

1. INTRODUCTION

Mobile devices such as smart phones have become more and more widely used in our daily lives. These devices are often embedded with Global Positioning System (GPS) sensors that allow people to use their location information for personalised online services. For instance, people can share locations with their friends on social media platforms such as Facebook[1] or record their lives on Foursquare.[2] Mobile applications such as Glympse[3] allow people to share their real-time locations with others to help them schedule meetings. On the one hand, such location-sharing services (LSSs) provide us conveniences, keep us in touch with our friends, and bring fun to our social lives. On the other hand, the increasing amount of location exposure created by these services introduces risks for people's privacy.

Existing user studies have shown that location is the most sensitive and valued personal information among people's mobile data [34]. Some locations visited, such as clinics and religious sites, may be sensitive and people may not wish to share these [4]. In addition, over-exposed location-sharing may lead to the risks of being stalked [15], which leads to concerns about location privacy in LSSs [36]. At the same time, incentives from businesses such as "badges" or simple monetary payments have been shown to influence users into over-sharing [18, 37], as do less transparent mechanisms such as exploiting targeted advertising [12]. The creation of appropriate location-sharing policies is therefore necessary to protect people's location privacy. But location-privacy preferences are dynamic based on context [2, 11] (e.g., time, location category, recipient), which means that fine-grained location-sharing policies may be needed for people to control the disclosure of their locations. Moreover, people find it difficult to manually configure these location-privacy policies [32], and so mechanisms are needed to make location-privacy tools more usable.

To relieve people from the burden of location-privacy configuration, many researchers have proposed the use of machine-learning techniques to recommend location-privacy preferences from individual users' data [6] or crowdsourced data [35, 40, 41]. These are all based, however, on the structure of a centralised recommender system to which people contribute their personal data. User studies have shown that people have privacy concerns about providing their data to such a centralised location-privacy recommender system [42]. These concerns have negative effects on their satisfaction about their choices, their perceived recommendation quality, and their acceptance of the recommendations. To deploy location-privacy recommender systems, they must be acceptable by people. It is therefore necessary to alleviate such concerns.

One possible solution is to let people exchange data with each other by themselves (i.e., without a central server) and generate recommendations on their devices locally. Unlike traditional recommender systems such as music or movie recommenders, location-

[1]http://www.facebook.com/
[2]http://foursquare.com/
[3]http://www.glympse.com/

.

privacy recommenders are used in mobile computing scenarios that comprise many mobile devices. These devices tend to be portable and embedded with short-range communication interfaces such as 802.11 or Bluetooth, which means that they can physically meet and exchange data with each other, thus creating an *opportunistic network* [29]. In this paper, we propose a decentralised location-privacy recommender system using opportunistic networks. This decentralised structure allows devices to exchange data with each other through short-range communication interfaces and generate recommendations locally. We compare the performance of this opportunistic recommender system with a traditional centralised recommender system and show that, without a central server, it is still possible to keep recommendation accuracy close to that of its centralised counterpart. Decentralised systems introduce new threats, however, such as malicious nodes who may wish to bias the recommendations. To alleviate the effects of such attacks, we explore the use of a reputation scheme based on node encounter frequencies.

The contributions of this paper are:

1. The design of a decentralised location-privacy recommender without centralised servers collecting users' data;

2. Analysis of the effectiveness of attacks that maliciously change recommended location-privacy settings;

3. A reputation scheme based on people's encounter frequencies in opportunistic networks that can significantly alleviate the effect of sampling attacks.

This paper is organised as follows. In Section 2, we discuss the state-of-the-art in location privacy recommenders and attacks against recommenders. In Section 3, we present our proposed recommender and reputation system. Section 4 outlines the design of our experiments and metrics used, and Section 5 discusses the results. We discuss potential applications and limitations of our work in Section 6 and conclude in Section 7.

2. RELATED WORK

2.1 Location-privacy recommenders

People's location-privacy preferences are complicated. On the one hand, people may need fine-grained settings to capture their dynamic location-privacy preferences [3, 20]. On the other hand, manually configuring fine-grained settings is burdensome [23]. Existing research has shown that people's location-privacy preferences can benefit from recommendations and suggestions [1, 22, 26]. To help people with location-privacy configuration, researchers have proposed to use machine-learning techniques to recommend users' location privacy preferences, thereby setting location privacy policies automatically. These proposed methods can be categorised into recommenders that use individual's data and those that use crowdsourced data.

Sadeh et al. [32] use machine-learning classifiers based on random forests to recommend users' location-privacy preferences. Recommendations are based on individual users' data and results show that recommendation accuracy is higher than that of user-defined rules. Similarly, Bigwood et al. [6] compare the performance of different machine-learning classifiers when recommending location-privacy preferences and their experimental results show that a classifier based on rotation forest can achieve a accuracy of 86%, which is also better than users' predefined settings.

Instead of just individual users' data, crowdsourced data have also been used to recommend location-privacy preferences. Toch proposes a Super-Ego crowdsourcing framework [35] that recommends location-privacy preferences from place-based crowdsourcing and semantic-based crowdsourcing. Experimental results show that a combination of crowdsourcing and personal bias has the best accuracy. Researchers have also applied collaborative filtering (CF) to location-privacy recommenders. Xie et al. [40] propose a privacy recommender that combines both user-based CF and item-based CF, and show that recommendation accuracy outperforms other baseline schemes. Zhao et al. [41] also propose a location-privacy recommender based on user-based CF. Their experimental results show that the recommender has better performance than schemes based on individual user's data when the training data are insufficient.

While these proposed systems all appear to provide accurate recommendations, they are all based on centralised structures. To alleviate the negative effects of people's privacy concerns about providing their data to a centralised recommender [42], in contrast to the existing works, we investigate the feasibility of deploying a location-privacy recommender in a decentralised fashion. In addition, compared with other proposed decentralised recommender systems [13, 24, 33], we demonstrate the vulnerability of decentralised recommender to sampling attacks, as explained in the next subsection, and propose a reputation scheme that mitigates the effectiveness of such attacks.

2.2 Attacks against recommenders

In both centralised and decentralised CF-based recommenders [31], recommendations are based on users' input. Since everyone can contribute their ratings, a recommender is vulnerable to those malicious users who try to bias the recommendations. These malicious users, i.e., attackers, can create a number of fake profiles and use these to inject modified data into the recommender. Such attacks are known as *shilling attacks* [21].

Depending on the attackers' knowledge about the recommender systems that they are trying to attack, shilling attacks can be categorised as *low-knowledge* or *high-knowledge* [16]. Many low-knowledge attack detection methods have been proposed for centralised recommenders [5, 9, 10, 38]. High-knowledge attacks are not as widely studied, and one particular type of high-knowledge attack, the *sampling attack* [8], has rarely attracted attention because attackers are considered incapable of accessing samples of real users' data to conduct the attack. In a decentralised recommender, however, people forward data for each other, which means that attackers can easily use the real users' data that they receive to generate fake profiles. These profiles are difficult to be detected through traditional analysis based on similarity and statistical features [9, 10, 38] because they are very similar to real profiles. Combining trust measures in the recommender has been shown to alleviate the effect of shilling attacks [27]. The proposed trust model is independent of user similarities, which means it can detect sampling attack profiles even if they are highly similar with real users' profiles. The trust model, however, is posterior, which means it needs a period of time to accumulate the trust from the recommendation results, and during this period the attackers can still launch successful attacks.

Compared with existing work in shilling-attack detection, we investigate the effect of the sampling attack and propose a reputation scheme to filter out shill profiles. Unlike the trust model, our scheme does not need to accumulate reputations from recommendation results. Reputation systems have been widely used in opportunistic networks to detect users' misbehaviour such as being selfish [7] or tampering with stored data [30]. Quercia et al. [30] propose a decentralised reputation system using public-key encryption and a gossip protocol to prevent users from modifying their local ratings. The assumption of their work is that each attacker only has one unique profile. In this paper, we consider attackers with the stronger capability to create multiple fake profiles.

3. APPROACH

We now describe the design of our decentralised location-privacy recommender. We then demonstrate how one type of shilling attack, the *sampling attack*, can be applied against a decentralised recommender system. Finally we introduce our reputation scheme, based on node encounter frequencies, to alleviate the effect of the sampling attack.

3.1 Decentralised location-privacy preference recommenders

Our decentralised location-privacy recommender system is based on two assumptions. Firstly, in LSSs, people move around with their mobile devices and encounter each other. This means that our recommender system can use an opportunistic network composed of mobile devices where people exchange location-privacy preferences when they encounter each other, without the existence of a centralised server. Secondly, we observe that location-privacy recommendations are only requested when people arrive at a place and decide to publish a location check-in. Therefore, there may be enough time for one's device to receive adequate data on the way from one place to another place before a recommendation is requested. Figure 1 demonstrates how the recommender works.

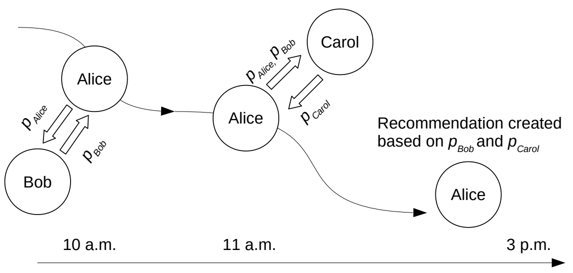

Figure 1: At 10 a.m., Alice's device and Bob's device encounter (move within communication range) each other and they exchange their stored location-privacy preferences. Then at 11 a.m., Alice's device encounters and exchanges data with Carol's device. When Alice arrives at her destination at 3 p.m. and wants to share her location, the system uses the data that she received from Bob and Carol to generate a location-privacy recommendation on her device locally.

3.1.1 *Preliminaries*

We use $U = \{u_1, u_2, ..., u_{N^U}\}$ to represent all the users that use the decentralised recommender system and N^U for the number of users. Each user has a profile p, which is represented by $p = (id, R, ts)$, where id is the profile identity, R is the user's location-privacy preference, and ts is the timestamp of the last update for this profile.

We assume that the LSS in question has a set of time slots T and a set of location categories L. The set of possible contexts can then be represented by $C = T \times L = \{c_1, c_2, ..., c_{N^C}\}$ and $N^C = N^T N^L$. A location-privacy preference is represented as $R_i = (r_{i,1}, r_{i,2}, ..., r_{i,N^C})$, where $r_{i,j}$ is the binary privacy setting (share or not) of a user u_i in a context c_j. In this paper, we only take location categories and time slots into account when constructing contexts because of the limitation of the data we used in our experiment. When more kinds of data (e.g. recipients) are available, the contexts can be extended by adding the new data as additional dimensions.

For a profile p, the initial value of ts is the time when p is generated. Each time a user updates their location-privacy preference in p; i.e., making a decision about whether to check-in in a certain context, ts is updated to the current time. Table 1 summarises the terms that we use to describe our system.

For users	
u	a user
N^U	the number of users
U	a user set, $U = \{u_1, u_2, ..., u_{N^U}\}$
T	a time slot set, $T = \{t_1, t_2, ..., t_{N^T}\}$
L	a location category set, $L = \{l_1, l_2, ..., l_{N^L}\}$
C	a context set, $C = T \times L = \{(t_1, l_1), (t_1, l_2), ..., (t_{N^T}, l_{N^L})\}$ $= \{c_1, c_2, ..., c_{N^C}\}$
N^C	the number of contexts, $N^C = \lvert C \rvert = N^T N^L$
R	a location-privacy preference, $R_i = (r_{i,1}, r_{i,2}, ..., r_{i,N^C})$, $r_{i,j}$ is u_i's location-privacy setting (share or not) in c_j
p	a profile of a user, $p = (id, R, ts)$, id is the profile identity, R is the user's location-privacy preferences, and ts is the profile's last update time
$P^{received}$	a set of received profiles
For attackers	
C^{target}	a target context set
int	the intent of attack (*push* or *nuke*)
s	a shill record, $s = (id^{affected}, id^{shill}, ts)$, $id^{affected}$ is the id of the affected profile, id^{shill} is the id of the shill profile made from the affected profile, and ts is the last update time of the affected profile
S	a shill record set, $S = \{s_1, s_2, ...\}$
P^{shill}	a shill profile set

Table 1: Terms and symbols used in our system

3.1.2 *Exchanging location-privacy preferences*

Each user u_i keeps a set of received profiles, $P_i^{received}$, from other encountered users. When two users u_i and u_j encounter each other, we consider two schemes of data exchange as follows:

- *Decentralised Individual Exchange (D-Ind)*: u_i and u_j only exchange their own profiles p_i and p_j.
- *Decentralised Set Exchange (D-Set)*: u_i and u_j will not only exchange their own profiles, but also all of the other profiles that they have received.

In *D-Ind*, after receiving p_j, u_i checks if p_j is already in $P_i^{received}$. If not, or if the received p_j is newer than the previously existing p_j (this can be done by comparing the timestamp of the received p_j and the timestamp of the existing p_j), then u_i adds, or updates, p_j in $P_i^{received}$. In *D-Set*, this check is done on every profile in the received set.

3.1.3 Local recommendations

Once a user arrives at their destination during a certain time slot and wants to publish a location check-in, we use the location-privacy preferences in all of the user's received profiles to recommend a location-privacy setting locally. The algorithm of our recommender is user-based CF [31], based on the assumption that people who have similar location-privacy preferences can use their data to help each other.

First, we calculate the similarities between u_i's location-privacy preferences R_i and all location-privacy preferences in $P_i^{received}$. We use 5 to represent a "share" setting and use 1 to represent a "not share" setting, following the convention from previous work [41].[4] R_i and all the Rs in $P_i^{received}$ can thus be represented as vectors that only contain 1 and 5. Then we can calculate the cosine similarity between two vectors as the similarity of two users' location-privacy preferences.

Next, we use the preferences with the highest similarities with R_i to recommend a location-privacy setting in the current category of location l and in the current time slot t. Since we can obtain the current context c_j from l and t, we denote the location-privacy recommendation for user u_i in context c_j as:

$$\hat{r}_{i,j} = \bar{r}_i + \frac{\sum_{u_k \in N_j(i)} w_{i,k}(r_{k,j} - \bar{r}_k)}{\sum_{u_k \in N_j(i)} |w_{i,k}|}$$

where $N_j(i)$ is the set of users in $P_i^{received}$ whose preferences contain a setting in context c_j. $w_{i,k}$ is the cosine similarity between the preferences R_i and R_k.

To decide whether to recommend sharing, we compare the recommendation result $\hat{r}_{i,j}$ with the median value θ (3 in this case) of the "share" setting and the "not share" setting. Then the final decision made by the recommender for u_i in the current context c_j is:

$$decision_{i,j} = \begin{cases} \textit{not share} & \text{if } \hat{r}_{i,j} \leq \theta \\ \textit{share} & \text{if } \hat{r}_{i,j} > \theta \end{cases}$$

3.2 Sampling attack

We now introduce a potential attack, i.e., the sampling attack, against our decentralised recommender system. Compared with centralised recommender systems, decentralised recommender systems are more vulnerable to the sampling attack because all nodes receive and store others' data. This enables attackers to use their received preferences from real users as samples to generate shill profiles. Hence, these shill profiles are highly similar to real profiles, which makes them difficult to be detected based on preference similarity. Figure 2 shows an example of how a sampling attacker node generates shill profiles in our recommender system. We first discuss the incentives and abilities of sampling attackers in LSSs. Then we formally describe the process of a sampling attack against our recommender system.

3.2.1 Incentives and ability

Since location-privacy recommenders automatically configure people's location-privacy settings, malicious users may conduct sampling attacks to influence the recommendations, thereby influencing people location check-in behaviours. For example, a business owner may want everyone who visits their shops to share the locations, in order to make these shops more popular on social media.

[4]Since preferences are normalised when calculating recommendations and the final decision is made by comparing the recommendation result with the median value of these two different values, the selection of these two values does not influence the final decision.

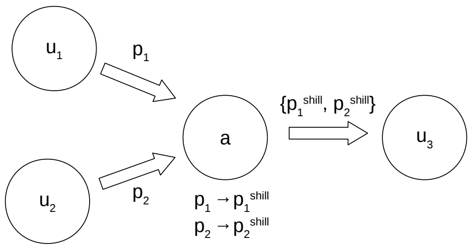

Figure 2: A sampling attacker a receives two profiles p_1 and p_2 from two real users u_1 and u_2 respectively. The attacker a then uses p_1 and p_2 as samples to generate two shill profiles p_1^{shill} and p_2^{shill}. When a meets u_3, it sends the two shill profiles $\{p_1^{shill}, p_2^{shill}\}$ rather than the two real profiles $\{p_1, p_2\}$ to u_3.

Similarly, they would want everyone who visit their rivals' places to not share the locations, in order to decrease their popularity.

We consider that sampling attackers have the following capabilities:

- Attackers can take part in the decentralised recommender systems. They can receive data from others, store the data received on their devices, and inspect the stored data.
- Attackers can generate multiple profile *id*s and use these *id*s to generate multiple shill profiles based on the received real profiles.

3.2.2 Preliminaries

We assume that a sampling attacker has a set of target contexts and intent for these contexts; for example, an attacker can have a *push* intent for one of their target locations to encourage people to publish check-ins at this location.

3.2.3 Attack Process

In both *D-Ind* and *D-Set*, an attacker first decides the set of target contexts C^{target} and their intent int. The attacker then takes part in the service to encounter other users. When the attacker encounters a user, the attacker exchanges data with the user and generate shill profiles based on the data receives from the user. Algorithms 1 and 2 give formal descriptions of the attack process in *D-Ind*. The attack process in *D-Set* is to repeat the algorithm 1 and 2 for all the profiles in the received list.

When the attacker sends shill profiles to the user, if there have not been any shilling profiles generated yet, the attacker sends an empty profile. Otherwise, in *D-Ind*, the attacker randomly selects one shill profile in P^{shill} to send. In *D-Set*, the attacker sends the whole P^{shill} to the user.

3.3 Encounter-frequency-based reputation

Since the shill profiles are generated from real profiles and the only difference between them is the ratings for the target contexts, detection based on similarity is difficult. In addition, since it is not costly for attackers to generate shill profiles, they can easily pass similarity-detection thresholds by generating more shill profiles with gradually changing similarities. Therefore, we need a solution that uses information beyond the features inside shill profiles.

Algorithm 1: CheckShill. Checking whether to make a new shill profile or to update an existing shill profile.

```
Data: $p$ is the received profile.
$S$ is a set of victim records.
Result: $P^{shill}$ is the set of shill profiles.
begin
    if $\exists s \in S, s.id^{affected} = p.id$ then
        if $p.ts > s.ts$ then
            $P^{shill} \leftarrow$ MakeShill($p$, false);
            $s.ts \leftarrow p.ts$;
    else
        $P^{shill} \leftarrow$ MakeShill($p$, true);
```

Algorithm 2: MakeShill. Making a new shill profile or updating an existing shill profile based on C^{target} and *int*.

```
Data: $p$ is a real user's profile that the attacker uses as a
      sample to generate shill profiles.
$isNew$ represents whether to make new shill profiles or to
update existing shill profiles.
$int$ represents the intent of the attacker on the target context
$C^{target}$.
Result: $P^{shill}$ is the set of shill profiles.
begin
    if $isNew$ then
        Apply a new $id^{shill}$;
        Add $(p.id, id^{shill}, p.ts)$ into $S$;
        $R^{shill} \leftarrow p.R$;
        foreach $c$ in $C^{target}$ do
            if $c$ is not rated in $R^{shill}$ then
                Change its rating in $R^{shill}$ based on $int$;
        $p^{shill} \leftarrow (id^{shill}, R^{shill}, p.ts)$;
        $P^{shill}.add(p^{shill})$;
    else
        Find the old shill profile $p^{old}$ of $p$ in $P^{shill}$;
        Update the $p^{old}.R$ and $p^{old}.ts$ based on $p.R$, $C^{target}$,
          $int$, and $p.ts$;
```

An important feature of nodes in opportunistic networks is that they physically encounter each other. The encounter frequency is decided by the number of nodes and their mobility patterns. To increase encounter frequencies, attackers need to inject more nodes carried by multiple people that have different trajectories, which is more difficult than faking profile features. We thus propose a reputation scheme based on the encounter frequency of nodes. For example, in Figure 3 (a), both u_1 and u_2 are real users and they encounter (move into communication range) three times. Because u_2 only keeps one profile p_2 on its device and it identifies itself as p_2 every time with u_1, then from u_1's perspective, profile p_2 has a reputation of 3. But for an attacker a, as shown in Figure 3 (b), and it keeps three shill profiles that are p_1^{shill}, p_2^{shill}, and p_3^{shill} on its device, for each of these encounters, it can only claim to be one of these three shill profiles. As a consequence, from u_1's perspective, the reputations of the attacker a's shill profiles would be lower than the reputations of real users' profiles, since attackers have to divide the opportunities to increase reputations for the different shill profiles they have.

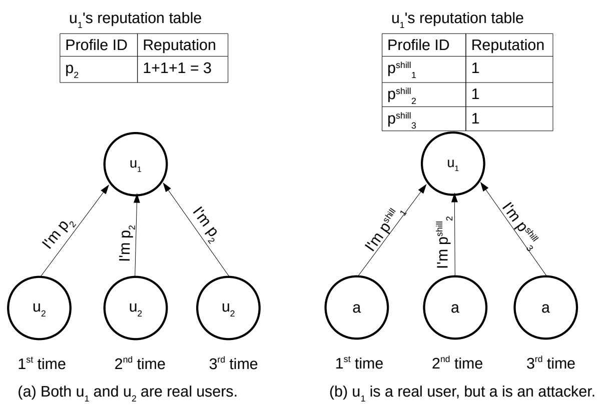

Figure 3: Encounter frequency based reputation scheme. In (a), real users u_1 and u_2 encounter three times. Each time, u_2 identifies its profile p_2 with u_1 and the reputation of p_2 is 3. In (b), a is an attacker and has to divide the same amount of reputation to its different shill profiles.

Therefore, from a user u_i's perspective, the profile p_j's reputation, $rep_{i,j}$, is the frequency by which u_i encounters p_j. Before making recommendations, u_i uses the average reputation $\overline{rep_i}$ of all the received profiles as the reputation threshold. The profiles that are used in making recommendations are

$$P_i^{candidate} = \{p_j | p_j \in P_i^{received}, rep_{i,j} \geq \overline{rep_i}\}$$

In our reputation scheme, as long as an attacker has multiple shill profiles, the encounter-frequency-based reputation has to be divided among these shill profiles. Thus each of the shill profiles is likely to have lower reputation than real profiles do. The attacker may choose to only produce one shill profile to make it have the same amount of reputation as a real profile. In this case, our reputation scheme cannot discriminate this single shill profile. However, a single shill profile's influence on biasing the recommendation results is low. If the attacker wants to increase the attack effectiveness, he or she has to deploy multiple devices that hold single shill profiles, which is an increase of attack expense.

4. EVALUATION

To evaluate the effectiveness of our decentralised opportunistic location-privacy recommender, we employ network simulations driven by real-world traces.

4.1 Simulation setup

We use the Opportunistic Network Environment (ONE) simulator [19] to evaluate our decentralised recommender in an opportunistic network scenario. Each simulation round has 24 hours, with time divided into five time slots, i.e., morning, noon, afternoon, evening, and night (Table 3). To make our simulations realistic, we use the st_andrews/locshare dataset from the CRAWDAD data archive [28], which contains the location-privacy preferences of 40 participants collected in the town of St Andrews. We simulate 40 nodes based on these users, each with different location-privacy preferences taken from the dataset. To evaluate the success of the sampling attack, we add one attacker node that uses all of its received profiles to generate shill profiles. All of the nodes' mobility patterns are restricted to a map of the road layout of the town of St Andrews. When a node moves, it first chooses its destination from the map, with the probability of choosing a point of interest (POI) as destinations is 0.8. We created five POIs that represent the locations of university buildings and night clubs in the town. Once the destination is decided, the node traverses the shortest path to this destination. After arriving, the node waits for a period of time (between 0 and 120 seconds) and then chooses its next destination. Each simulation is repeated for 100 rounds, with the initial positions and mobility patterns of nodes generated with different random seeds. Table 2 shows the details of our simulation setup.

Parameters	**Values**
simulation time	86400 seconds (24 hours) / round
time update interval	2 seconds
transmit range	10 metres
number of nodes	41 (40 real users, 1 attacker)
walking speed	0.0 m/s to 1.5 m/s
number of points of interests	5
probability of visiting POIs	80%
world size	4500 metres * 3400 metres
movement map	streets of St Andrews
movement model	shortest-path map-based movement
wait time	0 seconds to 120 seconds
router	direct delivery
number of rounds	100

Table 2: Simulation setup

Note that we do not take into account the influence of data transmit speed and storage size of nodes, as we hold them constant to compare between the various recommenders. Due to the design of our recommender, the main payload of profiles are sets of preferences, which are represented as binary vectors. Thus we believe that the transmission expense and storage expense of these preferences are uninfluential. Once two nodes move into communication range, all data exchange between them is done in one simulation update interval (2 seconds).

4.2 Recommendation performance

To test our recommender, we allocate each node in the simulation one of the 40 location-privacy preferences from the st_andrews/locshare dataset. This dataset includes six location categories:

$$L = \{Food\ \&\ Drink, Leisure, Retail, Residential, Academic, Library\},$$

while the times in the dataset are converted into five time slots:

$$T = \{Morning, Noon, Afternoon, Evening, Night\}.$$

Each instance in the dataset is in the format as $(id, t, l, decision)$ that means one location sharing *decision* (share or not) of a participant in time slot t and location category l. This *decision* is only for time when that instance was collected. During the data collection, one participant might repeat visiting same location in the same time slot, which means for the same (id, t, l) in the dataset, there are different *decision*s. Therefore, for each participant, we use their most frequent *decision* in (t, l) as the location-privacy preference r in this context.

We test the performance of our recommender on the fly. The simulation time starts at 0700 in the morning. Once a node arrives at a destination, it decides whether publish a check-in or not. The probability of publishing a check-in in different time slots are calculated from the percentage of instances of each time slot in the entire st_andrews/locshare dataset, as shown in Table 3.

time slot	**probability**
Morning (0700 – 1159)	18%
Noon (1200 – 1359)	13%
Afternoon (1400 – 1659)	23%
Evening (1700 – 2059)	28%
Night (2100 – 0659)	17%

Table 3: Probability of check-in in different time slots

If the node decides to check-in, we first convert the current simulation time in the corresponding time slot. Then if there are any settings in the node's allocated preference in the current time slot, we randomly choose one of them and compare it with the recommended setting by our recommender. The probable outcomes of these comparisons are shown as Figure 4. This setting will not be chosen again in the same round of simulation. After the comparison, the node's profile is updated by adding the tested setting into the preference of the profile and the profile's timestamp is updated to the current time. The node uses this updated new profile for generating recommendations and exchanging with other nodes in the future. To implement our recommender, we use the Lenskit recommender toolkit [14] as our user-based CF engine with a maximum neighbourhood size of 8, as tested to have the best recommendation performance using the same dataset and algorithm in our previous work [41].

		Recommended Setting	
		share	**not share**
Actual Setting	**share**	True Positive (TP)	False Negative (FN)
	not share	False Positive (FP)	True Negative (TN)

Figure 4: Confusion matrix of actual setting and recommended setting.

The *accuracy* of the recommender is calculated as:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

and the privacy *leak*, that is, the overexposure caused by the recommender, is calculated as:

$$leak = \frac{FP}{TP + TN + FP + FN}$$

4.3 Sampling attack

To simulate the sampling attack, we consider that the attacker node has the ability to generate unlimited *id*s for shill profiles. For each real user profile that the attacker receives, it generates one corresponding shill profile. Thus the attack size in our experiment is 100%. Compared with the commonly used attack size (15%) [8, 25, 39], the attacker in our simulation is stronger.

For each round of simulation, the attacker node randomly chooses one target location category and one attack intent (push or nuke), i.e., the C_{target} is the target location category combined with all the time slots. For each real user node, two parallel recommenders are stored for analysing the attack effectiveness. One of them is influenced by the attacker's input, but the other one is not. Once the real user node requests a recommendation, we compare the output of both recommenders and use $ChangedRec(C_{target}, int)$ to represent the set of changed recommendations due to the attacker's input.

The target recommendations that the attacker aims to change is the set of recommendations requested in C_{target} and, without the influence of the attacker's input, the recommendation are different from *int*. For example, if the recommendation in a target context is "not share" without the existence of the attacker, then it is a target recommendation that a "push" attacker aims to change. Similarly, all of the "share" recommendations in the target contexts are the target recommendations of a "nuke" attacker. We use $TargetRec(C_{target}, int)$ to represent the set of target recommendations. Therefore, for one simulation round, given an attacker with C_{target} and *int*, the attack success ratio is:

$$Suc(C_{target}, int) = \frac{|TargetRec(C_{target}, int) \cap ChangedRec(C_{target}, int)|}{|TargetRec(C_{target}, int)|}$$

4.4 Encounter-frequency-based reputation

To evaluate the effectiveness of our reputation scheme, we compare the attack success ratio of the attack with and without the reputation system. We refer to this reputation scheme as *D-Set-Rep* in our analysis. We also compare our reputation scheme with an existing trust model for recommender systems [27], adapting the item-level trust model from their work as location-level trust.[5] We refer to this scheme as *D-Set-Trust*. We then have profile p's trust when using it to recommend location-privacy preferences in location category l as:

$$Trust^L(p,l) = \frac{|\{(r_k, l_k) \in CorrectSet(p) : l_k = l\}|}{|\{(r_k, l_k) \in RecSet(p) : l_k = l\}|}$$

$RecSet(p)$ represents the set of all the recommendations that p has been involved in and $CorrectSet(p)$ represents the set of correct recommendations that p has been involved in. Therefore $Trust^L(p,l)$ is the percentage of correct location-privacy preference recommendations that p has made in location category l. Note that this trust value, like our reputation value, is calculated by each user locally and so different users will have different trust values for other users.

When user u_i generates recommendations in location l, for each profile p_k, we combine their similarity $w_{i,k}$ and p_k's location-level trust $TrustL(p_k, l)$ into a trust-based weighting:

$$w(u_i, p_k, l) = \frac{2 \times w_{i,k} \times Trust^L(p_k, l)}{w_{i,k} + Trust^L(p_k, l)}$$

and use this weighting to replace $w_{i,k}$ in the recommendation process.

Since a profile does not have a trust value unless it contributes to recommendations, it needs a initial trust value for bootstrap. If u_i uses p_k to generate a recommendation in location category l for the first time, we set the initial trust value $Trust^L(p_k, l)_0$ to be $w_{i,k}$, i.e., when generating this recommendation, the trust-based weighting of p_k is its cosine similarity.

In each round of the simulation, for each node, once it decides to check-in and there is a setting to be evaluated, we compare the setting with the recommendations of all the different schemes and record the comparison results. We can then use a paired *t-test* to compare the differences between the performance of these different schemes.

5. RESULTS

Our experiments aim to discover: (i) the difference in performance of centralised and decentralised location-privacy recommenders; (ii) the effect of sampling attack on our recommenders; and (iii) the effect of our reputation scheme in mitigating the sampling attack.

5.1 Decentralised recommender accuracy

We first evaluate the performance of the decentralised recommenders *D-Ind* and *D-Set*, comparing their *accuracy* and *leak* with a centralised oracle recommender *C-Rec* that has access to all of the data in the simulation. Since the *C-Rec* has more data than the decentralised recommender systems during the simulations, it acts as a benchmark for the ideal performance that *D-Ind* and *D-Set* can achieve.

Figures 5 and 6 show that *D-Ind* and *D-Set* have similar performance, with their *accuracy* being 9% and 11% lower than that of *C-Rec*. In terms of *leak*, the increases are 4% and 3% for *D-Ind* and *D-Set* respectively. These results are within our expectations, since our decentralised recommenders need more time than the centralised one to have adequate data to produce accurate recommendations. Of the decentralised recommenders, *D-Set* has better performance than *D-Ind*. To investigate why, we adapt the *message coverage* metric from other work [17] to measure the *profile coverage* of both schemes. Given one profile, its *coverage* can be measured by the number of nodes that have received this profile divided by the number of nodes that should receive this profile. As there are 40 nodes in our simulation, each time a node generates a new profile, there are 39 nodes that could receive it. For each round of simulation, we measure the average *coverage* of all the profiles. Figure 7 shows that the *coverage* in *D-Set* is much higher than in *D-Ind*. The nodes in *D-Set* not only send their own profiles to encountered nodes, but also forward their received profiles, and so once a new profile is generated, it can cover more nodes in *D-Set* than in *D-Ind*. As a consequence, the recommendation performance in *D-Set* is better. This suggests that *D-Set* is a better choice than *D-Ind* to use when implementing decentralised recommenders. In the rest of this section, we use only *D-Set* as representative of our decentralised recommender for comparing with other schemes.

The overall performance only indicates the difference between the recommenders at the end of a simulation. But performance may change over time. We group the recommendation results of each simulation round into 90-minute buckets[6] and analyse the *accuracy* and *leak* of different recommenders in each time interval. Figures 8 and 9 show that the performance difference between the decentralised recommenders and the centralised recommender is greatest at the beginning of the simulation, when the recommenders are going through cold start. As time goes on, the performance of the decentralised recommender systems approaches the centralised

[5] We adopt location-level trust rather than O'Donovan and Smyth's profile-level trust because shill profiles in sampling attack are highly similar with real profiles, which means even if they make incorrect recommendations in the target location category, their trust values can still recover by making correct recommendations in other location categories. The location-level trust model can ensure that shill profiles' trusts in the target location category are lower than real profiles since their recommendations in the target location category are always incorrect.

[6] We experimented with buckets smaller than 90 minutes, but found that some time intervals in the dataset lacked sufficient sample points for analysis.

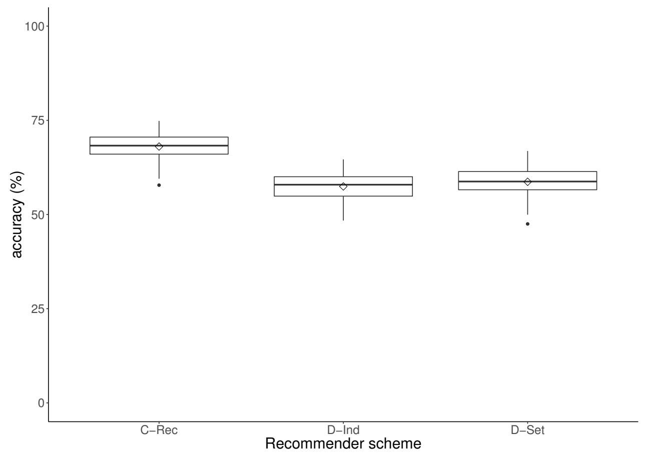

Figure 5: Overall *accuracy* of different recommenders in 100 rounds of simulations. The centralised recommender (*C-Rec*) has the highest average recommendation accuracy (68%) compared with the decentralised recommenders' *accuracy* (both $p < 0.01$). For the decentralised recommenders, the average *accuracy* of *D-Ind* is 57%, 2% lower than the *accuracy* of *D-Set*, 59% ($p < 0.01$).

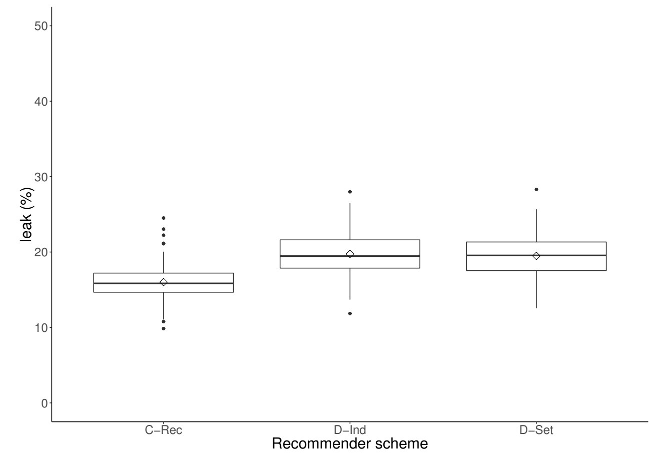

Figure 6: Overall *leak* of different recommenders in 100 rounds of simulations. As in *accuracy*, the centralised recommender (*C-Rec*) has the best performance, with an average *leak* of 16% (both $p < 0.01$). The difference between *D-Ind* (20%) and *D-Set* (19%) is not statistically significant ($p > 0.01$).

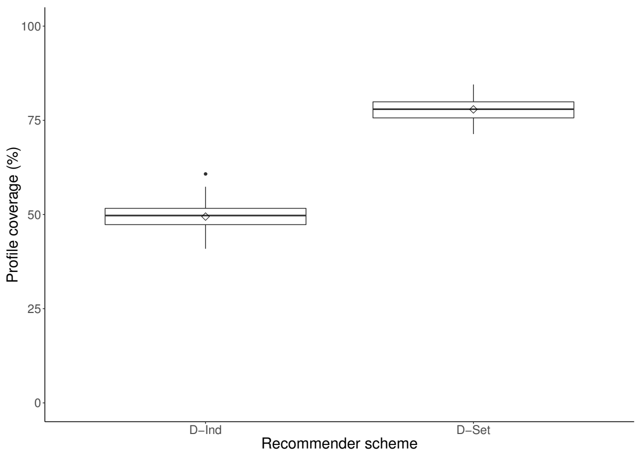

Figure 7: Overall profile *coverage* of decentralised recommenders in 100 rounds. Due to the data forwarding, the average *coverage* (78%) of *D-Set* is higher than the average *coverage* (49%) of *D-Ind* ($p < 0.01$).

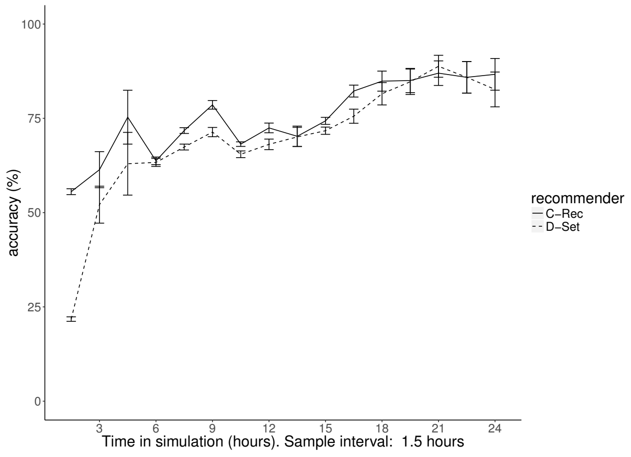

Figure 8: *accuracy* of *C-Rec* and *D-Set* over time. The difference between the two schemes' *accuracy* becomes small as the simulation time goes on. After 4.5 hours of simulation time, the *accuracy* of the two schemes are close.

recommender's performance; after 4.5 hours of simulation time, the average *accuracy* difference and the average *leak* difference between *D-Set* and *C-Rec* are 3% and 1% respectively. When LSSs are used in the real world, people are unlikely to publish all of their check-ins within one day, which means that there would be more time for our decentralised recommender to collect adequate data before making recommendations. Hence we believe the decentralised recommender's performance would be closer to that of *C-Rec* in real-world applications.

5.2 Attack Effectiveness

To study the effect of the sampling attack in our recommender system, we first examine the attack without any mitigation, i.e., without a reputation scheme.

As shown in Figure 10, across the 100 rounds of simulation, the average attack success ratio is 57%. This result is from those target recommendations whose contexts are in C^{target} and the original recommendation results are different from *int*; by simply generating 40 shill profiles (one for each real profile), one attacker node can change more than half of the recommended location-privacy settings which requested by real users in the target contexts.

5.3 Mitigation Effectiveness

To evaluate the effectiveness of the reputation scheme, we examine attack success with reputation (*D-Set-Rep*) and location-level trust (*D-Set-Trust*). their mitigation effectiveness. Figure 10 shows that with the reputation system, the attack success ratio in *D-Set-Rep* drops from 57% to 8%. Moreover, the difference between the attack success ratios of *D-Set* and *D-Set-Trust* is minimal, at only 2%. This is due to the posterior feature of the trust model, which means that there have to be enough recommendations made before trust values can accumulate. Until this occurs, attackers' trust val-

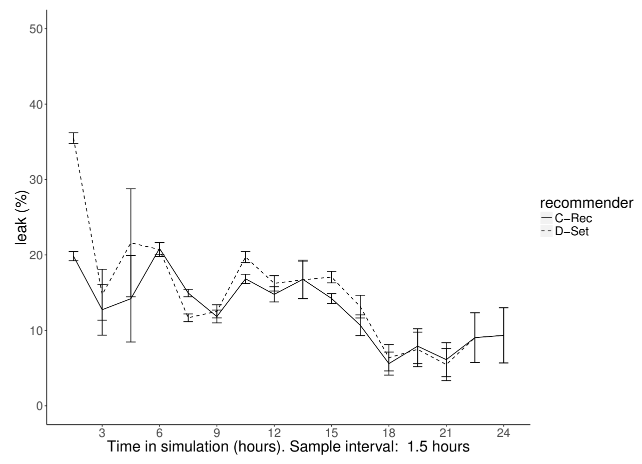

Figure 9: $leak$ **of *C-Rec* and *D-Set* with the change of simulation time. After 4.5 hours of simulation time, the** $leak$ **of the two schemes are close.**

ues are no lower than those of real users, and sampling attacks can still be successfully conducted.

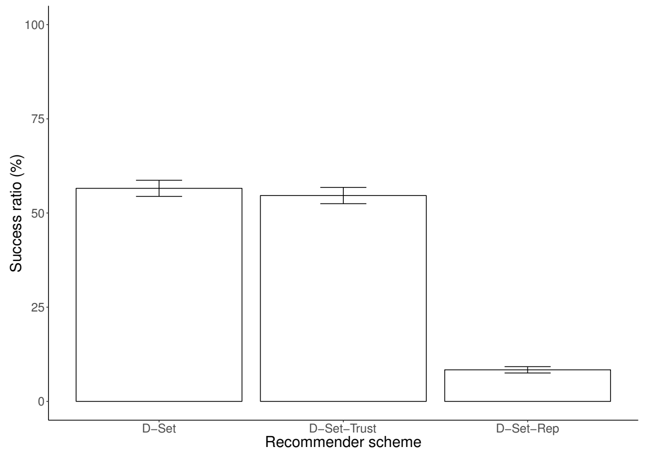

Figure 10: The percentage of successful attacks using different recommenders. The average attack success ratios of *D-Set* is 57%. The location-level trust's effect on alleviating the sampling attack is minimal. The attack success ratio of *D-Set-Trust* (55%) is 2% lower compared with *D-Set* ($p < 0.01$)**. With the reputation scheme *D-Set-Rep*, the average attack success ratio drops to 8%** ($p < 0.01$)**.**

Our results suggest that, in opportunistic networks, the encounter frequency of profiles can be used as a proxy for reputations, and can effectively alleviate the effect of shilling attacks against decentralised recommenders. As in the trust recommenders, the design of our reputation scheme is independent of the content of profiles, which means that it is difficult for attackers to bypass the reputation filter by elaborating the shill profiles. In addition, unlike the posterior trust metrics, our reputation scheme does not need recommendation results to update the reputation values, which makes it quicker to function.

Besides the mitigation effect, we are also interested whether our reputation scheme would decrease the $accuracy$ and $leak$ of the recommender, since the reputation scheme may filter out real users' profiles that have low encounter frequencies and influence the recommendation performance. Figures 11 and 12 show overall performance for *D-Set-Rep*. The reputation scheme's influence on $accuracy$ and $leak$ is 2% and 1% respectively, which is minimal compared with the overall performance.

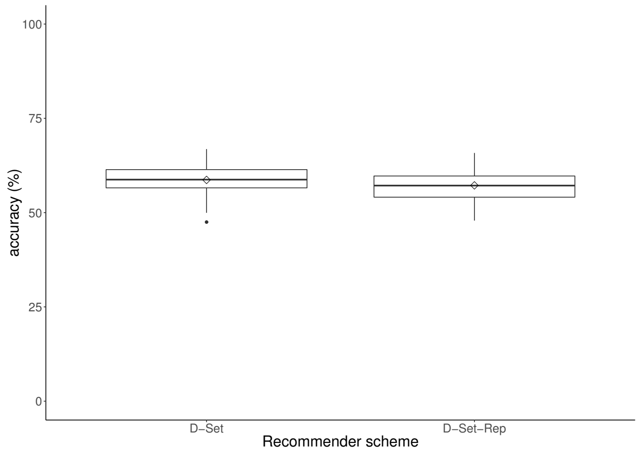

Figure 11: Overall $accuracy$ **of *D-Set* and *D-Set-Rep* in 100 rounds of simulation. The average** $accuracy$ **of *D-Set-Rep*, 57%, is 2% lower than the average** $accuracy$ **of *D-Set*, 59%** ($p < 0.01$)**.**

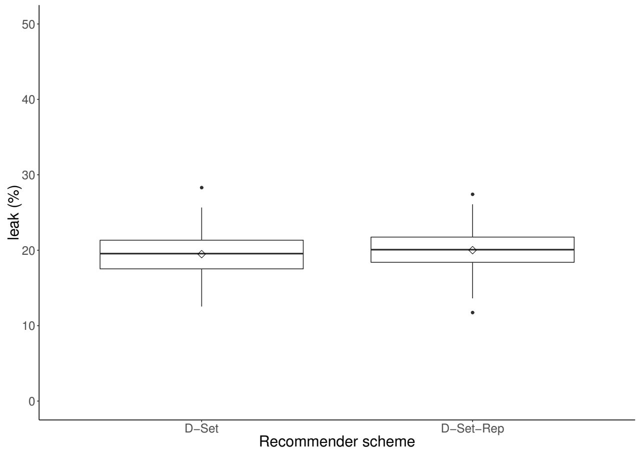

Figure 12: Overall $leak$ **of *D-Set* and *D-Set-Rep* in 100 rounds of simulation. The average** $leak$ **of *D-Set-Rep*, 20%, is 1% higher than the average** $leak$ **of *D-Set*, 19%** ($p < 0.01$)**,**

5.4 Multiple attacker nodes

Our experimental results indicate that our reputation scheme can significantly mitigate the effect of sampling attack by one attacker node generating multiple shill profiles. The reason is that each shill profile on the attacker node has fewer opportunities than real users to gain encounter-frequency-based reputation. One way to increase the opportunities for gaining reputation is to increase the number of attack nodes. We examine how many nodes the attacker needs to deploy in *D-Set-Rep* to achieve the same attack success that it can easily achieve by generating shill profiles in *D-Set*.

We set up the attacker to deploy multiple nodes in *D-Set-Rep*. In each round of simulation, all nodes controlled by the attacker have the same C_{target} and int. Whenever these nodes generate shill profiles, shill profiles generated from the same real profile have the same profile *id*. Therefore the reputation of a shill profile to a real user can be increased by encountering different attacker nodes.

Figure 13 shows that the attack success ratio goes up as we increase the number of nodes controlled by the attacker. Due to the mitigation from our reputation scheme, the attacker needs to deploy at least 30 nodes to achieve the same attack success ratio achieved in *D-Set*. Compared with simply generating shill profiles, deploying multiple nodes is more expensive. If the attacker wants the shill profiles' reputations to increase on more real users' sides, the shill profiles need to earn reputation more frequently. Therefore these attacker nodes need to be carried by different people, and these people must have diverse mobility patterns to meet more real users.

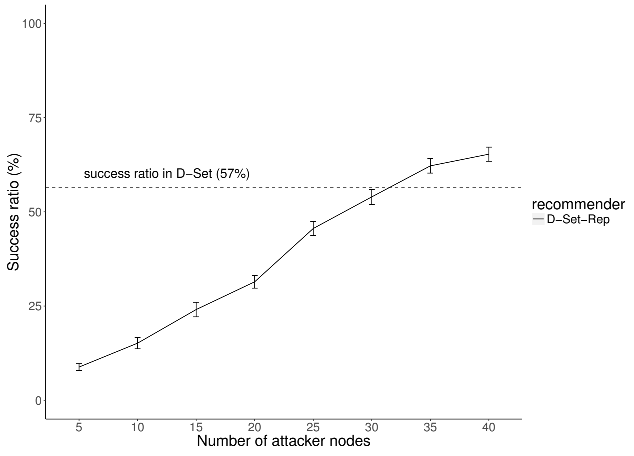

Figure 13: The change of success ratio when the attacker deploys multiple devices in *D-Set-Rep*. The dashed line is the success ratio achieved by deploying only one device in *D-Set*, i.e., without the reputation scheme. Our reputation scheme significantly increases the expense (the deployment of at least 30 devices) for an attacker to achieve the same attack success ratio.

6. DISCUSSION

Our results suggest that decentralised recommender systems based on opportunistic networks can be an alternative to centralised recommenders for recommend location-privacy settings. The decentralised recommender's performance is comparable with the centralised recommender's, once it has received adequate data. More importantly, it does not need the support of a central server. In our previous user study [42], we have found significant privacy concerns from users about providing their data to a central server and such concerns have negative effects on users' perceived recommendation quality, satisfaction, and acceptance of recommendations. Decentralisation enables us to remove the source of such concerns. However, there are limitations in such decentralised structures.

Our results show that nodes in decentralised recommenders need enough time to encounter and receive data from others. In our experiments, all of the nodes keep moving and encountering each other and they may only publish location check-ins once they arrive at destinations. These two assumptions are reasonable in LSSs where there is enough time for a node to receive adequate data between two recommendations being made. However, in some other recommendation scenarios, people may require recommendations earlier and more frequently than in LSSs. For example, music recommenders may need to provide the "Top-N" songs recommendations as soon as someone joins the service. In this case, decentralised recommenders may not be suitable to make accurate recommendations because new users have not encountered enough users to receive data.

Our results also show that using node encounter frequencies for reputation can significantly prevent our recommender system from being abused by shilling attacks. Although we only test the sampling attack in our experiments, our reputation scheme is independent of how the shill profiles are elaborated. This makes our reputation scheme also suitable for preventing other types of shilling attacks. In future work, we plan to compare our reputation scheme's mitigation effectiveness on different types of shilling attacks in decentralised recommender systems. In addition, compared with obfuscating shill profiles to bypass similarity-based detection, it is more complex for attackers to change reputations in our scheme, since their encounter frequencies are influenced by multiple features such as speed and trajectories. We also plan to investigate how more sophisticated attackers can change these features to increase their reputations.

Although we have evaluated the influence of our reputation scheme on recommender performance, we only consider linear increases in nodes' reputations. In some applications, old nodes' reputations may need to be reset periodically or increase in different ways to new nodes, as otherwise new nodes will never have chances to take part in making recommendations. We leave this for future work.

7. CONCLUSIONS

Location-privacy recommenders have been proposed to help people with their location-privacy settings. However, people have privacy concerns about their centralised structure and so may be less likely to accept the recommendations. In this paper, we propose a decentralised location-privacy recommender based on opportunistic networks that allows people to exchange data with each other through short-range communications and generate recommendations locally. Our experimental results show that the performance of such decentralised recommenders is close to that of centralised recommenders once adequate data have been received. In addition, our experiments show the effect of the sampling attack on our decentralised recommenders, and we propose a reputation scheme based on node encounter frequencies that can significantly decrease the sampling attack success ratio compared with a location-level trust model.

8. REFERENCES

[1] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal. Your location has been shared 5,398 times!: A field study on mobile app privacy nudging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 787–796, Seoul, Korea, Apr. 2015. https://doi.org/10.1145/2702123.2702210.

[2] D. Anthony, T. Henderson, and D. Kotz. Privacy in Location-Aware Computing Environments. *IEEE Pervasive Computing*, 6(4):64–72, Oct. 2007. https://doi.org/10.1109/MPRV.2007.83.

[3] M. Benisch, P. G. Kelley, N. Sadeh, and L. F. Cranor. Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. *Personal and Ubiquitous Computing*, 15(7):679–694, Oct. 2011. https://doi.org/10.1007/s00779-010-0346-0.

[4] A. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, Jan. 2003. https://doi.org/10.1109/mprv.2003.1186725.

[5] R. Bhaumik, C. Williams, B. Mobasher, and R. Burke. Securing collaborative filtering against malicious attacks through anomaly detection. In *Proceedings of the 4th Workshop on Intelligent Techniques for Web Personalization (ITWP)*, Boston, MA, USA, July 2006. Online at http://www.aaai.org/Library/Workshops/2006/ws06-10-006.php.

[6] G. Bigwood, F. Ben Abdesslem, and T. Henderson. Predicting location-sharing privacy preferences in social network applications. In *Proceedings of the 1st Workshop on Recent Advances in Behavior Prediction and Pro-active Pervasive Computing (AwareCast)*, Newcastle, UK, June 2012. Online at http://www.ibr.cs.tu-bs.de/dus/Awarecast/awarecast2012_submission_1.pdf.

[7] G. Bigwood and T. Henderson. IRONMAN: Using social networks to add incentives and reputation to opportunistic networks. In *Proceedings of the IEEE 3rd International Conference on Social Computing (SocialCom)*, pages 65–72, Boston, MA, USA, Oct. 2011. https://doi.org/10.1109/passat/socialcom.2011.60.

[8] R. Burke, B. Mobasher, and R. Bhaumik. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of 3rd Workshop on Intelligent Techniques for Web Personalization (ITWP)*, pages 17–24, Edinburgh, UK, Aug. 2005.

[9] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik. Classification features for attack detection in collaborative recommender systems. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 542–547, Philadelphia, PA, USA, Aug. 2006. https://doi.org/10.1145/1150402.1150465.

[10] P.-A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management (WIDM)*, pages 67–74, Bremen, Germany, Nov. 2005. https://doi.org/10.1145/1097047.1097061.

[11] S. Consolvo, I. E. Smith, T. Matthews, A. Lamarca, J. Tabert, and P. Powledge. Location disclosure to social relations: why, when, & what people want to share. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 81–90, Portland, OR, USA, Apr. 2005. https://doi.org/10.1145/1054972.1054985.

[12] S. Coutts. Anti-choice groups use smartphone surveillance to target 'abortion-minded women' during clinic visits. *Rewire*, 25 May 2016. Online at https://perma.cc/XD2J-LGJJ.

[13] L. Del Prete and L. Capra. diffeRS: A Mobile Recommender Service. In *Proceedings of 11th International Conference on Mobile Data Management (MDM)*, pages 21–26, Kansas City, MO, USA, May 2010. https://doi.org/10.1109/MDM.2010.22.

[14] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. Rethinking the recommender research ecosystem. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys)*, pages 133–140, Chicago, IL, USA, Oct. 2011. https://doi.org/10.1145/2043932.2043958.

[15] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 71–80, Alexandria, VA, USA, Nov. 2005. https://doi.org/10.1145/1102199.1102214.

[16] I. Gunes, C. Kaleli, A. Bilge, and H. Polat. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review*, 42(4):767–799, Dec. 2014. https://doi.org/10.1007/s10462-012-9364-9.

[17] W. He, Y. Huang, K. Nahrstedt, and B. Wu. Message propagation in ad-hoc-based proximity mobile social networks. In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 141–146, Mannheim, Germany, Mar. 2010. https://doi.org/10.1109/PERCOMW.2010.5470617.

[18] L. Hutton, T. Henderson, and A. Kapadia. "Here I am, now pay me!": Privacy concerns in incentivised location-sharing systems. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks (WiSec)*, pages 81–86, Oxford, UK, July 2014. https://doi.org/10.1145/2627393.2627416.

[19] A. Keränen, J. Ott, and T. Kärkkäinen. The ONE simulator for DTN protocol evaluation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools)*, Rome, Italy, Mar. 2009. https://doi.org/10.4108/ICST.SIMUTOOLS2009.5674.

[20] B. P. Knijnenburg, A. Kobsa, and H. Jin. Preference-based location sharing: are more privacy options really better? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 2667–2676, Paris, France, Apr. 2013. https://doi.org/10.1145/2470654.2481369.

[21] S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th International Conference on World Wide Web (WWW)*, pages 393–402, New York, NY, USA, may 2004. https://doi.org/10.1145/988672.988726.

[22] J. Lin, B. Liu, N. Sadeh, and J. I. Hong. Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings. In *Proceedings of the Symposium On Usable Privacy and Security (SOUPS)*, pages 199–212, Menlo Park, CA, USA, July 2014. USENIX Association. Online at https://www.usenix.org/conference/soups2014/proceedings/presentation/lin.

[23] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove. Analyzing Facebook privacy settings: user expectations vs. reality. In *Proceedings of the 2011 ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 61–70, Berlin, Germany, Nov. 2011. https://doi.org/10.1145/2068816.2068823.

[24] B. N. Miller, J. A. Konstan, and J. Riedl. PocketLens: Toward a Personal Recommender System. *ACM Transactions on Information Systems*, 22(3):437–476, 2004. https://doi.org/10.1145/1010614.1010618.

[25] B. Mobasher, R. Burke, R. Bhaumik, and J. Sandvig. Attacks and Remedies in Collaborative Recommendation. *IEEE Intelligent Systems*, 22(3):56–63, May 2007. https://doi.org/10.1109/MIS.2007.45.

[26] J. Mugan, T. Sharma, and N. Sadeh. Understandable Learning of Privacy Preferences Through Default Personas and Suggestions. Technical Report CMU-ISR-11-112, Institute for Software Research, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, Aug. 2011. Online at http://reports-archive.adm.cs.cmu.edu/anon/isr2011/abstracts/11-112.html.

[27] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI)*, pages 167–174, San Diego, CA, USA, Jan. 2005.

https://doi.org/10.1145/1040830.1040870.

[28] I. Parris and F. Ben Abdesslem. CRAWDAD data set st_andrews/locshare (v. 2011-10-12). Downloaded from http://crawdad.org/st_andrews/locshare/, Oct. 2011. https://doi.org/10.15783/C7WW2F.

[29] L. Pelusi, A. Passarella, and M. Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*, 44(11):134–141, Nov. 2006. https://doi.org/10.1109/mcom.2006.248176.

[30] D. Quercia, S. Hailes, and L. Capra. MobiRate: making mobile raters stick to their word. In *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp)*, pages 212–221, Seoul, Korea, Sept. 2008. https://doi.org/10.1145/1409635.1409664.

[31] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 175–186, Chapel Hill, NC, USA, Oct. 1994. https://doi.org/10.1145/192844.192905.

[32] N. Sadeh, J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker, and J. Rao. Understanding and capturing people's privacy policies in a mobile social networking application. *Personal and Ubiquitous Computing*, 13(6):401–412, Aug. 2009. https://doi.org/10.1007/s00779-008-0214-3.

[33] R. Schifanella, A. Panisson, C. Gena, and G. Ruffo. MobHinter: Epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys)*, pages 27–34, Lausanne, Switzerland, Oct. 2008. https://doi.org/10.1145/1454008.1454014.

[34] J. Staiano, N. Oliver, B. Lepri, R. de Oliveira, M. Caraviello, and N. Sebe. MoneyWalks: A human-centric study on the economics of personal mobile data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, pages 583–594, Seattle, WA, USA, Sept. 2014. https://doi.org/10.1145/2632048.2632074.

[35] E. Toch. Crowdsourcing privacy preferences in context-aware applications. *Personal and Ubiquitous Computing*, 18(1):129–141, Jan. 2014. https://doi.org/10.1007/s00779-012-0632-0.

[36] J. Y. Tsai, P. G. Kelley, L. F. Cranor, and N. Sadeh. Location-sharing technologies: Privacy risks and controls. In *Proceedings of the Research Conference on Communication, Information and Internet Policy (TPRC)*, Aug. 2009. Online at http://ssrn.com/abstract=1997782.

[37] G. Wang, S. Y. Schoenebeck, H. Zheng, and B. Y. Zhao. "Will check-in for badges": Understanding bias and misbehavior on location-based social networks. In *Proceedings of the 10th International AAAI Conference on Web and Social Media (ICWSM)*, pages 417–426, Cologne, Germany, May 2016.

[38] C. Williams, B. Mobasher, R. Burke, J. Sandvig, and R. Bhaumik. Detection of obfuscated attacks in collaborative recommender systems. In *Proceedings of the ECAI 2006 Workshop on Recommender Systems*, pages 19 – 23, Riva del Garda, Italy, Aug. 2006.

[39] C. A. Williams, B. Mobasher, and R. Burke. Defending recommender systems: detection of profile injection attacks. *Service Oriented Computing and Applications*, 1(3):157–170, Nov. 2007. https://doi.org/10.1007/s11761-007-0013-0.

[40] J. Xie, B. P. Knijnenburg, and H. Jin. Location sharing privacy preference: analysis and personalized recommendation. In *Proceedings of the 19th International Conference on Intelligent User Interfaces (IUI)*, pages 189–198, Haifa, Israel, Feb. 2014. https://doi.org/10.1145/2557500.2557504.

[41] Y. Zhao, J. Ye, and T. Henderson. Privacy-aware Location Privacy Preference Recommendations. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous)*, pages 120–129, London, UK, Dec. 2014. https://doi.org/10.4108/icst.mobiquitous.2014.258017.

[42] Y. Zhao, J. Ye, and T. Henderson. The Effect of Privacy Concerns on Privacy Recommenders. In *Proceedings of the 21st International Conference on Intelligent User Interfaces (IUI)*, pages 218–227, Sonoma, CA, USA, Mar. 2016. https://doi.org/10.1145/2856767.2856771.