Protecting Location-privacy with Appointment Cards in OMSN

Authors Name/s per 1st Affiliation (*Author*)

dept. name of organization

name of organization

City, Country

e-mail address if desired

Authors Name/s per 1st Affiliation (*Author*)

dept. name of organization

name of organization

City, Country

e-mail address if desired

Authors Name/s per 1st Affiliation (*Author*)

dept. name of organization

name of organization

City, Country

e-mail address if desired

*Abstract*—Users face location-privacy risks when accessing Location-Based Services (LBSs) in an Opportunistic Mobile Social Networks (OMSNs). In order to protect the original requester’s identity and location, we propose a location-privacy obfuscation protocol Appointment Card (ACP) protocol that utilizes social ties between users. To facilitate the obfuscation operations of queries, we import a kind of message called the Appointment Card (AC). The original requesters can send their queries to the LBS directly using the information in the CA, so that the query time cost of ACP is similar as no-privacy protocols, while it avoids the original requester being detected by the LBS. Also, a path for reply message is built when the query is sent, thus saving lots of time. Simulation results show that our protocol can reach a higher query success ratio comparing to the existing protocol.

Keywords—obfuscation; encryption; location-privacy; opportunistic mobile social networks

# Introduction

Location-privacy is becoming a major concern in OMSNs which is a kind of Delay Tolerant Networks (DTNs) [1] featuring lack of continuous connectivity. More specifically, in OMSNs, it is not necessary for senders to have an end-to-end routing path to their destinations. Nodes make contacts when they encounter each other. LBSs are common applications in OMSNs and they are widely used in "military and government industries, emergency services and the commercial sector" [2], especially after the proliferation of localization technologies, like GPS. LBS users often send their location to LBS providers. Many people access to LBSs with their portable devices, so their locations are also bound to the devices. In this case, LBS users face a continuous risk that their location may be leaked from LBS applications. This makes people unwilling to use LBSs. Thus, protect location privacy has been a critical issue in LBSs.

Early location privacy protection methods such as obfuscation algorithms generate anonymized areas for the original LBS requesters so that the requesters are mixed in a group of other LBS users, like [3]. Users send an anonymized area instead of an exact coordinator to LBS providers when launching a LBS request. After that, social ties are imported into obfuscation algorithms to improve their performances in security. [4] is a typical one among these algorithms. It uses social ties to determine trustable friends who could be chosen as agencies to forward obfuscation queries. [5] and [6] are algorithms which aims to improve its delivery performance. However, compared to [4], the query success ratios of [5] and [6] increase only 5% and 11% respectively, which are not obvious.

These papers assume that attackers can access the LBS servers, which enables them to learn LBS users’ identities and locations. If a user sends a query to a LBS servers with his real identity and location (e.g., a query asking for a path from his current location to a certain place), attackers can locate the user easily. Therefore, hiding the original requester’s real identity and location is the key of this problem.

Our protocol ACP is also built on the above assumption. It focuses on the location-privacy of the users instead of the LBS server. In other words, the location of the LBS server is published, while the locations and the identities of the users are privacy. More specifically, the goal of our protocol is protecting the original requester’s location-privacy including his identity and location, because the attacker can infer where the user is and where he will go easily if a user uses his own identity and location to send a query. Besides, the attacker can also infer other information based on those information, like health, status, preference and so on.

The basic idea of the ACP is using the identity of another user who is called the agency to send the original requester’s query. Even though that strategy reveals the agency’s identity, it does not cause any harm to the privacy of him for the reason that there is no relationship between the agency and the query, so attackers cannot infer any useful information of the agency. For example, Alice is the original requester, and Bob is an agency. Alice sends a query to find a restaurant nearby using Bob’s identity. When the attacker learns Alice’s query, he believes that Bob is the person who is going to a restaurant near a certain place instead of Alice. While Bob is not at the location shown in the query, neither does he want to go to a restaurant. Therefore, attackers can not infer any information about Alice nor Bob.

Inspired by [4] and [7] that uses social network for messages forwarding in mobile ad hoc networks, we propose the distributed location-privacy algorithm ACP to guarantee location-privacy and reach a higher query success ratio. The introduction of social networks enables us to hide the original requester’s information behind his friends. We also introduce the Appointment Card (AC) as a kind of intermediary which records a serial of agencies, so that the query of the original requester uses the identity of the first agency in the AC, also the reply of the query can be delivered back to the original requester along the serial of agencies. The last agency called trusted agency in the AC is a user in the social tie of the original requester, in other words, the last agency is a friend (or a friend of friends) of the original requester. The trusted agency separates the strangers in the AC and the original requester, so that no stranger knows the identity of the original requester.

The process of distributed social based location privacy protocol (SLPD) in [4] also uses the social tie to protect users’ location-privacy. When a user wants to send a query to the LBS server, he sends the query to a friend first. The friend forwards the query to the next friend, and that process repeats for times, which is called the obfuscation phase, where is a system parameter. The last friend sends the query to the LBS server using his own identity and location.

It is obviously that the obfuscation phase in [4] might cost a lot of time, especially when the users do have few friends. As a result, the query expires before it is delivered to the LBS with a high probability, so that the query success ratio cannot be high. In the ACP, original requesters already get the information of agencies from ACs, which enables them to send the query to the LBS server directly using the identity of the first agency.

Simulation results show that the query success ratio of the ACP is similar as that of the no-privacy protocol Binary Spray and Wait (BSW) [19]. It saves so much time when sending the queries that the original requester can receive the reply earlier than that in [4], even though the process of the reply part is more complex in the ACP than that in [4].

The rest of this paper is organized as follows: section II presents related protocols in privacy-protection. The process of ACP is described in III section, then in the IV section, we show our experiment results. Lastly, we conclude the paper.

# Related work

Users face risks when they access to a semi-trusted LBS provider, because anyone who has access to data in LBSs is able to threaten LBS users’ location-privacy. Considering that LBSs rely on location-aware computing, it is unavoidable to leak users' location from LBSs. Therefore, balancing “these two competing aims of location privacy and location awareness” [8] is always a challenge.

Some early solutions, like [9] and [3], are generating a specific area based on *k*-anonymity [10] for each user who needs to send queries. For example, [9] gives a rectangle as an anonymized area, in which all nodes form a group to hide the original requester. But it requires at least *k* connected agents to complete its obfuscation process. [3] uses a central anonymity server as a mix router. As a result, it is necessary for each node to have a continuous connection with the server. That is hard to achieve in a spare DTN. With a similar problem as in [3], [11] employs a matchmaker which is used to match users and advertisements, then users can achieve anonymization of their identities and locations from the matchmaker. However, the matchmaker is a high-risk in the network, because it collects so much private information. In the work in [12], exact locations and requests from clients are replaced by a location anonymization engine before they arrive at LBS providers. Since the anonymization engine learns all exact locations and requests, it becomes a better target for malicious attack.

There are protocols with more servers. The servers are settled in the network and each one of them takes charge of a certain area. [13] uses roadside units (RSUs) as mix servers in a vehicular DTN, and the destination is encrypted during forwarding, so eavesdropping queries cannot help attackers to locate users. But deploying the RSUs is not always feasible. In [14], sensor nodes which are scattered throughout the network provide anonymized locations for users. Since the sensor nodes’ coverage should possess a non-overlapping characteristic, it is difficult to deploy them in real-world. Besides, the mix servers and sensor nodes might be more prominent targets than LBS providers.

However, the system could also be a distributed one, like protocols [4], [5] and [6]. The obfuscation processes are performed by each separated node independently without any help from a third-part device. The utilization of label as [15] makes it easier for nodes to mix themselves into a group, which is a significant difference from these protocols and the previous ones. Algorithms above use groups instead of an area to protect users. [5] imports 2 concepts: the obfuscation phase and the free phase. In the obfuscation phase, queries must be transmitted between friends for *k* times. When there are only a few friends in the network, it is hard for a node to find an available next hop in obfuscation phase. [5] and [6] attempt to improve [4]’s performance in the obfuscation phase, which might be a safety tradeoff, because some ineligible users in [4] are chosen as friends based on the additional standards imported by [5] and [6].

# Appointment Card Protocol

## System model

The network architecture consists of two main entities: Users and Location-Based Service Providers (LBSPs). Users are mobile and communicate with others in a certain range, e.g., the communication range of their portable devices. For a user, other ones in the social network are either strangers or his friends whom he can detect when they are in his communication range. Let denotes the relationship strength between user and , if is larger than a specific friend threshold , user and are friends. LBSPs are fixed and not connected in a social network, which provides Location-Based Services (LBSs) for users. We assume that the only information which is necessary for the LBS is the location of the original requester (), but he should still give an identity to the LBSP so that the LBSP can reply to that identity.

Attackers can access the database of LBSPs, so that they can learn everything recorded in LBSPs, including users’ identities and locations. If sends queries with his own identity and accurate location, his will be located by attackers easily, so the key is to hide his identity and location. We propose an Appointment Card Protocol (ACP) to protect the identity and location-privacy of the original requester by providing a fake identity which disenable attackers to locate the original requester using records in the LBSP database.

## Appointment Card Protocol Overview

We propose an Appointment Card Protocol (ACP) to protect original requesters when they are served by LBSPs. A user () generates his own Appointment Cards (ACs) containing his own identity, a unique number called (Creator Appointment number) and a location. The appointment cards are exchanged when he encounters another user. When the original requester sends a query, he chooses an AC and sends the query using the identity which is in the AC. A LBSP replies to when it receives the query. is actually the one who generates the AC and the first agency of the AC. He re-transmits it to the next agency () whom he exchanged the AC to, and so do the following users until the reply reaches the last agency. The last agency is responsible to forward it to .



Fig. 1 Example

As shown in Fig. 1, the whole process can be considered as the following parts: 1). exchanging cards among all users who are called agencies (i.e., 1 and 2); 2). exchanging cards among friends (i.e., 3); 3). sending query using information of appointment cards (i.e., 4); 4). forwarding the reply among agencies (i.e., 5, 6 and 7); 5). relay to the original requester (i.e., 8). This figure will help us to describe the protocol in the whole paper.

## Appointment Card

Since the original requester cannot use his own identity to communicate with the LBS provider (LBSP), he must use others’ identity () to send queries, so that the LBSP can reply to the original requester through . Appointment cards make it possible that agencies forward the reply to one by one. In other words, the appointment card indicates a path, through which can get its reply. In the Fig. 1, we can see that , and are strangers, so that attackers can hardly infer from the identity . is a friend in ’s social tie (’s friend or his friends’ friend, ...) and is also the only one who knows how to reach . Therefore, it is hard for attackers to infer from .

Notice that received ’s appointment card from a stranger who knows the information of that appointment card and the identity of the next agency , so that it is not safe for to use the appointment card. The appointment card cannot be used until exchanges it to a friend (e.g., , and ). The friend (i.e. , or ) can use it to send queries or exchange it to other friends. The appointment card is called ready appointment card after it arrives a friend, or it is called distributing appointment card.

All users in the network are responsible to generate their Appointment Cards (ACs) respectively and they are called the creators of their own ACs. The creator writes down his own identity (), a unique Creator Appointment Number () on his AC. When users exchange appointment cards, they modify (the Agency ID) and (the Agency appointment number) in the AC to enable the next agency identify whom is the previous one. Entries of AC are shown in TABLE I.

TABLE I APPOINTMENT CARD

|  |  |
| --- | --- |
| Entry | Explanation |
|  | The identity of the creator who generates the AC. |
|  | A unique number that distinguishes an AC from other ones generated by the same creator. |
|  | The identity of an agency who gives the AC to the recent holder. (The previous agency) |
|  | A unique number that distinguishes an AC from other ones transmitted by the same agency. |
| timeout | The time when the AC expires. |
| EQ | A queue (Exchange Queue) which records users who exchange the AC in order. |
| EQL | The length of EQ. |

## AC life cycle

The life cycle of an AC starts when it is generated by its creator. The first (see TABLE II) agencies add their identities into its EQ before exchanges it, which increase the length of the EQ. When the AC’s EQL reaches , it is eligible to be used in a query and called a ready AC. A ready AC cannot be exchanged to strangers. When an AC is used in a query, it is marked as a used AC by the original requester. No matter what state (distributing, ready or used) an AC is in, it can expire.

TABLE II SYSTEM PARAMETERS

|  |  |
| --- | --- |
| Parameters | Explanation |
|  | The minimum length of the EQL so that an AC becomes a ready AC. |
|  | The number of distributing ACs’ lists per user. |
| GS | The generating speed of appointment cards |
| AC Timeout (AT) | The timeout for appointment cards |

### Obfuscation distance

The obfuscation distance is the number of exchange before an AC is switched to the ready state. In other words, an AC must be exchanged for times before it becomes a ready AC.

As shown in Fig. 2, an AC is exchanged by , , ..., . Since those agencies are strangers, the only relationship between two adjacent agencies is that they encounter each other somewhere. The relationship between and becomes weaker when we increase k. In other words, attackers can hardly infer the identity of when he only knows the identity of , and his difficulty increases with the parameter . As the original requester is in the social tie of , it is hard for attackers to infer the original requester, too.



Fig. 2 obfuscation distance

### Friends obfuscation distance

Since is a stranger for , it is possible that is exactly the attacker. We assume that the attacker knows that the original requester’s () is in the social tie of . The attacker can assume that is a close friend of , so the identity of gives the attacker a good tip to infer . A solution to prevent the agency from learning easily is that the last agencies are friends, as shown in Fig. 3. The is called friends obfuscation distance, and the last m agencies are called trusted agencies.



Fig. 3 friends obfuscation distance

It is true that is a friend of , . But two trusted agencies who are not adjacent might have a weak relationship (e.g., a friend of friends). Since there are at least trusted agencies between the stranger and , can hardly infer based on information he learns. When , all ACs can only be exchanged between friends only. When , the last agency is the only one who is in the social tie of the original requester.

However, friends encounter each other rarely so that a big increases the difficulty of distributing ACs. In this paper, we assign to 1.

### Distributing segment

We assume that generates 3 ACs (i.e. , and ). It exchanges these ACs to , then exchanges them to and so on. At last, all of them reach . In this case, has several ACs whose routes (the list of agencies) are the same, so that has no choice. In other words, since using these ACs results in the same reply routes, is restricted to choose an optimal AC based on specific situation.

If agencies exchange ACs to different users, the above problem will not happen. We use a system parameter distributing segment to avoid exchange all ACs which are received together to the same user. Each user has distributing AC lists. It put received distributing ACs in one of those lists randomly. If exchanges ACs with another user, selects one of the lists randomly. Only ACs in that list will be exchanged to the other user.

For example, we assign . When receives , and from , may put and in its list1, while in its list2. If encounters , can exchange either and or only to . In this way, we separate ACs generated by the same creator.

### Generating Speed

Since ACs could expire, users must generate new ACs continuously. We use GS to denote the speed of generating new ACs per user. Each user generates a new AC every GS second.

### AC timeout

Let denote the timeout of ACs. An AC expires after it is generated for seconds. When an AC expires, all agencies delete its information from their memory.

## Generating appointment cards

Maintaining is a certain number of ACs is a prerequisite for a user to sending queries, so that users must generate ACs continuously on 2 principles, which are fairness and continuity.

ACs impose burdens on creators and agencies. In fact, agencies (a creator must be an agency) are unlikely to benefit from relaying messages to the original requester. Because they need to allocate memories to save ACs’ information and cost more energy for forwarding replies. To avoid users going into overload, fairness is an essential part of the generating strategy. In other words, users should generate a similar amount of ACs.

We assume that the AC timeout is 30 minutes, and every user generates 100 ACs at the beginning (the 0th minute) and generates no ACs until the 30th minute. It is hard for a user to select an appropriate AC for his query at the 29th minute, because all agencies of a selected AC will remove its information from their memories 1 () minute later, so that they cannot forward any replies, when the AC expires. Therefore, the generating strategy should be a steady and sustainable, instead of an intensive one.

In our protocol, each user generates a new AC every GS second. Since an AC has a AT-seconds timeout and we assume that no AC is used during that time, there are about appointment cards hold by one user in the network. In other words, each user maintains about ACs which is generated by himself, and we achieve the fairness part. Since users generate ACs continuously, ACs have various timeout so that it is likely for a user to pick an AC who does not expire in a long time.

## Exchange distributing appointment cards

Users exchange their distributing appointment card as frequently as possible, so that a distributing appointment card can be switched to a ready one quickly. Still, there are some other conditions which should be obeyed when exchanging a distributing appointment card.

We assume that two friends (e.g., A and B) walk together for a long time. If A generates an AC (e.g., ) at and exchanges it to B, then is given back to A and so on. Then the last agency might be A or B, which loses the meaning of exchanging, because it is easy for an attacker to infer the last agency from the creator A.

We import an parameter to optimize the agency selection strategy. If a user gets an AC at time , he cannot get that AC again before . For example, the user B receives from the user A at , then B send it to a user C, and C can also send it to others. If a user carrying encounters B before , he cannot send to B. Therefore, if the parameter is infinite, there is no repeated agency in an AC’s EQ. In other words, an AC never reaches an agency twice with an infinite .

Besides, we should also avoid ACs to having the same sequence of agencies, as we mention in the distributing segment subsection. Now we propose the strategy of exchanging distributing ACs.

Let us take a pair of users Alice and Bob as an example. If Alice encounters another user Bob, Bob tells Alice whether he trusts her (She is viewed as a friend by Bob). Alice picks one of her distributing AC lists (e.g. list1). Alice traverses all ACs in the list1, and ACs which match all the following two conditions are exchanged to Bob.

1. If the length of the AC’s EQ (i.e. EQL) is not shorter than , then Alice must be a friend of Bob.
2. Bob does not carry the AC in the recent (time).

When Alice is sending a distributing AC to Bob, she adds her identity and recent time to the AC’s EQ. Besides, Alice must modify the AC’s to her own identity and its to a new one. The information in TABLE III are recorded in her memory where is called the relay table. We should notice that the first agency (i.e., ) does not have a , because he is exactly the one who generates the AC, then the should be his own identity. When the user B gets those ACs, he puts each one of the received ACs to its AC lists respectively and randomly.

TABLE III RELAY TABLE



Fig. 4 constitute query

|  |  |
| --- | --- |
| Entries | Explanation |
|  | The generated by the previous agency |
|  | The identity of the previous agency |
|  | The generated by the current agency |
|  | The identity of the next agency |
| EQL | The length of the AC’s EQ (larger than 0) |
| AT | The time when the AC expires. |

## Exchange ready appointment cards

Users asks for Ready ACs only from their friends, which ensures that the information of the Ready ACs hold by a user is not exposed to strangers. The strategy of exchanging ready ACs faces 2 main problems. First, some users send much more queries than others; second, the number of each user’s ready ACs should be similar.

We also take friends Alice and Bob as an example. Alice has 20 ACs, while Bob has 10 ACs. When they encounter each other, it is reasonable that they both gives half of their ready ACs to the other one. As a result, they both will have half of total (15) ACs. However, that strategy does not work all the time.

The problem becomes more complex in the following condition. Alice is a friend of Bob, while Bob is not a friend of Alice. In other words, Bob trusts Alice, but Alice does not trust Bob. We assume that Alice is a trustful but suspicious girl, so that many users trust her while she trusts few users while Bob is opposite. When users encounter Alice, they ask for ready ACs from her, but Alice rarely asks for ready ACs. Therefore, Alice carries few ready ACs, while Bob carries so many ready ACs. When Alice and Bob encounter each other, it is Bob who is asking for ready ACs instead of Alice. To make the strategy fair and efficient, users must compare the number of their own ready ACs and the number of ready ACs carrying by the other user when two users are exchanging ready ACs.

When Bob encounters Alice, Bob tells Alice the number of his ready ACs (i.e., ) and whether he wants Alice’s ready ACs (if he trusts Alice). If Alice learns that Bob needs her ready ACs, she compares the number of her ready ACs (i.e., ) and . If , Alice gives no ready AC to Bob; otherwise, she gives to Bob. In this way, ready ACs do not concentrate in a group of users who trust many users. Users do not modify any information in the ready ACs including and , so that ready ACs keep static after they leave the last () agency.

## Sending queries

For an original requester who want to send queries to the LBSP, his query must be delivered to the LBSP while the LBSP cannot learn the identity of him. Besides, the reply from the LBSP must be delivered to . The basic idea is that sends his query using another user ()’s identity to the LBSP, so that LBSP can reply to , then forward the reply to .

In order to enable LBSP to reply to , ’s query includes an identity , which is the in the AC. also needs the to identify the AC used by the query (and the reply). So that the query includes the and the in the AC, as shown in Fig. 4. The network can deliver that query to the destination LBSP easily with any DTN protocols.

The AC is marked as used when the query is ready to be sent, so that the AC cannot be used again in any other queries. Neither can it be exchanged to other users.

## Sending replies

### The LBSP part

When the LBSP received the query, it learns that the sender’s identity is instead of , which protects the location privacy of . The LBSP reply to using the information in the query, as shown in Fig. 5.



Fig. 5 constitute replies

The receiver of the reply is , and the reply also includes the which is equal to the in the query.

### The first agency

When the first agency (i.e., ) gets the reply from the LBSP, he learns the in the reply. Since he is the first agency (and the creator) of the AC used in the reply (and the query), the information of that AC is in his reply table. There must be an entry as shown in TABLE IV in his reply table, if that AC does not expire. He uses his identity () and the in the reply as the key to search his reply table, so that he knows the identity of the next agency (i.e. ), which enables the first agency to forward the reply to the correct agency . The unique number can help when he checks his reply table.

TABLE IV reply table entries of the first agency

|  |  |
| --- | --- |
| Entries | Values |
|  |  |
|  |  |
|  |  |
|  |  |
| EQL |  |
| AT | The time when the AC timeout. |

The first agency modifies the reply’s sender ID (to ), receiver ID (to ) and (to ), then forward the reply to .



Fig. 6 the reply of the first agency

### Intermediate agencies

The process of forwarding replies for the intermediate agencies (the second to the one) is similar with that of the first agency. We take the second agency as an example. When the second agency receives the reply forwarded by the first one, he learns the which is equal to (see Fig. 6) from the reply. There should be an entry in his reply table as shown in TABLE V.

TABLE V reply table entries of the second agency

|  |  |
| --- | --- |
| Entries | Values |
|  |  |
|  |  |
|  |  |
|  |  |
| EQL |  |
| AT | The time when the AC timeout. |

He uses the sender ID () and the of the reply as a key to search his reply table, so that he knows the identity of the next agency (i.e. ), which enables him to forward the reply to . The unique number can help when checks his reply table.

For each agency , where , he searches his reply table for a correlative entry, when he receives a reply. The and in the entry should equal to the sender ID and the in the reply. The is the next agency whom is responsible for forwarding the reply to. also assign to the in the reply to help (must be ) search ’s reply table. The modified reply is shown in Fig. 7.



Fig. 7 the reply of the second agency

### The last agency

The last agency also searches for a reply table entry based on the reply, while he does not get the identity of the next agency, because he is the last one. The reply table entry of the last agency is shown in TABLE VI.

TABLE VI reply table entries of the last agency

|  |  |
| --- | --- |
| Entries | Values |
|  |  |
|  |  |
|  | void |
|  |  |
| EQL |  |
| AT | The time when the AC timeout. |

The last agency finds that the EQL is equal to , so he notices that he is the last agency. He has no agency to be forwarded to, but forwards to the original requester. He uses his identity () and to generates a pseudonym . The function is a public pseudonym generating function that everyone in the network knows, including the original requesting .

When sends his query, he also gets the same pseudonym using the pseudonym generating function. Note that he can get parameters from the AC. and are equal to the and in the AC. He uses that pseudonym as his identity before he gets the reply.

When users deliver the reply from the last agency, they are looking for a user whose identity is that pseudonym. Since the original requester is the only user who uses the pseudonym as his identity, gets the reply.

## Appointment number

The appointment number including and is significant information in the AC. We explain the rules of generating them in detail and talk about the effect of ACs’ timeout mechanism in this section.

### Card appointment number

The card appointment number () is a number which is used to identify ACs generated by the same creator. In other words, if two ACs are generated by the same creator and they do not expire, their must be different. Therefore, when the first agency looks up his reply table, he cannot find two entries which have the same pair of and .

### Agency appointment number

The agency appointment number () is a number used to identify ACs who have the same agency. Agencies generate a new for ACs before they exchange those cards to others. In other words, an agency gives every AC passed on by him a unique , which helps the next agency search for the AC in his reply table. Since the is unique, an agency who is not the first one cannot find two entries which have the same pair of and , neither.

### Timeout

Agencies delete the entries, which contains the information of expired ACs, from his reply table. Consequently, agencies cannot forward replies using ACs which expire before this very moment. That is the reason why an original requester must choose a AC which expires after his query and reply timeout. While the timeout mechanism has more advantage than disadvantage.

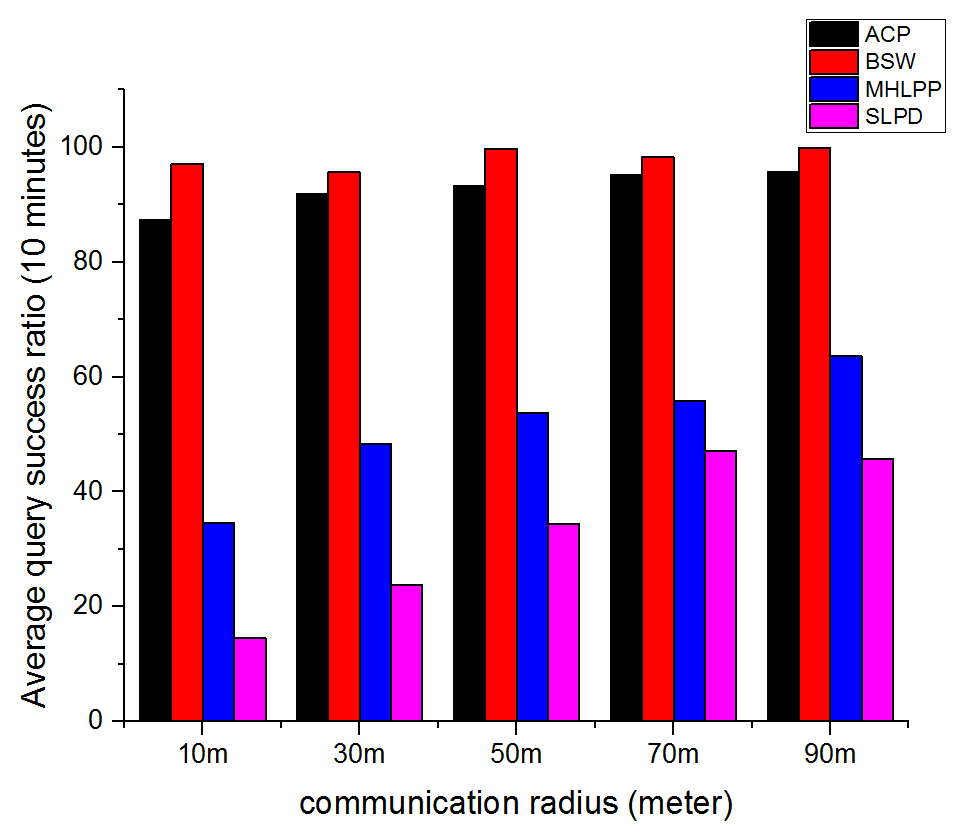


Fig. 8 average query success ratio (10 minutes)

Since users are moving, the distance between agencies and the original requester might be too large after a long time. As a result, it is hard for agencies to forward the reply back to the original requester, so that the original requester takes a higher risk when he uses an AC which is generated for a long time. Then these kind of ACs might not be used after a period, while it cost agencies a few memories to save the ACs’ information in their reply table. Therefore, all users remove the information of expired ACs to save their memories.

We should also notice that an unexpired AC and an expired one might have the same or . Because no agency keeps the record of the expired AC, so that the duplication cannot confuse agencies.

# Performance simulations

We use the map of Helsinki in our simulator to evaluate ACP. It is also compared against Binary Spray and Wait (BSW), distributed social based location privacy protocol (SLPD) and our Multi-Hop Location-Privacy Protection (MHLPP). Users are moving on the map along streets continuously, while there is a LBSP fixed at a random location on the map.

For each user, we give him random social values between 0% and 100%, each corresponding to all other users. Each value has the same probability, so we can compute the expected number of friends of a user. If a user whose social value is larger than 85% is called a friend and there are n users in the network, there are friends.

The Shortest Path Map-Based Movement (SPMBM) is used in our experiment. For each experiment, we give the simulator a random seed so that it can generate pseudo-random number based on the seed. Therefore, all the factors including users’ speed and locations are the same if two experiments have the same random seed. All those four protocols are tested using the same serial of random seed.

Before each experiment, the simulator runs for 800 seconds (simulator time). Then we pick 100 users out of 126 users randomly, and each of them send a query to the LBSP. Tests last for about 20 minutes (simulator time).

## Average query success ratio

The query success ratio is the percentage of delivered queries among a number of attempts. Since users sending 100 queries in each experiment totally, if queries are delivered to the LBSP at time , the query success ratio of time is .

As shown in Fig. 8, we compare the average query success ratio of the four protocols with 5 kinds of communication radius (10, 30, 50, 70 and 90 meters). We observe that the ACP and the BSW get a high query success ratio, while the MHLPP and the SLPD are lower than the former two protocols. It is obviously that the BSW is the highest one, because it is a no-privacy protocol. The ACP is just a little lower than BSW, because the query delivery process of the ACP is almost the same as that of BSW. Since users of ACP must wait for available ACs, they cost more time to initial their queries. But the ACP and the BSW are in the same level, comparing to the other two protocols. The MHLPP and the SLPD need to find friend to obfuscate their queries, which baffles their delivery process.

The experiment results when we test 20 minutes is shown in Fig. 9. Comparing to the average query success ratio at 10 minutes, the MHLPP and the SLPD achieve much higher success ratio after 20 minutes than at 10 minutes. That is because it cost them so much time in their obfuscation phases when they need to find friends. In fact, some of queries of the MHLPP and the SLPD still do not finish their obfuscation phase at 20 minutes.

The communication radius can influence the success ratio. In most of cases, the success ratio rises when we increase the communication radius, and its influence is especially obvious under 50 meters. A large communication radius makes it easily for users to encounter others, which is good for them to forward queries. However, a user who is so far away from the destination does not want the intermediates of his query encounters many users nearby. Because all users who carries copies of that query are near the sender instead of the destination, which decreases their query success ratio. Therefore, when the communication radius reaches 70 meters, the success ratios almost stay at the same level.

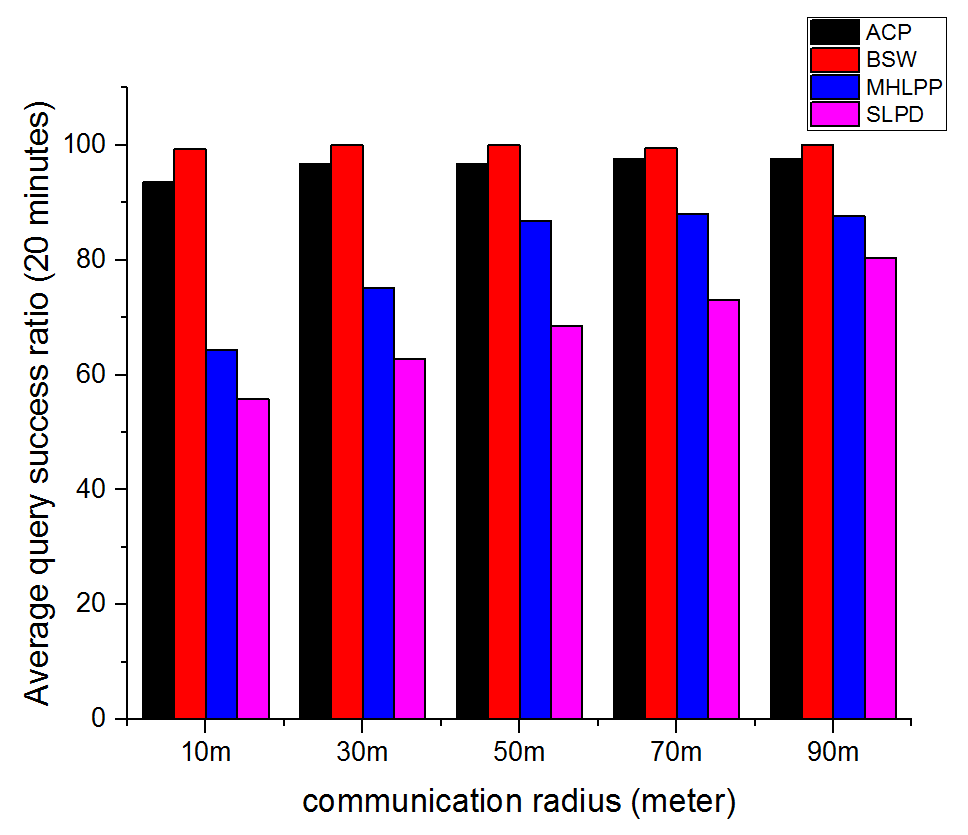


Fig. 9 average query success ratio (20 minutes)

## Average reply success ratio

When the LBSP receives a query, it sends a reply to the requester. If the reply arrives the original requester, we view it as success, otherwise, the reply is failed. Since there are 100 queries in each experiment, the number of replies should be equal to 100. Even though some of queries are not delivered to the LBSP quickly, they will be there if the test last for a long time.

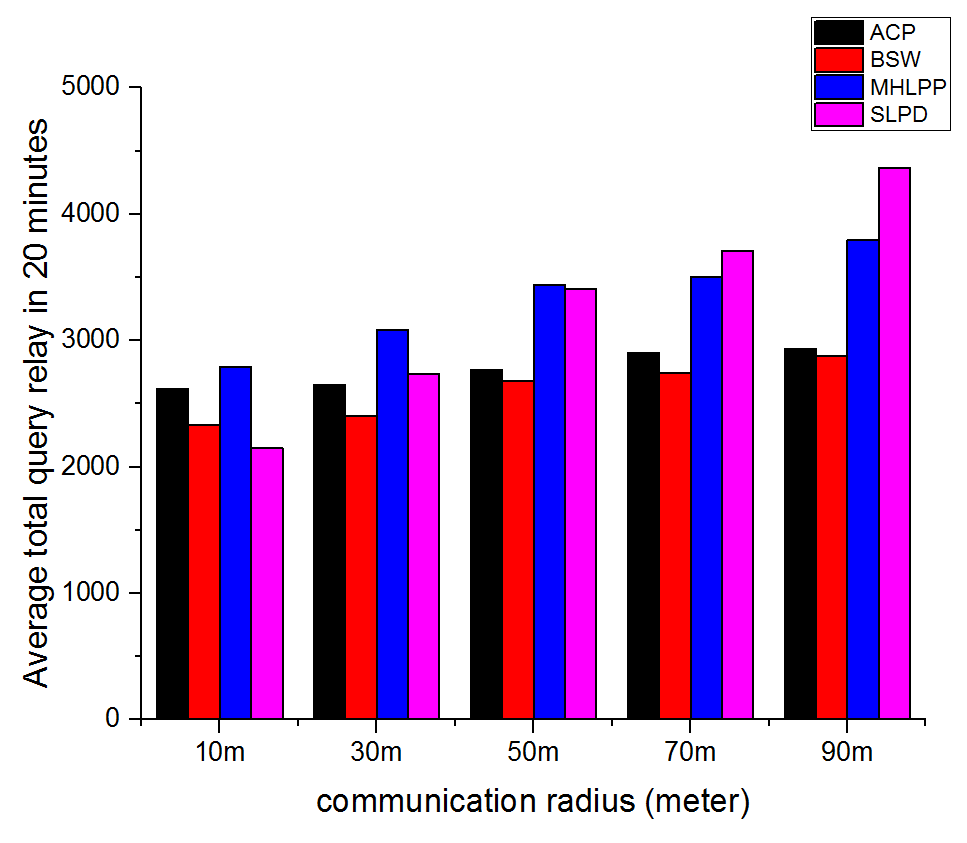


Fig. 11 average total number of forwarding queries at 20 minutes

In Fig. 11, the BSW has a significant and reasonable higher success ratio than all other protocols, because it is a no-privacy protocol. The ACP is higher than the MHLPP and the SLPD, but its advantage is not as large as that in the query process. In fact, the reply process of the MHLPP and the SLPD are simpler than that of the ACP, but the ACP saves so much time in its query process that it earns a better reply success ratio than the other two.

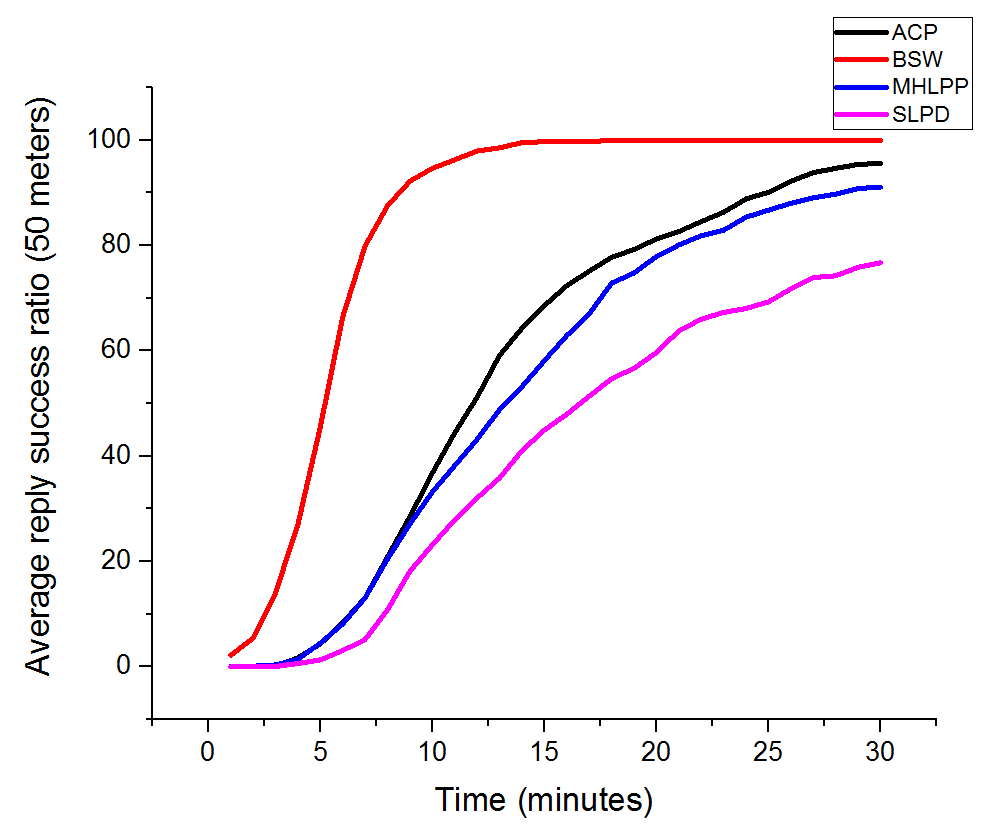


Fig. 10 average reply success ratio with 50-meters communication ratio

## Total number of query relay

The query delivery processes of all the four protocols use the BSW protocol. The BSW makes copies for queries and gives half of copies to any users it encounters. That is a significant cost for the network, so we count the number of forwarding queries to evaluate the cost of the four protocols. For example, in the SLPD, there are two phases: the obfuscation phase and the free phase. In the obfuscation phase, a query is forwarded among one-hop friends for times. After that, it is forwarded by the BSW protocol. The BSW protocol makes copies for the query and gives half of the copies to any encountering users. Then the number of forwarding queries should be about . If a user encounters the destination, he gives all its copies to the destination. When the number of his copies is larger than 1, the number of forwarding queries should be smaller than . The smaller that number is, the smaller cost of the network is.

In Fig. 11, we compare the total number of forwarding queries with four protocols. We observe that all the four protocols are at a similar level, the BSW and the ACP is a little lower than the other two. For the ACP and the BSW, they deliver queries so fast that users who carries more than one copies give all their copies to the destination at one time, as a result, many copies have no chance to be forwarded separately. While the MHLPP and the SLPD have obfuscation phases, the queries start to be delivered freely (in a BSW way) at a random place where might be so far away from the destination, so that almost all copies can be forwarded respectively.

The communication radius effects the total number of the forwarding queries, especially for the MHLPP and the SLPD. Those two protocols can finish their obfuscation phase more quickly which a larger communication radius, so that more queries can be forwarded freely (in the BSW way), which makes their total number of forwarding queries larger.

## Memory cost

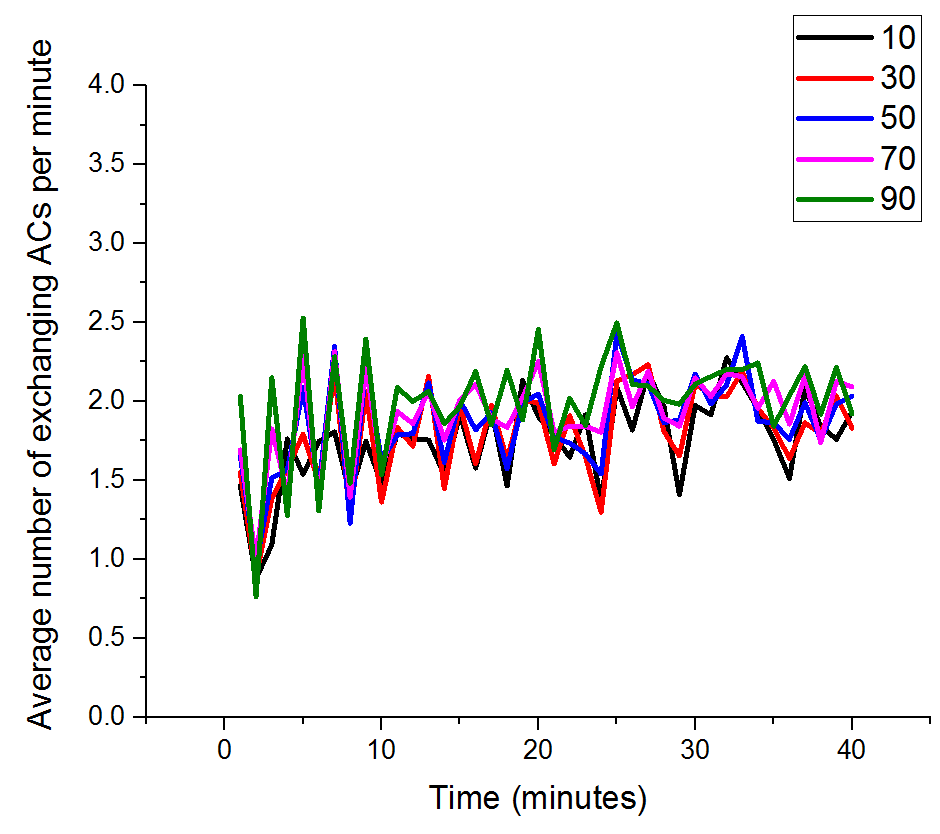


Fig. 13 Average number of exchanging ACs per minute

We count the number of queries carried by each user to evaluate the memory cost of the four protocols. Several copies of a query are counted for only once.

In the Fig. 12, we compare the average number of queries each user carrying with the four protocols at 20 minutes. We observe that the BSW is the highest in most of the cases and the ACP always stays at a similar level as the BSW. The data of other two protocols (the MHLPP and the SLPD) increase as the communication radius. The MHLPP even excesses the BSW when the communication radius is 90 meters. The reason is that quite a number of the BSW and the ACP users forward their copies to the destination so that there is no copy with them at 20 minutes, while the rest of them cannot forward their copies to the destination even given a large communication radius. While the number of the other two protocol’s free phase queries is significantly influenced by the communication radius. The more queries are in the free phase, the more copies are in the network.

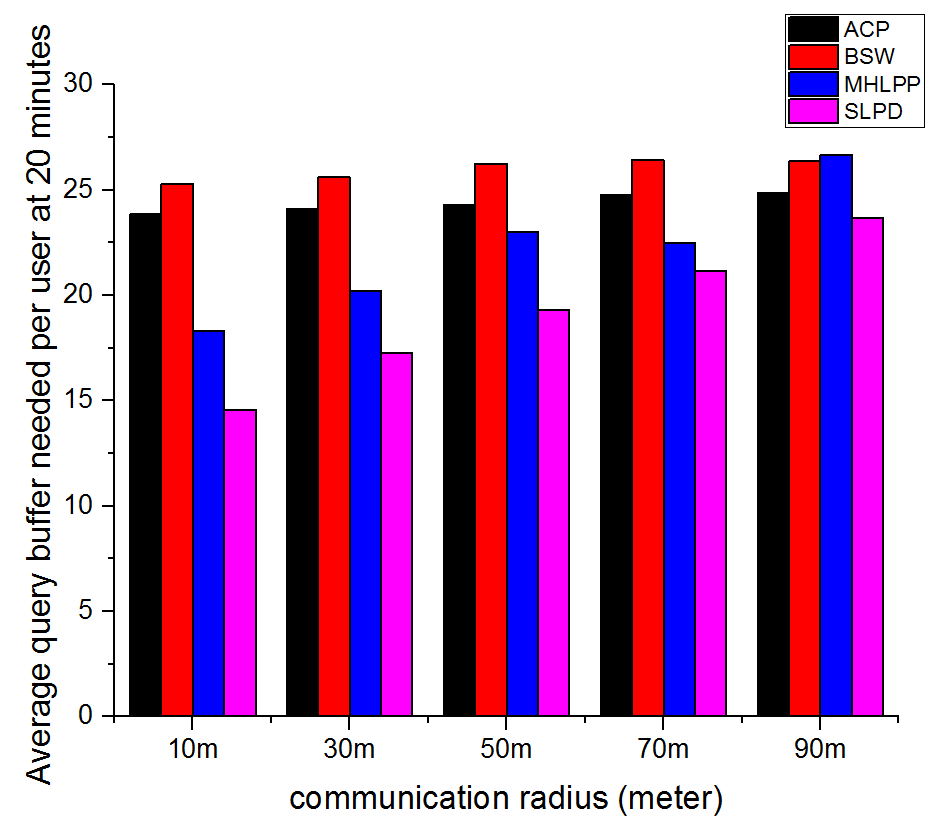


Fig. 12 average query buffer needed

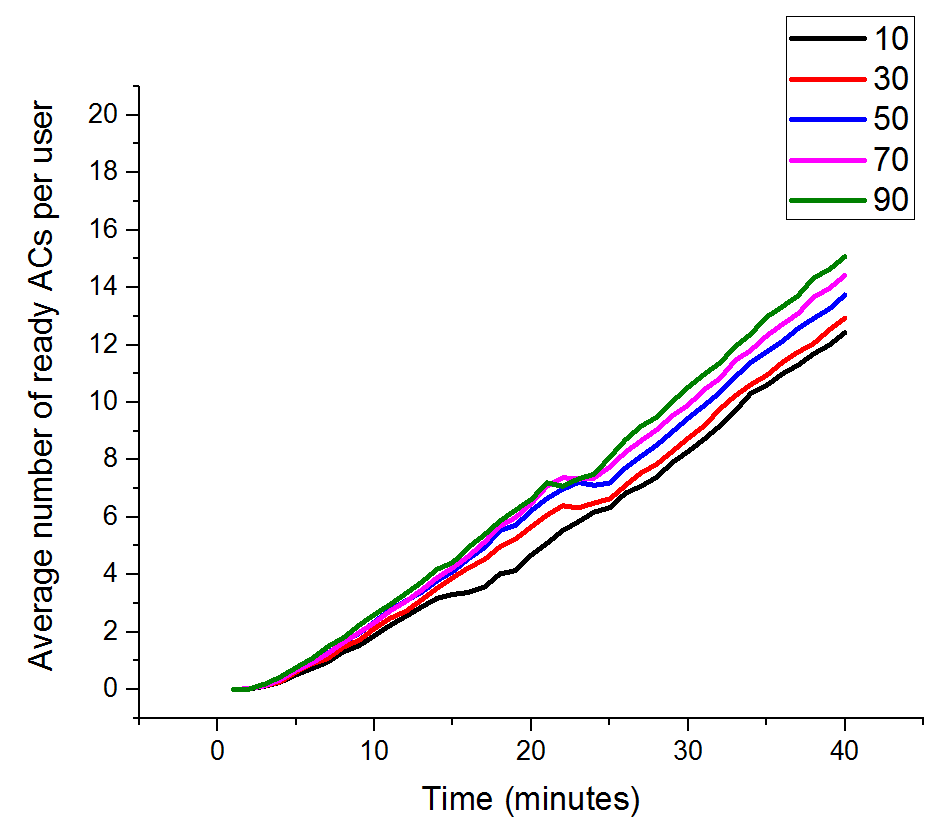


Fig. 14 Average number of ready ACs per user

## Distributing appointment cards

Exchanging ACs is a feature of the ACP, which imports burden into the network. We count the number of exchanging ACs per minute to evaluate the extra cost of the ACP.

In Fig. 13, we count the total of exchanging ACs processes in the whole network. For example, if a user Alice encounters another user Bob, the total of exchanging ACs processes increases by one when Alice exchanges any ACs to Bob. We count the number of those exchanging processes occur per minute. As shown in the figure, the exchanging processes do not occur frequently, but about 2 times per minutes. Since the size of an AC is a small, it does cost the network many resources. At the same time, users can get many ACs to help them send queries, as shown in Fig. 14. The number of ready ACs per user is raising smoothly and steadily.

# Conclusion

In this paper, we proposed a distributed location-privacy preserving protocol named ACP based on social-relationship and encryption. Simulation results show that it has a better performance on query-delivery success ratio and provides an acceptable obfuscation. The ACP costs more time when forwarding the reply, but the total time of the query delivery and the reply delivery is still shorter than the location-privacy comparison. The major disadvantage is that the ACP must exchange ACs continuously, which cost more network resources. But that cost is even low if users do not send so many queries.

##### References

1. K. Fall, "A Delay-Tolerant Network Architecture for Challenged," in Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications , pp 27-34, Karlsruhe, Germany, August 25 - 29, 2003
2. J. Schiller, A. Voisard, "Location-Based Services," Morgan Kaufmann , 2004
3. M. Gruteser, D. Grunwald, “Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking,” in Proceedings of the 1st international conference on Mobile systems, applications and services, pp 31-42, 2003
4. S. Zakhary, M. Radenkovic, "Utilizing social links for location privacy in opportunistic delay-tolerant networks," in ICC, 29 November 2012
5. S. Zakhary, M. Radenkovic, A. Benslimane, "The quest for location-privacy in opportunistic mobile social networks," in IWCMC, 2013 9th International
6. S. Zakhary, M. Radenkovic, A. Benslimane, "Efficient Location Privacy-Aware Forwarding in Opportunistic Mobile Networks," in IEEE Transactions on Vehicular Technology, Volume: 63, Issue: 2, Feb. 2014
7. E. Daly, M. Haahr, “Social Network Analysis for Routing in Disconnected Delay-Tolerant MANETs,” in Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing, 2007
8. M. Duckham, “Moving forward: Location privacy and location awareness,” in Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS, pp 1-3, 2010
9. T. Hashem, L. Kulik, “safeguarding location privacy in wireless ad-hoc networks,” in UbiComp, pp 372-390, 2007
10. L. Sweeney, "k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY," in Int. J. Unc. Fuzz. Knowl. Based Syst., 10, 557 (2002).
11. M. Mano, Y. Ishikawa, “Anonymizing User Location and Profile Information for Privacy-aware Mobile Services,” in Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Location Based Social Networks, pp 68-75, 2010
12. T. Wang, L. Liu, “Privacy-aware mobile services over road networks,” in Proceedings of the VLDB Endowment, pp1042-1053, Vol 2, August 2009
13. R. Lu, X. Lin, X. Shen, “SPRING: A Social-based Privacy-preserving Packet Forwarding Protocol for Vehicular Delay Tolerant Networks,” in INFOCOM, 2010 Proceedings IEEE
14. J. Zhang, C. Chow, “REAL: A Reciprocal Protocol for Location Privacy in Wireless Sensor Networks,” in IEEE Transactions on Dependable and Secure Computing, Vol 12, 2015
15. P. Hui, J. Crowcroft, “How Small Labels create Big Improvements,” in PERCOMW ’07. IEEE, pp 65–70, 2007
16. Diffie, W., Hellman, M., "New directions in cryptography," in IEEE Transactions on Information Theory, 1976
17. P. Jacquet, P. Muhlethaler, T. Clausen, “Optimized link state routing protocol for ad hoc networks,” in Multi Topic Conference, 2001
18. R. Xiang, J. Neville, M. Rogati, “Modeling Relationship Strength in Online Social Networks,” in Proceedings of the 19th international conference on World wide web, pp 981-990, April 26 - 30, 2010
19. T. Spyropoulos, K. Psounis, C. S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," in Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, pp 252-259, 2005