Received February 20, 2020, accepted March 1, 2020, date of publication March 5, 2020, date of current version March 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2978488

Achieving User-Defined Location Privacy Preservation Using a P2P System

SHENGCHAO LIU[1,2,3], JESSIE HUI WANG[1,3], JILONG WANG[1,3], (Member, IEEE), AND QIANLI ZHANG[1,3]

[1]Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China
[2]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[3]Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China

Corresponding author: Jessie Hui Wang (hwang@cernet.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2017YFB0503703.

ABSTRACT As location-based services become widely used in daily life, there is growing concern in preserving location privacy of users to avoid that attackers infer information about users by collecting and analyzing requests initiated by users. We argue that a good location privacy preservation scheme should have these properties. First, a user should never expose its precise location to any other entity. Second, a user should be able to specify its own requirement on the strength of privacy preservation, since a stricter preservation requirement may increase its overhead. Third, the scheme should be able to preserve as many as possible aspects of users' privacy under various attacks. With these desired properties in mind, we carefully design an *encoding scheme of users' identifiers* and a *fully distributed architecture* for our purpose and propose a privacy preservation scheme based on them. With the help of the encoding scheme and the distributed architecture, we develop a *distributed negotiation algorithm* to help users conduct negotiations among themselves to find their cloaked regions that satisfy their self-defined requirements without exposing their precise locations. The negotiations are completed without coordination from any central servers, and a random proxy is selected for each individual request, therefore the potential risks caused by any central server (location-based service servers or trusted-third-party servers) are mitigated as much as possible. Experiments show that our scheme can satisfy different strengths of privacy preservation required by each user even under the most severe scenarios.

INDEX TERMS k-anonymity, location-based service, location privacy, peer-to-peer.

I. INTRODUCTION

Nowadays, a lot of online devices have been equipped with positioning modules like GPS. The explosive growth of these devices and the increasing speed of mobile Internet have led to the rapid development of various Location Based Services (LBS) [1], such as Point of Interest (POI) searching and navigation. These location-based services bring great convenience to people's daily life and have attracted large amounts of users.

In order to enjoy a location-based service, users must submit their requests together with their locations to the LBS servers. Taking a user who wants to find out the answer of 'where is the nearest parking lot?' as an example. The user needs to assemble a request message that includes three elements (the *identity*, the *location* and the *query* (the question)). Here, the identity is used to receive the answer from the server, and it can be a virtual identity that could disclose the real identity, *e.g.*, the IP address. The location can be obtained by the user from positioning modules like GPS. The query reveals what information the user expects to get from the server with respect to the location.

During running, a LBS server can collect a lot of request messages. It can further analyze these messages to get private information about users such as personal addresses, routines, and health or economy conditions [2]–[4]. Even if the LBS server is trustworthy, it can be compromised by attackers. Then the attackers can have access to these data and they can analyze the data for private information about users. Of course users do not want their privacy exposed [5], so it is necessary to develop location privacy preservation schemes to solve or mitigate the risks mentioned above.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhe Xiao.

A location privacy preservation scheme should be able to preserve all aspects of users' privacy, *i.e.*, *identity*, *precise location* and *content of the query*. In recent years, there have been many location privacy preservation schemes proposed [5]–[10]. In some schemes, users run the proposed algorithms themselves to preserve their privacy. Specially, users can generate requests with coarse-grained locations (regions) or nearby locations instead of their precise accurate locations. Although the precise location can be preserved, the server of the location-based service is still be able to know the identity of the initiator of each request because of the direct communication between them. Besides the risk caused by direct communication, one user may not be able to know sufficient information to generate a proper coarse-grained region or nearby location. For example, if the coarse-grained region is too small and there is only the user itself in the region, an attacker who has known the user's location from other sources can associate requests using this region with the user, and the privacy of the user is exposed.

Then researchers propose to introduce Trusted Third Parties (TTPs) to act as proxies between users and LBS servers. In this type of schemes, users send requests to a TTP. The TTP would generate a proper region or location for each request, and then forward each request to the server. In this way, users do not need to communicate with the server directly and thus the risk of information leakage is mitigated. Furthermore, the TTP is collecting requests from a lot of users, and it is able to generate proper *cloaked regions* for users. A cloaked region is said to achieve *k-anonymity* [11] if there are at least k users in the region who are using this region in their requests. In this way, even the attacker knows the locations of users, it cannot map one request to the user initiating it because there are at least k users exploiting the same cloaked region.

However, just like LBS servers, TTPs are also in the risk of being compromised. Therefore, distributed P2P (Peer-to-Peer) schemes without TTPs are proposed. These mechanisms rely on the cooperation of users to preserve their privacy. Generally, users are organized into a P2P network where each user can find its nearby users to collaborate with. Early P2P schemes focus on the elimination of extra TTPs but assume that users should trust each other. For example, precise locations are exchanged between users in [12] to negotiate for cloaked regions. A cloaked region needs to cover all users that reach the agreement and would be used as their locations in their requests. Some P2P schemes [13], [14] achieve location privacy without exposing precise locations between users, however, like most existing P2P schemes, they apply to ad-hoc or opportunistic networks.

User peers in ad-hoc or opportunistic networks communicate with each other directly via radio signals or indirectly through relays among peers. However, the range of direct communication between two mobile devices is generally no more than 100m. With such a feature, schemes based on ad-hoc networks can deal with small densely populated areas well but are not suitable for large area scenes or sparsely populated areas. The networks are easy to split when peers are far apart, reducing possible collaborators among peers, even if in small areas.

Some schemes introduce caching mechanism into the P2P system to reduce communications with LBS servers, thus achieving location privacy and enhancing performance. However, the entities where data is cached often act as TTP-like entities. There are also some schemes, such as [15], [16], that include actual locations in final requests to LBS servers, which may expose user's privacy when the adversary has known some side-information of the target user.

With these problems in mind, we design a distributed P2P-based scheme with the following desired properties.

- First, we do not require users to expose their precise locations during negotiations. We design a mechanism for users to generate their identifiers and encode their coarse-grained locations in their identifiers.
- Second, we allow users to specify their own requirements on the strength of privacy preservation. The requirements are also encoded in their identifiers.
- Third, we design a *distributed* negotiation mechanism to determine cloaked regions for users based on their coarse-grained locations and personalized privacy requirements. We also design a P2P architecture to help users form an Internet-overlay network for negotiations no matter how far away these users are. We notice that current well-designed decentralized P2P architectures face several challenges when they are used for our purpose, therefore we make some important modifications based on one popular P2P architecture, *i.e.*, Kademlia, to make it work well in our problem.
- Fourth, for each request, the initiator uses a random node in the P2P network as a proxy to forward the request. In this way, direct communication with servers is avoided and it makes the system more resistant to attackers.

We implement our scheme and evaluate its performance under different experiment settings. In our experiments, the traces of moving users are generated using the Network-based Generator of Moving Objects proposed in [17] on the map of San Francisco Bay Area. We consider the extreme cases in which the whole P2P network has been cracked, and compare our scheme with CloakP2P [12] in terms of resistance to two typical attacks, *i.e.*, center-of-Cloaked-Region attack and correlation attack. The results show that our scheme can satisfy privacy preservation strength required by users even under attacks. We also look into the sizes of cloaked regions derived by our distributed negotiation algorithm. The cloaked regions of some users are larger in our system. It is the cost to realize stronger location privacy preservation. Fortunately, it only causes a little more communication overhead (the LBS server would return information about a larger area), which is not the main concern of users.

The rest of the paper is organized as follows. We review related work in Section II. In Section III, we describe the location privacy preservation problem and provides an overview of our solution. Details of our proposed scheme are presented in Section IV, including the design of identifier, the mechanism to connect all users, and the distributed negotiation algorithm. We present our experiment results in Section V and conclude this paper in Section VI.

II. RELATED WORK

Location privacy preservation has received lots of attention from researchers. Many location privacy preservation mechanisms have been proposed. There have been a few studies and surveys [5]–[9] which provide state-of-art reviews of these mechanisms. In this section, we briefly summarize solutions of three problems related to our work, *i.e.*, location preservation, identity preservation, and TTP-free architectures.

A. PRESERVING LOCATION INFORMATION IN REQUESTS

Obfuscation mechanism is widely used in privacy preservation systems to protect true and precise locations for location-based service requests. The goal can be achieved by replacing true and precise locations by dummy locations or coarse-grained areas, or by adding noises into true locations.

In [18]–[20], true locations are preserved by sending dummy locations together with the true locations in requests. In [18], dummy locations are randomly selected for each individual request. The works [19], [20] make dummy locations looking more realistic by considering trajectories of the user when generating dummy locations for multiple requests of a single user. In [21], the authors further consider the popularity of an area when using locations in the area as dummy locations, which makes it more difficult for attackers to find out dummy locations.

Replacing precise locations by coarse grained areas is used in [22], [23]. They generate a circular area to be used as the location of the initiator when sending a request.

Differential privacy is used in [24], [25] to preserve true locations of requests by adding noises into true locations.

These mechanisms are generally achieved by users themselves and requests are directly sent to servers, thus identity information can not be preserved.

B. PRESERVING IDENTITY INFORMATION FOR REQUESTS

We always want to decouple the identity of a user with the requests the user initiates to protect privacy of users from attacks. *Mix-zone* and *k-anonymity* are two commonly used types of anonymization mechanisms to preserve identify information in the area of location privacy preservation.

Mix-zone is first proposed in [2] and the user identity is preserved with pseudonyms in this mechanism. The pseudonym of a user is changing in areas called mix-zones with the help of TTPs. Therefore, it is hard to infer the user identity and associate requests with users for an observer (except for the TTP). In [26], the authors use this mechanism to achieve location privacy at social spots such as road intersections. In [27], a mix-zone framework is proposed to hide precise locations for mobile crowd sensing (MCS) applications. In [28], a dynamic mix-zone generation method is proposed to preserve location privacy in vehicular networks. MobiMix [29] is another mix-zone mechanism for road networks to improve effectiveness on location privacy preservation, including the resistance to transition attacks.

Mix-zones are generally predefined areas where users change their pseudonyms so that the observer cannot associate pseudonyms with users before they enter and after they leave the mix-zones. While in k-anonymity, cloaked regions can be constructed anywhere when necessary and the identity of a user must be k-anonymized, *i.e.*, indistinguishable with at least other $k-1$ users.

K-anonymity is originally used in databases to make a record not distinguishable with other $k-1$ records. In [30], k-anonymity is first introduced into location privacy preservation problem and since then various mechanisms achieving k-anonymity have been proposed such as [30]–[37].

Generally, k-anonymity is achieved by having at least k users exploit a common cloaked region in their requests. The key issue here is how to find k users who are willing to share a common cloaked region.

In [30], the locations of users are indexed into a quad-tree. The lower a quadrant is in the tree, the smaller the quadrant is. A user would traverse the quad-tree from the root until a quadrant covering less than $k-1$ users is reached. Then the parent of the quadrant is used as the cloaked region for the user. In this solution, k is a system-wide setting. Users can not specify their personalized requirements of k and the minimum size of their own cloaked region.

In [34], each user can have its own requirement on the minimum required k and the minimum area size of a cloaked region. The minimum quadrant (can be combined with brother quadrant) covering at least k users is returned for a user as its cloaked region. But not all of the users in a returned cloaked region employ this cloaked region because each user obtains its own minimum cloaked region, which does not conform to the definition of k-anonymity, *i.e.*, the actual number of users employing the returned cloaked region might be smaller than a user's required k.

There are many other mechanisms achieving k-anonymity. Many of the mechanisms (including mix-zones) rely on TTPs to operate. Typically, users submit their privacy requirements including the minimum required k and the minimum size of cloaked region together with their locations to a TTP so that the TTP can collect enough information to generate proper cloaked regions for users based on their locations and requirements. TTP is also used to forward requests to LBS servers without exposing user identities of the requests. However, TTPs may not be trust-worthy and they are also under the risk of attacks.

C. DISTRIBUTED P2P-BASED SCHEMES

Researchers have proposed some distributed P2P-based location privacy preservation schemes to remove the risk caused by TTPs.

Chow *et al.* first propose a P2P model in [12] which we refer to as *CloakP2P* hereafter. Users are self organized into an ad-hoc network as peers and collaborate with each other to generate cloaked regions for themselves. During collaborations, peers should trust each other and share actual locations with each other. When a peer wants to get its cloaked region, it needs to find other $k-1$ nearest peers and exploit the minimum region covering these k peers as its cloaked region to achieve k-anonymity. Clearly, in this scheme, the cloaked region is very small for each user and a user is of high possibility closest to the center of its cloaked region. The latter characteristic could be exploited by adversaries as described in [38]. CloakP2P also has the disadvantage of [34] as we described in Section II-B.

Since then various distributed P2P schemes have been proposed such as [38]–[42]. The major difference lies in the method of cloaked region generation and generated cloaked regions are generally bigger than CloakP2P. Within these schemes, Privé [39] and Mobihide [40] assign each user an index based on a Hilbert-space curve [43] and organize peers in a structured topology like Chord [44], enabling deployment on the Internet. The others organize peers in ad-hoc networks. However, these schemes assume that users are trustworthy and they can share sensitive information like true locations with each other to help generate cloaked regions.

Hu and Xu propose a scheme in [13] that can distributively calculate the cloaked region for peers without revealing precise location information to other peers. However, the proximity information used in cloaking is measured by peers through the received signal strength (RSS) or the time difference of arrival (TDOA) of beacon signals among peers themselves. The same method is used in positioning technologies, meaning accurate location information is still at risk of exposing. Another scheme named X-region [14] also assumes that users are not trustworthy and user peers generate their own cloaked regions through x-region exchanging wherein x-regions are coarse-grained areas where users reside. The two schemes also organize peers in ad-hoc networks.

There are also schemes based on Opportunistic Mobile Networks (OppMNets) such as [45]. Like ad-hoc networks, these schemes also require nodes to communicate with each other directly or via multi-hop route among peers, but messages are routed opportunistically which induces high latencies in end-to-end communications.

The scheme in [46] preserves users' location privacy under road-network. Although precise locations are not exposed, the scheme introduces a central server in each area and peers submit their information and requests to respective center servers in their own areas. P4QS [47] is another scheme with the same problem, in which a peer in an area acts as a TTP-like entity. Compromise of the TTP-like entities would expose user's privacy.

There are many schemes implementing caching mechanism to help protect privacy and improve user experience, such as [15], [48]–[52], and most of them cache data in peers. In [48], some special peers run as serving nodes who are responsible for caching all POI data for the region it is located in and serve neighboring peers' queries. Serving nodes are determined by a TTP-like entity and serving nodes themselves act like TTPs. In [49], some components are responsible for different tasks (including caching and anonymization) and they act like TTPs. The scheme in [15] preserves location privacy by having a request contains multiple actual locations of different users (including the request initiator). MobiCrowd [50] is a scheme in which a user can hide in the mobile crowd while using the service. In MobiCrowd, data is cached in peers. A peer first query nearby peers in their cached data and contact the server directly only when no answer from cached data. CAST [51] assume that mobile users are regular members of the area they visit. Peng *et al.* propose a scheme in [52] for continuous queries in which trajectory privacy is guaranteed by caching-aware collaboration between users. Most of these schemes organize peers in ad-hoc networks.

ABAKA [53] is a scheme that applies in ad-hoc or opportunistic networks and relies on random multi-hop forwarding to find collaborators for k-anonymity. ABAKA can guarantee p-sensitive k-anonymity in which at least p different values for each group of sensitive attributes are used. Sensitive attributes are specified by users themselves. Besides, ABAKA relies on Certification Authorities(CAs), which act as TTP-like entities, to authenticate users and assign them private keys for data encryption.

In [16], Rebollo-Monedero *et al.* propose a scheme on ad-hoc networks. The scheme mixes queries from many users and prevents providers from binding queries to users, *i.e.*, a request received by the LBS server contains multiple queries with each from a user. However, the scheme assumes that queries submitted to the LBS must be kept accurate, therefore user locations in the queries are accurate, no noises added or cloaked region employed. If an adversary has known the location of a target user and compromised the LBS server, the adversary can easily filter out the query of the target.

QDER [54] is a scheme that breaks the correlation between query and location. Each original query (location not included) is divided by the user into blocks. Then query blocks are exchanged many times with collaborative users and finally sent to the LBS server by users. From the server view, each user location corresponds to various queries, and each query comes from various locations. QDER also applies in ad-hoc networks.

In summary, we present Table 1 to show the differences between existing distributed P2P schemes and our scheme in terms of some important features of privacy preservation systems. We can observe that most schemes are applied in ad-hoc or opportunistic networks (i.e. Feature A). Users relies on the ad-hoc or opportunistic networks to communicate with each other via a single-hop (direct communication through radio signals) or a multi-hop route (among peers themselves). It is suitable for local densely populated areas but cannot deal with large area or sparsely populated areas. For example, if two groups of users are beyond the communication range and there is no available relay node, the two groups would form two ad-hoc networks and users in one group are unable to collaborate with users in the other group.

TABLE 1. A comparison of existing distributed P2P schemes with our scheme. (✓ indicates the schemes are different from ours on the corresponding feature)

Feature	A	B	C	D
[12], [38], [41], [42]	✓	✓		
[13], [14], [45], [50]–[52], [54]	✓			
[46], [48], [53]	✓			✓
[49]		✓		✓
[47]				✓
[15], [16]			✓	
[39], [40]		✓		

A: ad-hoc or opportunistic networks
B: peers trust each other
C: actual locations in final requests
D: TTP-like entities exist

On the other hand, the assumptions that peers trust each other (i.e. Feature B) and the existence of TTP-like entities (i.e. Feature D) violate the original intention of distributed P2P schemes, *i.e.*, there should be no trusted third parties.

Also, the exposure of actual locations in final requests (i.e. Feature C) may disclose location privacy when the adversary has enough side-information about a target.

In our scheme, we use coarse-grained locations to preserve location information and employ k-anonymity and random proxy mechanism to preserve identify information. We do not require users to expose precise their locations to each other, and the cloaked regions of users are negotiated through a totally distributed P2P system. Furthermore, our scheme can run over Internet and a user peer in our system can find as many collaborators as possible so long as there exist such peers in the vicinity and can have numerous proxy candidates all over the world. Please note that global deployment does not mean that cloaked regions generated by our scheme are necessarily large. We will illustrate the distributed negotiation algorithm for cloaked region generation in Section IV-B.

III. PROBLEM DESCRIPTION AND SOLUTION OVERVIEW

A. PROBLEM DESCRIPTION

Assume there are n users who want to achieve location privacy while enjoying location-based services. Let us denote these users by $\mathcal{U} = \{u_1, u_2, \ldots, u_n\}$. Each time when a user initiates a request, the user wants to replace its precise location by a cloaked region and achieves k-anonymity. In this way, malicious entities cannot infer the user's accurate location and identity information. The required strength of privacy preservation of a user u_i can be specified using two parameters, *i.e.*, the minimum size of the cloaked region (denoted by a_i) and the minimum number of users that are using the same cloaked region (denoted by k_i).

Given the precise locations l_i and its required strength of privacy preservation a_i and k_i for every user $u_i \in \mathcal{U}$, what we would like to do is to find a cloaked region r_i for each user u_i, whose size is the *minimum* among all regions that satisfy the three conditions, *i.e.*, $l_i \in r_i$, $\mathcal{S}(r_i) \geq a_i$ and $|\mathcal{G}(r_i)| \geq k_i$. Here, $l_i \in r_i$ means the cloaked region r_i should cover the user's current location l_i; $\mathcal{S}(r_i)$ denotes the size of the region r_i and $\mathcal{S}(r_i) \geq a_i$ means that r_i satisfies u_i's requirement for the preservation of location information; and $\mathcal{G}(r_i)$ denotes the set of users that use r_i as their common cloaked region, $|\mathcal{G}(r_i)|$ is the number of users in $\mathcal{G}(r_i)$, and $|\mathcal{G}(r_i)| \geq k_i$ means r_i satisfies u_i's requirement for the preservation of identity information.

Please note that r_i should be with the minimum size among all regions that satisfy the three conditions. Otherwise, we can just select a very large region as its cloaked region, which obviously cannot work well. There are at least two reasons. First, an unnecessarily large cloaked region would unnecessarily degrade the performance and experience of the services because the user would receive a lot of information from the server that are very far away from its current location. Second, it would also increase the communication overhead among users that share this cloaked region.

B. SOLUTION OVERVIEW

As we stated in previous sections, depending on a centralized server to collect information of all users, determine cloaked regions, and forward requests from users to LBS servers would bring potential risks. Since the server might know the historical locations and historical requests of each user u_i, it is possible for the server (or malicious entities who crack the server) to infer private information about u_i. Therefore, *it is preferred to design a distributed system for the problem.*

In a distributed system without any server, no one can know all requests of one user. Inferring information from partial requests is more difficult. And malicious entities have to monitor a lot of links (instead of a single server) to collect original requests, which is infeasible in the real world.

In order to develop a distributed system/algorithm to determine r_i for each u_i and preserve users' privacy during enjoying LBS, we should solve the following sub-problems.

1) *connecting all users*. Each user can communicate with any other users to collect privacy-preserved information when necessary;
2) *negotiating cloaked regions*. Each user u_i can determine its own cloaked region r_i via negotiating with proper neighbor nodes.

Once the above two sub-problems are solved, each u_i can get its cloaked region r_i to be used in its requests as its location. Furthermore, u_i should not communicate with the LBS servers directly. Otherwise, its identity might be exposed and the LBS servers (or attackers who crack the servers) might be able to infer information about initiators of requests. An intuitive way is to forward the request by a collaborative peer, however, the user peers should not be fully trusted. Therefore, we need to conduct the following step.

3) *secure message forwarding via random proxies*. A random proxy should be selected for each individual request of each u_i.

Now let us take u_i as an example and summarize the procedure of u_i to further illustrate the framework of our solution. Assume u_i is using some location-based service.

TABLE 2. A summary of preservation methods used in our scheme for different aspects of privacy.

Aspect	Preservation Method
Identity	k-anonymity and Random Proxy Mechanism
Location	Cloaked Region
Query	Encryption Mechanism

Conventionally, its privacy would not be preserved and the LBS server would know its identity or/and precise location. Now it decides to join our system to preserve its privacy. It needs to know at least one node that is already in the system and this node would help it become a participant of the system. After joining the system, u_i can exchange information with any user that is thought to be useful and negotiate with useful users to determine r_i. After determining r_i, u_i always replaces its precise location with r_i when initiating a request, and the request would be sent to a random chosen user (proxy). Note that the query content should be secured with encryption mechanisms such as asymmetric encryption so that the chosen proxy is unaware of query content and only the server can decrypt. The proxy would replace u_i's identity with the proxy's identity and then forward the modified request to the target of the request, *i.e.*, the LBS server. The server generates a response for this request, which is returned to the proxy, and the proxy forwards the response to u_i. Note that the response should be also secured with encryption mechanisms so that only the original request initiator can decrypt. We can see that the precise location, identity and query are preserved with our system.

Please note that all users can move continuously, and the cloaked regions should be updated accordingly. For example, assume u_i's cloaked region is α at time t, *i.e.*, $r_i^t = \alpha$. It now moves out of the region, which makes α invalid for it, and now it needs to negotiate with other users to get a new cloaked region. And all other users that are using the old cloaked region α, *i.e.*, $\forall u_j \in \mathcal{G}(\alpha)$, might be affected because u_j's requirement k_j can become unsatisfied due to u_i's leave.

Finally, we summarize preservation methods of different aspects of privacy used in our scheme in Table 2.

In the following section, we would introduce how we solve the three sub-problems mentioned above, *i.e.*, *connecting all users, negotiating cloaked regions*, and *message forwarding via random proxies*.

IV. THE SOLUTION TO ACHIEVE PRIVACY PRESERVATION

A. CONNECTING ALL USERS

Among the three tasks summarized in the previous section, connecting all users to enable the information exchange among them is very important because it is the prerequisite to negotiation. In order to reduce the risk of information leakage as much as possible, our basic idea is that users should help each other, and all information exchanges and negotiations are conducted completely in a distributed manner. It is exactly the idea of P2P (peer-to-peer) networks.

We know that there have been various P2P networks. Tracker-based P2P networks are not suitable for our problem because they still have a centralized component, *i.e.*, tracker. Any distributed P2P network that satisfies following requirements can be employed: a) it operates on the Internet so that global users can collaborate with each other across the Internet; b) it has an efficient routing algorithm so that a peer can find a target efficiently; c) it has been implemented and used in various practical applications for relatively long time so that its performance has been tested and guaranteed.

To our knowledge, Kademlia [55] is the most widely used type of trackerless decentralized P2P architecture and it has been implemented in most famous P2P applications, such as BitTorrent, BitComet, μTorrent and ed2k network (eDonkey and eMule). Since Kademlia has been applied and well tested in practice, especially in popular P2P applications, its performance has been guaranteed and proved. Thus we employ Kademlia in our own P2P scheme to connect users.

However, when being applied to location privacy preservation, Kademlia has to be modified to solve several challenges. Now let us briefly introduce Kademlia, discuss the challenges and present how we improve it to address these challenges.

1) KADEMLIA

Basically, in a decentralized P2P network, each peer node should have an *identifier*, which is a unique index to locate it, and a routing table is also required for a node to search for any other nodes and route messages to their destinations.

In Kademlia, each peer has a randomly generated 160-bit node ID. When Kademlia is used for file sharing, each file would also have a 160-bit identifier. The location of one file, *i.e.*, the nodes who actually store the file, is saved in the peer whose node ID has the closest distance from the file ID. The distance between two 160-bit identifiers is defined as their bitwise exclusive or XOR metric interpreted as an unsigned integer. For example, the distance between a node ID 1100_b and a file/node ID 1010_b is 6 (1100_b *XOR* $1010_b = 0110_b = 6$). Here, the subscript "b" means the number is binary.

Each peer node must have a routing table to store some other nodes (including the node ID, corresponding IP address and the UDP port running Kademlia) to make all nodes connected. During communications, the routing tables of involved nodes would be updated. A peer would add the source peer of a message into its table (or update its attributes if the source peer has been in the table). The routing table is with a limited storage space and there must be a mechanism to evict some entries when the table is full.

In Kademlia, the storage space is split into N buckets (N is the number of bits for node IDs), and a bucket stores at most ρ entries. The i-th bucket ($0 \le i < N$) of a peer p is only used to store peers whose distance to p is between 2^i (inclusive) and 2^{i+1} (exclusive). Each entry (which corresponds to a peer) has an attribute of "last seen time". When one bucket is full, a ping message is sent to the least-recently seen peer in the bucket. The node would be evicted from the bucket if it is unresponsive, and the new entry would be added into the bucket. Otherwise (the node responds successfully), the bucket would not be changed. We can see that the eviction mechanism prefers to keep old peers, which is based on an assumption that the node that has stayed in the system for a long time would stay for longer time than other nodes.

A new node can only join the P2P network via an existing node in the P2P network. Generally, some long-running nodes (referred to as boot nodes) are published together with the P2P applications or on websites. The existing node would help the new node construct an initial routing table by adding itself into the new node's routing table. Then the new node would populate its own routing table and add itself into other nodes' routing tables through communications with other nodes. The routing table would be updated continuously when the node is active in the P2P network.

When a peer with ID p_1 wants a file with ID f, it would conduct two steps:

1) p_1 would search for the peer that has the shortest distance to f.
   It uses f as the target identifier and starts an iterative search process. During iterations, Kademlia updates a list, whose size is ρ (a system-wide parameter), to save at most ρ closest nodes to f that have been found. Elements in the list is sorted in ascending order according to their distance to f. Initially, the list only has one element, which is p_1 itself. For each element in the list, if it has not been requested by p_1 in this searching process, p_1 would request the element to check its routing table and return at most ρ closest nodes to f in its routing table. When the element fails to respond, it would be removed from the list. Otherwise, for each returned node, there can be three possible cases, 1) the node has been in the list or has been requested, in which nothing is needed to do. 2) the node is not in the list and it is closer to f than the farthest node in the list, the node would be added into the list and the farthest node would be removed. 3) the node is not in the list and it is farther to f than all elements in the list, in which nothing is needed to do.
   The iteration process would stop once we find the target f or all elements in the list have been requested for checking. The target node with the same identifier with f or the closest node among all elements would be returned as the result of searching.
2) Let p^* be the closest node found from the step 1, and it knows the nodes (including their individual node IDs, IPs and ports) that actually store the file f. By contacting p^*, p_1 can learn these nodes and retrieve the file f from these nodes.

Since the identifiers of nodes and files are in the same format, searching a node is the same as the step 1 of searching a file, except that a file search can stop successfully at a node closest to the file but a node search successes only when the node is found.

Finally, a node leaves the P2P network silently in Kademlia without notifications to other nodes. It would be evicted from other nodes' routing tables as mentioned before.

2) CHALLENGES TO APPLY KADEMLIA IN LOCATION PRIVACY PRESERVATION

Kademlia can connect users and construct a P2P network for information exchange among them, but there are some challenges that have to be solved when we apply it to solve our problem.

- *The identifier of one node should contain information about its geographical location.*
  In original Kademlia, the distance between two identifiers does not reflect any physical meaning. People do not care about which nodes save the location of a file. They just define a way to save the file location and then they can find the location given the file identifier according to the definition. Therefore, Kademlia generates a random identifier for each node and the distance is just defined as XOR of two identifiers.
  However, in our privacy preservation system, one user needs to frequently search for nodes in an interested geographical area with unknown identifiers instead of a file with known identifier in Kademlia. Therefore, the identifier of a peer node should contain information about its geographical location and the distance between two identifiers should reflect the geographical distance among the corresponding two nodes. For example, assume an area is defined to be with an area prefix *pfx* and the identifier of any node within this area must encode this prefix. Then, when a node needs to negotiate with nodes in this area, it can start a search process targeting at a random identifier with the prefix *pfx*. The search process would return the closest nodes to the random identifier, which are likely to be in the area. Please note that the identifier of one node should not reveal the precise location of the node to avoid information leakage.
- *Each node should keep the real-time information of its nearby nodes accurate and quickly accessible.*
  There are two potential issues with the routing table in the original Kademlia. First, it needs a number of iterations to search for a node using its identifier, which is time-consuming in worse cases. Second, when a node leaves the P2P network, the entries related to this node in the routing tables of other nodes would not be updated immediately, which makes the routing table inaccurate. These two issues do not bring problems in traditional scenarios. But in our system, the leave of one neighboring node of p may cause p's current cloaked region invalid for p, which is an event that should be handled immediately. Therefore p must contact with neighboring nodes periodically using keep-alive messages and notify nearby nodes before leaving. Obviously, due to the frequent negotiation and keep-alive communication among nearby nodes, it would be preferred that the neighboring node can be found without iterative search processes.
  Therefore, we propose to design some small routing tables to save the entries for specific nearby nodes

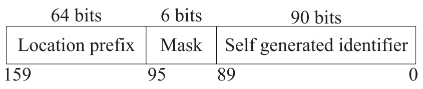


FIGURE 1. The structure of a node identifier.

(and other nodes that may be contacted frequently) as a complement to speed up the search process.

- *The eviction mechanism should evict the least-recently seen peer directly without ping test.*
  As we have explained, Kademlia evicts a node from its routing table only when the node is least-recently seen and is also unresponsive to the ping test. It is because Kademlia believes that the node that has stayed in the system for a long time would stay for longer time than other nodes.
  However, it is not the case in our scheme. As a node is moving, its identifier is also changing (note that the identifier is related to its geographical location). Although the node may be still active, it should not be saved in the current bucket any more (note that the bucket is determined by the distance between two nodes). We belive the newly seen node is more accurate. Therefore, we immediately evict the least-recently seen node without ping test and add the newly seen node into the routing table. The elimination of ping test also helps us reduce the communication overhead.

3) MODIFYING KADEMLIA TO ADDRESS CHALLENGES

In Section IV-A2, we summarize three challenges and propose schemes to solve them, among which the modification to address the third challenge is easy to implement and there is no need for further discussion. The proposals for the first two challenges, *i.e.*, the design of identifiers and the complementary routing tables, should be further discussed in this subsection.

- **The design of identifiers**

We design a hierarchical structure as presented in Fig. 1. Each identifier has a 160-bit length and it is composed of three segments, *i.e.*, *location prefix*, *mask* and *self generated identifier*.

The *location prefix* is used to encode the location of a peer and it locates in the highest bits as shown in Fig. 1. Although the location prefix has a 64-bit length, not all of the bits are used and the valid bits used for location prefix starts from the left, *i.e.*, the highest bits, and the invalid bits are just padded using zero. Therefore we need the segment *mask* to indicate the number of valid bits used in the location prefix of this identifier. In fact, we use the half of the valid length as the mask in this work. For example, if the mask is 8, it indicates that this identifier has $8 * 2 = 16$ valid bits in its location prefix segment. The prefix together with the mask identifies the geographical area of the node. The segment of *self generated identifier* is used to distinguish different nodes in this area. It can be generated randomly or via methods like uuid [56]. The uniqueness is ensured when a new node joins the P2P network.

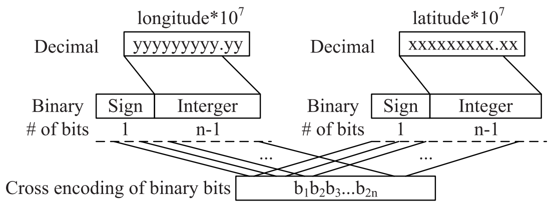


FIGURE 2. Binary cross-encoding of latitude and longitude in the location prefix segment.

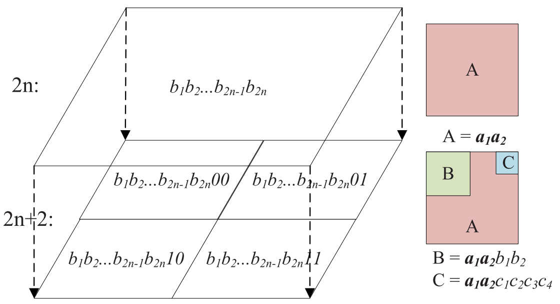


FIGURE 3. A parent area and its child (and successor) areas.

The identifier is used to uniquely locate a node and it is generated by the node itself based on its own location. Please note that the boot node is not necessarily located in the same area as the new node, *i.e.*, has the same location prefix and mask.

Our encoding scheme is inspired by the classless definition of the IP address space. An IP space is one-dimensional and it would be split into two halves when the length of prefix is increased by 1. The geographical area is with two dimensions, *i.e.*, latitude and longitude. We would like to split an area into four equal sub-areas when the prefix length is increased by 2, *i.e.*, one more bit for latitude and one more bit for longitude.

To ensure that the geographical distance can be reflected by the numeric difference of two identifiers, the left bits should always be more significant than the right bits in terms of presenting latitude and longitude of the node. We employ the cross-encoding method for binary latitude and longitude. As shown in Fig. 2, for a latitude or longitude, we first multiply the original decimal number with 10^7 and only reserve the integer part for encoding. The sign bit is set to be 1 only for south latitude and west longitude. Finally, the binary bits are cross-encoded into a binary string with first bit from longitude and second bit from latitude.

A longer location prefix indicates a smaller area, which means the node using this identifier reveals more precise information about its location. Fig. 3 shows that an area is split into four subareas when the mask is increased by 1, *i.e.* the valid length of prefix is increased from $2n$ to $2n + 2$. We would say the area indicated by the shorter prefix is the parent of the area indicated by the longer prefix. The four sub-areas share the same prefix bits with it parent area, but the lowest two bits are different for the four sub-areas.

Mask is designed for a user u_i to specify its desired strength of privacy preservation, and it can be viewed as the other way

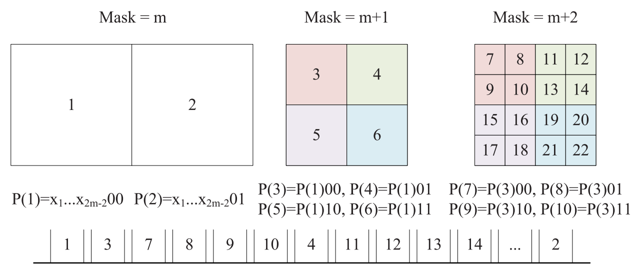


FIGURE 4. From two-dimensional geographical areas to one-dimensional location prefixes of identifiers. The upper portion shows areas in two-dimensional forms and the lower portion shows the relative encoding positions of these areas in encoding space in increasing order. $P(x)$ indicates the binary bits used for encoding area x. An area with mask of $m+1$ is a child area of one area with mask m. Similarly, an area with mask of $m+2$ is a child area of one area with mask of $m+1$.

to specify a_i, *i.e.*, the required minimum size of its cloaked region. For example, a user with intense privacy concern can use the prefix of the parent area (shorter prefix) to generate its identifier, while a user with light privacy concern can be viewed as in the sub-area (longer prefix) instead of the parent area when generating its identifier. The location prefix and mask of one identifier always specify an area. When a user generates its identifier using the location prefix and mask of an area α, in fact it means the user desires the area α as its preferred cloaked region. Here, the location prefix is determined by the user's location (since the area must cover its location), and the mask indicates its required strength of privacy preservation.

Let us denote the area specified by u_i's identifier as dr_i, which is the desired cloaked region of u_i. We further define u_i's *local users* as the users whose identifiers have the same location prefix and mask as u_i (so they are viewed as in the same area and we say they are local users). Obviously, the local users of u_i share the same desired cloaked region dr_i. Let us denote the set of users that desire an area α as the set $\mathcal{M}(\alpha)$. Obviously, the set of u_i's local users is $\mathcal{M}(dr_i)$.

A user u_i first negotiates with its local users. If an agreement can be reached among sufficient number of local users (their k_i are all satisfied), these users can use dr_i as their common cloaked region. Some users may fail in the local negotiation because they require larger k_i. These users have to participate in the local negotiation of its parent area. We use α^j to denote the jth level ancestor area of α and use $mask(\alpha)$ to denote the mask of the area α. Particularly, dr_i^j is the jth level ancestor area of dr_i. Clearly, dr_i^j and dr_i share the same highest $2 * mask(dr_i^j)$ bits in their location prefix segments and there should be $mask(dr_i) - mask(dr_i^j) = j$.

The right part of Fig. 3 illustrates an example in which one area is split multiple times, *i.e.*, the mask is increased by more than 1. We can see that the prefixes of two close areas B and C share the same high bits (a_1a_2) which means that the distance between prefixes of close areas is relatively small.

Fig. 4 further shows the location prefixes of parent areas and child areas. The bottom line shows the area indexes in the increasing order of their prefixes. We can see that geographically close areas roughly have small distance between their prefixes.

TABLE 3. Areas with different number of binary bits used in location encoding near the equator.

Bits (2*mask)	Range on latitude (or longitude)	Area size
2*9	93.38 km	8719 km^2
2*10	46.69 km	2179 km^2
2*11	23.35 km	545 km^2
2*13	5.84 km	34 km^2
2*15	1.46 km	2.13 km^2
2*16	729.53 m	0.53 km^2

TABLE 4. Three routing tables of u_i and the entries in them.

Table	Entries in the Routing Table
RtB	basic routing entries designed in Kademlia.
RtA	entries for local users that desires dr_i as the cloaked region.
RtU	entries for each ancestor area of dr_i.

In summary, our design of identifiers enables three necessary features: 1) The identifier contains information about the node's geographical location; 2) the distance between identifiers of two nodes approximately reflects their geographical distance; 3) users can specify their privacy preservation requirements in their identifiers.

Table 3 presents the size of one area as the mask is increased from 9 to 16. We can see that the size of one area with a mask of 16 is 0.53 km^2, which has been relatively small to specify a user's location. Therefore, the number of bits in the prefix segment is sufficient.

- **The design of routing tables**

As we have introduced in the previous subsection, the basic Kademlia routing table *RtB* cannot serve our purpose well. We plan to design two more routing tables, *RtA* and *RtU*, to complement the basic routing table *RtB*.

In our system, each user should keep the real-time information of its local users accurate and quickly accessible because it needs the accurate information frequently. In order to achieve this goal, we design a new routing table to save the real-time accurate information about local users, which is referred to as *RtA*.

The user u_i requires that its cloaked region should be shared by at least k_i users to preserve its identity. Our design is that its RtA has exactly $2 * k_i$ entry slots, wherein k_i slots are for left local user nodes and k_i slots are for right local user nodes. Here, "left" and "right" are defined based on their locations in the sorted list in terms of their identifiers. Later in the proof of the Argument in Section IV-B2, we will show the reason for this design.

Each entry is for a local user node of u_i. Remind that "local user" of u_i is a node with the same desired cloaked region dr_i (*i.e.*, with the same location prefix and mask). The entry in RtA saves the required k_i of the corresponding user.

Obviously, there might be empty entries in RtA if the number of left users or right users sharing dr_i is smaller than k_i. If the number of users sharing dr_i are larger

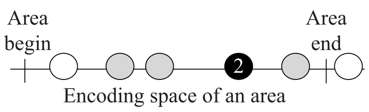


FIGURE 5. An example to show RtA of the black node whose required k is 2.

than $2 * k_i$, each node only has partial information of other nodes sharing dr_i.

Fig. 5 shows an example to illustrate RtA. Among all users that have the same dr_i with the black node, there are three nodes whose identifiers are smaller and one node whose identifier is larger than the black node. Since the required k is 2 for the black node, two left nodes and one right nodes are saved in the RtA of the black node.

When a new node is learned by u_i, it would be checked to see if the node should be added into RtA. If it is for a local user node of u_i and it is close enough to u_i (closer than one current entry in RtA or there is usable slot in RtA), it would be saved in RtA. Besides that, u_i would check every node in its RtA periodically using keeping-alive messages for two purposes. First, the messages are used to check the status of its local users in RtA. If the node is unresponsive or has moved out of dr_i, the node would be removed from u_i's RtA and a new closest node would move in if there exists one. Second, the messages are used to learn more local users. u_i would send the information about at most ρ left nodes and at most ρ right nodes in u_i's RtA in the keeping-alive messages, and then local users are exchanging information to learn each other as much as and as soon as possible.

Note that when a node leaves our system, the node should actively notify nodes in its RtA so that these local user nodes can update their routing tables immediately and try to find new cloaked regions as the reaction of the node leaving.

In case that u_i cannot find sufficient users in its desired cloaked region, *i.e.*, dr_i, it needs to negotiate with nodes in the parent area of dr_i. If the negotiation with nodes in the parent area also fails, the grandparent area would be checked and so on. Therefore, u_i should have at least one accurate pointer (entry) for each ancestor area to accelerate the process. We save the information about every ancestor area in the new routing table RtU as follows.

Assume that the mask in the identifier of u_i is $mask(u_i)$, then it has $mask(u_i) - 1$ ancestor areas. RtU is designed to have $mask(u_i) - 1$ buckets. Each bucket is working for one ancestor area and the maximum number of entries in one bucket is a system parameter that is not smaller than 1. u_i would send keeping-alive messages periodically to a random user in each individual bucket of RtU. Upon receiving a keep-alive message from a user in its successor area, the node sends at most ρ left nodes and at most ρ right nodes in its RtA in the response message, so that u_i's bucket of RtU can be quickly filled.

Initially, u_i's RtU is constructed by initiating a searching process to a randomly generated identifier in the corresponding ancestor area. The searching process would return at most ρ closest users to the random identifier as we described in Section IV-A1, among which some returned nodes in the ancestor area are saved in the bucket. A larger system parameter may reduce the number of initiated search processes, since the search is conducted only when all entries are invalid and removed, *i.e.*, the bucket is empty.

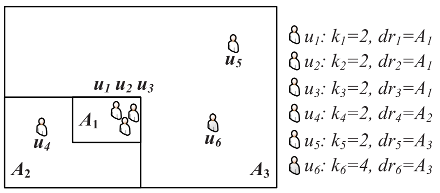

u_5
u_1 u_2 u_3
A_1
u_4
u_6
A_2
A_3
u_1: k_1=2, dr_1=A_1
u_2: k_2=2, dr_2=A_1
u_3: k_3=2, dr_3=A_1
u_4: k_4=2, dr_4=A_2
u_5: k_5=2, dr_5=A_3
u_6: k_6=4, dr_6=A_3

FIGURE 6. An example to illustrate the negotiation procedure.

B. NEGOTIATING CLOAKED REGIONS

With mechanisms in Section IV-A, all users have been connected and each individual node u_i knows some users in $\mathcal{M}(dr_i)$ (local users) through RtA and at least one user, if exists, in the set $\mathcal{M}(dr_i^j)$ for each of its ancestor area dr_i^j through RtU.

Theoretically, all local users of u_i can easily reach an agreement to use dr_i as their cloaked region if the following equation holds,

$$k_j \leq |\mathcal{M}(dr_i)| \qquad \forall j \in \mathcal{M}(dr_i). \tag{1}$$

Otherwise, some users with larger requirements on k will fail and have to find its partners, *i.e.*, the users that share the same cloaked region with it, from its ancestor areas. The final cloaked region of each of these users would be larger than its desired region.

In the following subsections, we would first illustrate the negotiation results using an example. Then we describe our distributed negotiation algorithm. Please note that the negotiation algorithm must be distributed to mitigate the risk of information leakage.

1) EXAMPLE AND FRAMEWORK OF NEGOTIATION

Let us illustrate the negotiation results using Fig. 6. There are six users in total in the system, and their information are listed at the right part of the figure, including their k_i and dr_i.

We can see that u_1, u_2 and u_3 share the same desired cloaked region A_1, and they satisfy the Equation 1. They can reach an agreement to use A_1 as their common cloaked region, *i.e.*, $r_1 = r_2 = r_3 = A_1$.

The user u_4 cannot find any local user that desires A_2 and $k_4 = 2$, which means it must contact its parent area A_3 for more nodes. Hopefully it can find u_5. These two users u_4 and u_5 can reach an agreement to use A_3 as their common cloaked region. u_5's privacy preservation requirement is satisfied perfectly, but u_4 is forced to use a larger cloaked region than it desires. It would incur more communication overhead for u_4 but does not increase its risk of information leakage.

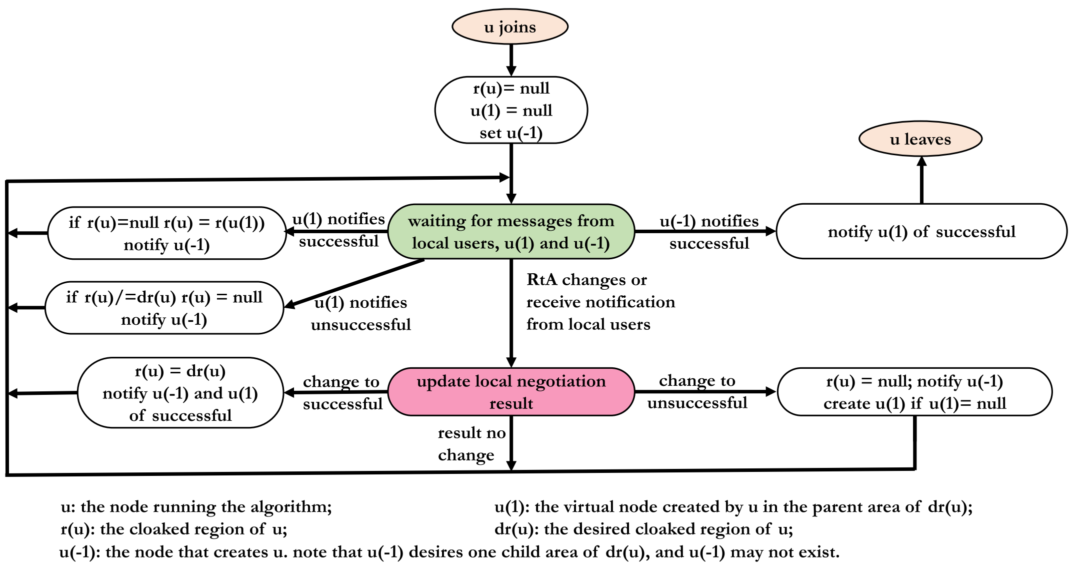

u joins
r(u)= null
u(1) = null
set u(-1)
u leaves
if r(u)=null r(u) = r(u(1))
notify u(-1)
u(1) notifies
successful
waiting for messages from
local users, u(1) and u(-1)
u(-1) notifies
successful
notify u(1) of successful
if r(u)/=dr(u) r(u) = null
notify u(-1)
u(1) notifies
unsuccessful
RtA changes or
receive notification
from local users
r(u) = dr(u)
notify u(-1) and u(1)
of successful
change to
successful
update local negotiation
result
change to
unsuccessful
r(u) = null; notify u(-1)
create u(1) if u(1)= null
result no
change
u: the node running the algorithm;
u(1): the virtual node created by u in the parent area of dr(u);
r(u): the cloaked region of u;
dr(u): the desired cloaked region of u;
u(-1): the node that creates u. note that u(-1) desires one child area of dr(u), and u(-1) may not exist.

FIGURE 7. **The distributed algorithm run by a node *u* to determine its cloaked region.**

The user u_6 has a larger k_6. Although it also desires A_3, but the total number of users that are willing to use A_3 is 3, which is less than k_6. Therefore, u_6's privacy preservation requirement cannot be satisfied.

We would like to emphasize the following points from this example.

1) Nodes are selfish.
In the above example, if one of u_1, u_2 and u_3 is sacrificed and agrees to exploit A_3 as its cloaked region, all users' requirements can be satisfied, *e.g.*, $r_1 = r_2 = A_1$ and $r_3 = r_4 = r_5 = r_6 = A_3$. However, the sacrificed user would have a larger cloaked region than it desires although its privacy preservation requirement can be satisfied. In a distributed algorithm, it is reasonable to assume that no user would like to sacrifice itself. Therefore, u_1, u_2 and u_3 would reach the agreement on A_1.
2) Nodes cannot be always satisfied.
If a user demands a very large k, its negotiation might fail and the system would notify the user. The user should decide to change its privacy requirement or give up using location-based services until it obtains a cloaked region.

2) DISTRIBUTED NEGOTIATION ALGORITHM

Our basic idea is as follows. A user u_i always participates in the *local negotiation*, which is to negotiate with u_i's local users. If u_i can reach an agreement with some local users to exploit dr_i as their common cloaked region, u_i successfully solves its problem and exploit $r_i = dr_i$. We say its status is successful in local negotiation.

Otherwise, if u_i fails in its local negotiation, it would create a virtual user node u_i^1 to participate in the negotiation among the nodes that desire the parent area of dr_i, *i.e.*, dr_i^1. In order to participate in the negotiation of the parent area, the virtual node should be created with an identifier that specifies the parent area as its desired cloaked region. In other words, the identifier of the virtual node should have the same highest $2*(mask(u_i)-1)$ bits in the location prefix segment as u_i, and its mask should be set as $mask(u_i)-1$.

The virtual node u_i^1 runs the same algorithm as real users. If it still cannot reach an agreement with sufficient number of local users of dr_i^1, the virtual node u_i^1 would continue to create a virtual node that desires the parent area of dr_i^1, *i.e.* dr_i^2, and participate in the negotiation of dr_i^2. This recursive procedure is executed until an agreement is reached in the local negotiation or no more ancestor area can be used to create a virtual node. Once an agreement is reached, say u_i^j reaches an agreement with local users of dr_i^j, the virtual node u_i^j would notify its creator u_i^{j-1} to use dr_i^j as its cloaked region. The notification would finally received by u_i, and u_i then gets $r_i = dr_i^j$. If the recursion stops because there is no more ancestor area, it means that u_i cannot get a cloaked region to satisfy its privacy requirements.

As a dynamic system, users may leave the system and new users may join the system, which may change the result of local negotiation. Once the result of local negotiation changes from "unsuccessful" to "successful" for u_i^j ($j \geq 0$ and u_i^0 means u_i), it needs to check whether it has created a virtual node u_i^{j+1} in its parent area. If yes, the node needs to notify the virtual node to leave the P2P network. In other words, the virtual node u_i^{j+1} should be destructed.

Before a user node is destructed or a user node leaves the system, it also needs to check whether it has created any virtual node. If it has a virtual node, the virtual node should also be removed from the P2P network.

Taking an arbitrary node u as example, we describe the distributed algorithm a node runs in Fig. 7. Please note that all nodes should run the same algorithm no matter the node is a virtual node or a real user.

3) THE LOCAL NEGOTIATION ALGORITHM

Now the only remaining problem is how the local negotiation is conducted in a distributed manner. Let us first prove the following argument.

Argument: For an area α, among all local users $u_i \in \mathcal{M}(\alpha)$, the user that requires the largest k can determine whether it can use α as its cloaked region only from its RtA, i.e.*, without extra information exchange with any other users.*

Proof: Without loss of generality, let us assume u_1 is the node that requires the largest k among all users in $\mathcal{M}(\alpha)$, *i.e.*, $k_1 \geq k_i(\forall i > 1)$.

Let $\mathcal{N}(u_i)$ denote the set of user nodes in u_i's RtA (RtA_i) whose k are no larger than k_i. In other words, we define that $\mathcal{N}(u_i) \triangleq \{u_j | k_j \leq k_i \text{ and } u_j \in RtA_i\}$. Obviously, each user node u_i can learn its own $\mathcal{N}(u_i)$ by simply checking the entries in its own RtA. Particularly, since $k_1 \geq k_i(\forall i > 1)$, $|\mathcal{N}(u_1)|$ is actually the number of all nodes in RtA_1.

If $|\mathcal{N}(u_1)| \geq (k_1 - 1)$, u_1 surely can draw a conclusion that it can use α as its cloaked region, *i.e.*, α can be used as the cloaked region for all users saved in RtA_1.

Otherwise, we have $|\mathcal{N}(u_1)| < (k_1 - 1)$. Remind that u_1 has k_1 slots for left users and also k_1 slots for right users. $|\mathcal{N}(u_1)| < (k_1 - 1)$ indicates that RtA_1 has empty slots for left local users and also has empty slot for right local users. It means u_1 knows clearly that it has saved all other local users in $\mathcal{M}(\alpha)$ [1] and the total number of local users of α is less than $k_1 - 1$. Therefore, by looking at its RtA, u_1 easily finds out it cannot use α as its cloaked region no matter what decisions other local users make. ■

According to the above argument, at least u_1 can determine whether it can use α, *i.e.*, whether the result it gets in local negotiation is successful or not. In case that u_1 is successful, all local users are successful because they require smaller k than u_1. We can ask u_1 to notify them that they are all successful. In case that u_1 is unsuccessful, it knows it has the information about all local users, therefore it is able to find out the negotiation result for each node (successful or not).

The problem is that a user does not know whether it is u_1, *i.e.*, whether it has the largest k among all local users. It can be addressed by asking every user to run the following distributed local negotiation algorithm.

Each user u_i keeps checking its local RtA. There are three possible cases.

1) if u_i finds that $|\mathcal{N}(u_i)| \geq (k_i - 1)$.
   It means u_i has found sufficient number (*i.e.*, $k_i - 1$) of partners to use dr_i as their common cloaked region. Thus u_i can determine its local negotiation result is "successful". It also notifies its partners (users in $\mathcal{N}(u_i)$) that they are "successful" in the local negotiation. In other words, they are assumed to use the desired cloaked region.[2]
2) Otherwise if RtA_i has empty slots for both left local users and right local users.
   In this case, u_i is assumed to know all local users and it can determine it is unsuccessful in the local negotiation about dr_i, *i.e.*, dr_i cannot be used as r_i. It is also responsible to determine the local negotiation results for other users in its RtA. The way it takes is as follows. For each user u_j in RtA_i, u_i calculates the number of users in RtA_i that are with no greater k than k_j, *i.e.*, $n_{ij} = |\{u_t | u_t \in RtA_i \text{ and } k_t \leq k_j\}|$. If $n_{ij} \geq k_j$, it means all these users in the set $\{u_t\}$ can share the cloaked region and their required k_t can be satisfied. u_i would notify all users in $\{u_t\}$ of the local negotiation result "successful".
3) Otherwise, u_i will wait for notifications from other local users. Furthermore, once it receives a notification of "successful", it would further propagate the good news to users in $\mathcal{N}(u_i)$, *i.e.*, its local users in RtA_i and be with smaller k than k_i.

According to our Argument at the beginning of this subsection, at least one node, *i.e.*, the node with largest k in this local negotiation, can determine its own local negotiation result. In other words, at least one node finds that it either satisfies the first condition or the second condition. Therefore, it can determine its own result and also starts to notify other users. The result would be propagated to all local users.

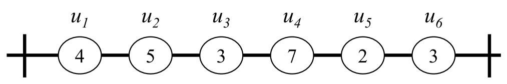
In the above procedure, a user only notifies other users of good news. Every node is set to be "unsuccessful" by default initially. Since the negotiation result may change due to the leaving of users, each node should remember the notifications it sends out. When the node leaves, it should withdraw the notifications it has sent out.

A user may receive notifications from multiple users. It does not matter because all notifications are asking it to use the same area as its cloaked region. But it is possible that one node can determine its local negotiation result and the result is inconsistent with notifications. This phenomenon occurs when some local users are in the procedure of information exchange and have not updated the view of the system. The node would trust itself and discard the received notifications. The results would be consistent later, which is similar to the converge of routing protocols.

We illustrate the local negotiation algorithm using the example shown in Fig. 8. There are six users in an area and the required k of each user is labeled on the circle that represents the user. This area is the desired cloaked region of these users. These users have been sorted according to their identifiers as shown by the line in the top of the figure. In order to make the example easy to understand, we list the RtA table of each node. We can see that u_2, u_3 and u_6 can find they satisfy the

[1]If u_1 has not learnt all local users via information exchange, it would temporally conclude that α cannot be used as its cloaked region. But it can draw a correct conclusion after it exchanges information fully with other local users. Due to the existence of RtA, the information exchange process would not take long time.

[2]Please note all users in the local negotiation have the same desired cloaked region.



$k_1 = 4$, RtA$_1$: left $= \{\_, \_, \_, \_\}$, right $= \{u_2, \boldsymbol{u_3}, u_4, \boldsymbol{u_5}\}$
$\mathcal{N}(u_1) = \{u_3, u_5\}$

$k_2 = 5$, RtA$_2$: left $= \{\_, \_, \_, \_, \boldsymbol{u_1}\}$, right $= \{\boldsymbol{u_3}, u_4, \boldsymbol{u_5}, \boldsymbol{u_6}, \_\}$
$\mathcal{N}(u_2) = \{u_1, u_3, u_5, u_6\}$

$k_3 = 3$, RtA$_3$: left $= \{\_, u_1, u_2\}$, right $= \{u_4, \boldsymbol{u_5}, \boldsymbol{u_6}\}$
$\mathcal{N}(u_3) = \{u_5, u_6\}$

$k_4 = 7$, RtA$_4$: left $= \{\_, \_, \_, \_, \boldsymbol{u_1}, \boldsymbol{u_2}, \boldsymbol{u_3}\}$, right $= \{\boldsymbol{u_5}, \boldsymbol{u_6}, \_, \_, \_, \_, \_\}$
$\mathcal{N}(u_4) = \{u_1, u_2, u_3, u_5, u_6\}$

$k_5 = 2$, RtA$_5$: left $= \{u_3, u_4\}$, right $= \{u_6, \_\}$
$\mathcal{N}(u_5) = \{\}$

$k_6 = 3$, RtA$_6$: left $= \{\boldsymbol{u_3}, u_4, \boldsymbol{u_5}\}$, right $= \{\_, \_, \_\}$
$\mathcal{N}(u_6) = \{u_3, u_5\}$

FIGURE 8. **An example to illustrate the local negotiation algorithm.**

first condition and therefore they know they are "successful" in this local negotiation, *i.e.*, they can use this area as their cloaked region. According to our algorithm, u_2 would notify the result of "successful" to all nodes in $\mathcal{N}(u_2)$, *i.e.*, u_1, u_3, u_5 and u_6. u_4 can find that it satisfies the second condition. Therefore u_4 determines it is "unsuccessful" in this local negotiation, and it is responsible to determine the results of all nodes in its RtA and notify them. As a result, u_1 and u_5 can also receive a notification of "successful" from u_4. In this way, all nodes can determine their results of local negotiation. Please note here we do not present all notifications between nodes for the sake of simplicity.

C. RANDOM PROXY

Now each node u_i has known its cloaked region. When u_i wants to use a LBS, it can assemble a request message $m = \{(r_i, query, P_m)\}$, wherein r_i is the cloaked region of u_i, *query* is the question u_i wants to know with regard to r_i, and P_m is the key u_i generates randomly for the server to encrypt the response message. The request message m would be encrypted using the public key of the LBS server.

For each request message, a random proxy is selected to forward the message from its initiator to the LBS server. In this way, we avoid the direct communication between the request initiator and the service server, so that the LBS server cannot infer the information about the message initiator easily. Further, since each message is forwarded by a random (thus different across messages) proxy, no user can collect all messages of an initiator, which reduces the risk caused by information inference using some techniques such as correlation attack. If an attacker wants to collect many requests to conduct attacks, it has to crack the P2P system and collect messages from a lot of vantage points. We will analyze the extreme cases later in the section of experiments.

The steps of message forwarding via a random proxy are as follows.

1) u_i generates a random identifier f.
2) u_i searches for the target peer with the identifier f.
3) If the target peer is found, the target peer is chosen as the proxy. Otherwise, one of the found ρ closest peer nodes to f is randomly chosen as the proxy. We denote the proxy as u_p.
4) m is sent to u_p and u_p would forward m to the server. Obviously, the IP address of m is changed after forwarding and then the IP address of u_i is preserved.
5) The server returns the response to u_p, and u_p forwards the response to u_i. Note that the response is encrypted with P_m by the server so that only u_i can decrypt.

Please note that the identifier of the original initiator does not exist in the request message, *i.e.*, the proxy and the LBS server don't need to know the identifier.

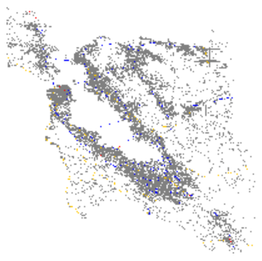


FIGURE 9. **San Francisco Bay Area.**

V. EXPERIMENT AND ANALYSIS

We implement the simulation system in Java and conduct experiments on a high performance server under different settings to evaluate its performance. The server we use is equipped with two CPUs of Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz which has 12 cores and 24 threads in total. The memory of the server is 128 GB and the operating system is CentOS Linux release 7.6.1810. We depend on the Network-based Generator of Moving Objects proposed in [17] to generate the traces of moving users on the map of San Francisco Bay Area (Fig. 9). The generator and the map can be accessed at [57].

The size of the map is about 159.9 $km * 158$ km. In our experiment, the masks of users are set to be 15 because a mask larger than 15 would make its desired cloaked region too small to preserve location privacy according to Table. 3. In order to give readers a basic understanding, we would like to mention that the total map in Fig. 9 can be covered by about 1, 5000 areas with a mask of 15.

Generally, users are not uniformly distributed in a map. After the Network-based Generator of Moving Objects [17] generates 10K user traces, we plot the distribution of these users on the map in Fig. 10. We can see that (a) nearly half of the users have less than 15 local users. The maximum number of local users is less than 400. (b) there are many cold areas with very few local users. Some areas have no user at all because these areas are in the water or have no road. Unless otherwise stated, users have the same anonymity requirement parameter k in an individual simulation and we conduct simulations under different k of 10, 20, 50, 100, 150 and 200 respectively to show the impacts of k.

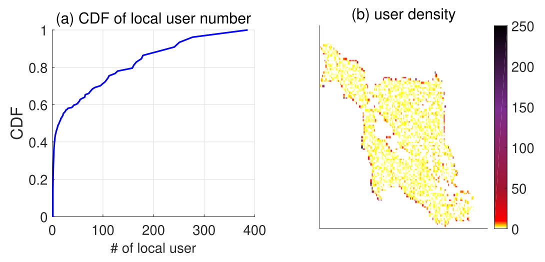


FIGURE 10. (a) The CDF of $|\mathcal{M}(dr_i)|$ (the number of local users) for each user u_i. (b) the user density at the beginning of simulation in each area with a mask of 15.

TABLE 5. Differences between distributed P2P schemes and our scheme.

Existing P2P-based schemes	Our scheme
based on ad-hoc networks	based on the Internet
peers trust each other	peers are not trustworthy
actual locations in final requests	cloaked regions in requests
TTP-like entities exist	No TTP-like entities

In the following, we first present a qualitative comparison of our method and other existing P2P-based schemes and then explain why we choose CloakP2P [12] as the baseline to conduct performance comparison. Finally, we present the quantitative experiment results on the resistance to attacks and the size of cloaked regions.

A. THE QUALITATIVE COMPARISON

We summarize the differences between existing distributed P2P-based schemes and our scheme in Table 5. As we have introduced in Section II-C, each existing P2P-based scheme has at least one of the four features and each feature has disadvantages. Please refer to Table 1 for details. Our scheme avoids the disadvantages by exploiting different architecture and mechanisms as we illustrated in previous sections. Since we have so much differences and apply to different scenarios, it is not appropriate to compare our schemes with them directly. What we have in common is that our scheme and many other existing P2P-based schemes employ cloaked regions and k-anonymity for the preservation of location and identity information.

Therefore, we focus on the quantitative experiment results on the resistance to attacks and the size of cloaked regions. We choose CloakP2P [12] as the baseline to conduct performance comparison because CloakP2P is able to generate very small cloaked regions (which is preferable) compared to other existing P2P schemes as we introduced in Section II-C. Many schemes, such as [38]–[40], also choose CloakP2P for comparison. By comparing the performance of our scheme and CloakP2P on the size of generated cloaked regions, we can evaluate the practicality and effectiveness of our scheme.

We focus on the resistance to attacks and the size of cloaked regions. Since Kademlia has been widely used in practice on the Internet for very long time, we ignore the performance evaluations on networking.

B. RESISTANCE TO ATTACKS

The most important feature we want to achieve is the ability to resist any attack on users' privacy, including their identities, precise locations and the content in their queries. In our design, we have exploited multiple techniques to preserve the privacy. For example, query messages are encrypted, therefore only the location-based service provider can learn their contents; users always use their cloaked regions instead of their precise locations and one cloaked region is shared by multiple users, therefore even the location-based service provider cannot learn the precise location and cannot determine the initiator of a request. Therefore, we can see that the scheme can work well even the location-based service server is cracked.

However, what if both the location-based service server and the privacy preservation system are cracked? As we have stated, if a Trusted Third Party (TTP) is used to determine cloaked regions for users and forward messages after processing, the TTP would be the target of attacks. Once the TTP is cracked, the attacker can learn all requests and infer private information. Therefore, we exploit two mechanisms to avoid the fragility of TTPs, 1) we resort to a P2P architecture to negotiate cloaked regions in a distributed manner, therefore no TTP is needed. 2) the message forwarding is conducted by random proxies, therefore there is no single entity that has knowledge of many messages. These mechanisms can mitigate the risk of privacy preservation systems being cracked. Unfortunately, the risk is not removed completely. We know that malicious attackers can collect information of peers by joining the P2P network and exchanging messages with peers, which means the privacy preservation system proposed by us can be cracked in extreme cases.

In extreme cases, one attacker can deploy malicious peer nodes to participate in negotiations in every area, and then it is capable of associating users' identities with their cloaked regions. Based on the information, the attacker can conduct *Center-of-cloaked-region Attack* or *Correlation Attack*. In this subsection, we will explain these two attacks and evaluate the ability of our scheme to resist these two attacks.

Our experiment settings are as follows. We generate traces for a number of users on the map, and then generate privacy requirements (k and mask) and requests for these users. We evaluate the scheme under three different distributions of requests among these users, *i.e.*, the uniform distribution, Zipfian [58] ($s = 0.5$) distribution and Zipfian($s = 0.8$) distribution. Note that the bigger s is, the more skewed requests are, *i.e.*, more requests are generated by fewer top active users.

1) CENTER-OF-CR ATTACK

In a center-of-CR attack, the attacker has cracked our P2P system by deploying many malicious peer nodes to participate in negotiations. Therefore, it has known each cloaked region and the corresponding users of each cloaked region. When the attacker intercepts a request, it first extracts the cloaked region from the request, and infers the user who is closest to the center of the cloaked region as the initiator of a request.

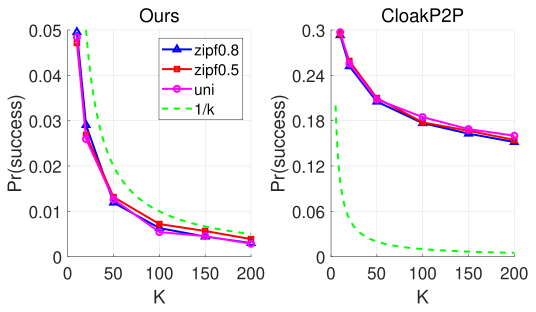


FIGURE 11. **The performance comparison between our scheme and CloakP2P under center-of-CR attacks (10K users, mask=15).**

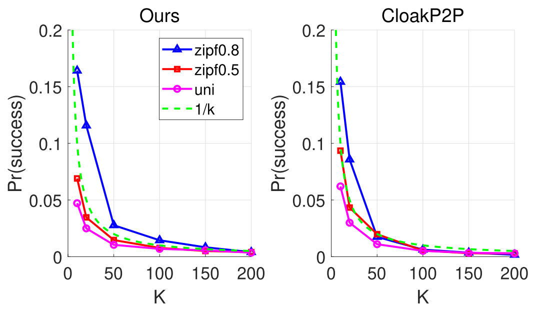


FIGURE 12. **The performance comparison between our scheme and CloakP2P under correlation attacks (10K users, mask=15).**

Obviously, it is required to know the precise location of each user in center-of-CR attacks. With our scheme, the precise location information is preserved, and then the attacker cannot launch a center-of-CR attack directly. Here, in order to evaluate the performance of resisting this kind of attacks, we just assume that the attacker has inferred the precise location of users using other methods.

In CloakP2P, the precise locations of users are required to be exposed to malicious peer nodes deployed by the attacker, therefore the attacker can launch a center-of-CR attack directly.

Fig. 11 presents the experiment results. In the experiments, there are 10K users who generate 10K requests in an hour. The performance is evaluated by the metric $Pr(success)$, which is the accuracy of the request initiators inferred by the attacker. If the accuracy is below $1/k$ (the inferred result is worse than a random guess), it can be regarded as the scheme is useful in terms of mitigating the risk of information leakage. We can see that our scheme achieves k-anonymity, *i.e.*, the curves are below $1/k$, no matter how the requests are distributed among users. CloakP2P is very vulnerable to center-of-CR attacks. It is a natural result of negotiating cloaked regions with closest neighbors.

2) CORRELATION ATTACK

In a correlation attack, besides cracking the P2P system, the attacker also intercepts requests at the LBS server. For each request, the attacker checks the cloaked region in it and maps it to a set of suspicious users (*i.e.*, users using this cloaked region). Note that we assume the attacker knows the users of each cloaked region at any time by participating in negotiations. As the attacker analyzes more and more requests, it can infer the initiator for each request in the following way.

Let $Q = \{q_1, q_2, \ldots, q_n\}$ be the set of intercepted requests, and $F(q_t)$ be the set of suspicious users for q_t. The attacker first calculates the number of occurrences of each user u_i appearing in $F(q_t)(t \in [1, n])$, *i.e.*, $O(u_i) = |\{q_t|u_i \in F(q_t), t \in [1, n]\}|$. Then the attacker infers that the request q_t is initiated by the user u_i that is in the set $F(q_t)$ and has the largest $O(u_i)$ among all users in $F(q_t)$.

Fig. 12 presents the performance of our scheme and CloakP2P under correlation attack. We can see that our scheme is useful under requests with uniform distribution and Zipfian($s = 0.5$) distribution. The curve under Zipfian ($s = 0.8$) distribution is above $1/k$, but it is approaching to $1/k$ as k increases. The overall performance is comparable to CloakP2P. Please note that it is very difficult for the attacker to satisfy all requirements of this attack, *i.e.*, cracking the system by deploying a lot of peer users, cracking or monitoring all (or most) requests to the location-based service server, and the monitored requests are heavily skewed among users.

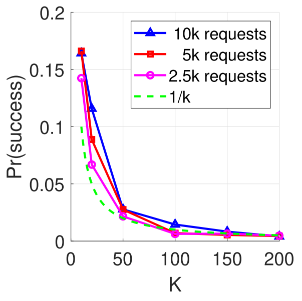


FIGURE 13. **The performance comparison among different number of requests under correlation attack (Zipfian($s = 0.8$)) for our scheme.**

Theoretically, as the attacker collects more and more requests, its inference accuracy would also increase. We conduct simulations for different number of collected requests and the distribution function of these requests among users is Zipfian($s = 0.8$), which is the worst case for our scheme. The result is presented in Fig. 13. We can see that although collecting more requests would increase the success rate of the attack, the success rate is decreasing and approaching to $1/k$ as k increases no matter how many requests are collected. Therefore, users can preserve their privacy by setting a larger k to resist correlation attacks effectively.

C. THE SIZE OF CLOAKED REGIONS

In the above subsection, we can see that the k-anonymity requirement can be satisfied by our system in most scenarios. Now we conduct experiments under different scenarios and look into the sizes of cloaked regions achieved by our system.

According to Table. 3, we regard that one user would like to specify its mask value between 9 and 15. A mask value beyond this range is either too big for location-based service

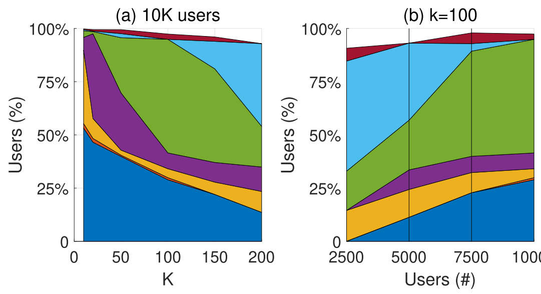


FIGURE 14. The ratios of users classified according to the size of cloaked region achieved by our system. From bottom to top, the respective size (measured by mask value) is from 15 to 9. The topmost white area represents the ratio of users that cannot obtain a satisfactory cloaked region.

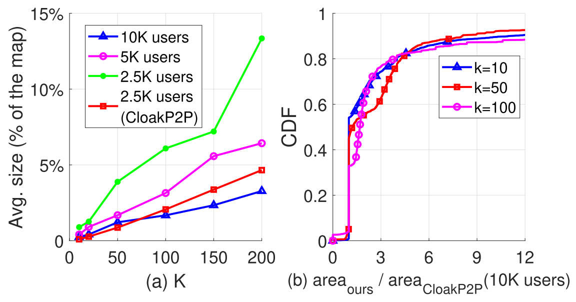


FIGURE 15. A comparison of the sizes of cloaked regions achieved by our scheme and CloakP2P. (a) shows the average size of cloaked regions. (b) shows the CDF of the ratio of the cloaked region size achieved by our scheme to CloakP2P.

or too small for privacy preservation. We would like to see the size of the final cloaked region for each user achieved by our system.

Fig. 14 presents the experiment results. We can see that some users cannot get cloaked regions with the specified desired size (specified by the mask value). As k increases, which indicates that users have stricter anonymity requirements, it is more likely for a user to get a larger cloaked region as Fig. 14 (a) shows. As the number of users increases, the user density also increases, thus more and more users can be successful in the local negotiation of smaller cloaked regions as Fig. 14 (b) shows. The result is consistent with our intuition.

In Fig. 15 (a), we vary the number of users and calculate the average size of the cloaked regions in our scheme under different k requirements. As a comparison, we also present the results under CloakP2P with 2.5K users. We can see that the average size increases with k and decreases with the number of users. Also, CloakP2P achieves smaller cloaked regions for users because it always exploits the smallest region covering at least k closest users.

In Fig. 15 (b), we compare the size of cloaked region achieved by our scheme with the size achieved by CloakP2P for each user. If the cloaked region a user u_i obtained in CloakP2P is smaller than its dr_i in our scheme, we extend the cloaked region in CloakP2P to the size of dr_i to avoid violating its privacy requirement. We can see that there are some users (although very few) who can achieve smaller cloaked regions in our scheme than in CloakP2P. There are nearly half of users who can obtain cloaked regions of the same size in two schemes ($x = 1$). But some users obtain larger cloaked regions in our scheme than CloakP2P. It is the cost to realize stronger location privacy preservation. Fortunately, it only causes some more communication overhead (the location-based service provider would return information about a larger area), which is not the main concern of users.

VI. CONCLUSION

In this article, we propose a distributed system to preserve various aspects of privacy of users in using location-based services. The privacy is preserved by replacing users' precise locations by their cloaked regions in requests and the cloaked region of one user should satisfy k-anonymity to realize anonymization and make it difficult for attackers to associate a request with its initiator. A user can specify its individual required parameter k of k-anonymity and the minimum size of its cloaked region.

We carefully design an encoding scheme of user identifiers, and the identifier of one user contains its coarse-grained location and also the size of its desired cloaked region. We also analyze a popular P2P architecture, Kademlia, and make necessary modifications to make it work well for our purpose.

With the help of identifiers and the P2P architecture defined by us, we design a distributed negotiation algorithm to determine a cloaked region for each user without exposing precise locations. Particularly, distributed negotiations are first conducted among users that desire the same cloaked region. Some users may be unsuccessful in the negotiations due to their stricter requirements, and then they would participate in negotiations of larger areas. We further mitigate the risk of information leakages by selecting a random proxy to forward each request between users and LBS servers.

Experiments show that our scheme can work well even under the most severe scenarios and it is achieved at the cost of that some users may obtain larger cloaked regions. Fortunately, generally the potential overhead caused by larger cloaked regions is not the main concern of users.

ACKNOWLEDGMENT

The authors would like to thank the editors and reviewers for taking time to review this article and for their suggestions that helped to improve this article.

REFERENCES

[1] R. Paulet, M. G. Kaosar, X. Yi, and E. Bertino, "Privacy-preserving and content-protecting location based queries," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1200–1210, May 2014.
[2] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervas. Comput.*, vol. 2, no. 1, pp. 46–55, Jan. 2003.
[3] M. Li, H. Zhu, Z. Gao, S. Chen, L. Yu, S. Hu, and K. Ren, "All your location are belong to us: Breaking mobile social networks for automated user location tracking," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*. New York, NY, USA: ACM, 2014, pp. 43–52.

[4] S. Zakhary and A. Benslimane, "On location-privacy in opportunistic mobile networks, a survey," *J. Netw. Comput. Appl.*, vol. 103, pp. 157–170, Feb. 2018.

[5] B. Liu, W. Zhou, T. Zhu, L. Gao, and Y. Xiang, "Location privacy and its applications: A systematic study," *IEEE Access*, vol. 6, pp. 17606–17624, 2018.

[6] C. Bettini, "Privacy protection in location-based services: A survey," in *Handbook of Mobile Data Privacy*. Cham, Switzerland: Springer, 2018, pp. 73–96.

[7] R. Shokri, J. Freudiger, and J.-P. Hubaux, "A unified framework for location privacy," 2010. [Online]. Available: http://infoscience.epfl.ch/record/148708

[8] J. Krumm, "A survey of computational location privacy," *Pers. Ubiquitous Comput.*, vol. 13, no. 6, pp. 391–399, 2009.

[9] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Pers. Ubiquitous Comput.*, vol. 18, no. 1, pp. 163–175, 2014.

[10] Z. Gardner, D. Leibovici, A. Basiri, and G. Foody, "Trading-off location accuracy and service quality: Privacy concerns and user profiles," in *Proc. Int. Conf. Localization GNSS (ICL-GNSS)*, Jun. 2017, pp. 1–5.

[11] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, May 2012.

[12] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Proc. 14th Annu. ACM Int. Symp. Adv. Geographic Inf. Syst. (GIS)*. New York, NY, USA: ACM, 2006, pp. 171–178.

[13] H. Hu and J. Xu, "Non-exposure location anonymity," in *Proc. IEEE 25th Int. Conf. Data Eng.*, Mar. 2009, pp. 1120–1131.

[14] Y. Che, Q. He, X. Hong, and K. Chiew, "X-region: A framework for location privacy preservation in mobile peer-to-peer networks," *Int. J. Commun. Syst.*, vol. 28, no. 1, pp. 167–186, Sep. 2013.

[15] K. Jung and S. Park, "Collaborative caching techniques for privacy-preserving location-based services in peer-to-peer environments," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 4497–4506.

[16] D. Rebollo-Monedero, J. Forné, A. Solanas, and A. Martínez-Ballesté, "Private location-based information retrieval through user collaboration," *Comput. Commun.*, vol. 33, no. 6, pp. 762–774, Apr. 2010.

[17] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.

[18] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *Proc. ICPS. Proceedings. Int. Conf. Pervas. Services*, ., Jul. 2005, pp. 88–97.

[19] T.-H. You, W.-C. Peng, and W.-C. Lee, "Protecting moving trajectories with dummies," in *Proc. Int. Conf. Mobile Data Manage.*, May 2007, pp. 278–282.

[20] P.-R. Lei, W.-C. Peng, I.-J. Su, and C.-P. Chang, "Dummy-based schemes for protecting movement trajectories," *J. Inf. Sci. Eng.*, vol. 28, no. 2, pp. 335–350, 2012.

[21] H. Jin Do, Y.-S. Jeong, H.-J. Choi, and K. Kim, "Another dummy generation technique in location-based services," in *Proc. Int. Conf. Big Data Smart Comput. (BigComp)*, Jan. 2016, pp. 532–538.

[22] C. A. Ardagna, M. Cremonini, E. Damiani, S. De C. di Vimercati, and P. Samarati, "Location privacy protection through obfuscation-based techniques," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*. Berlin, Germany: Springer, 2007, pp. 47–60.

[23] C. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati, "An obfuscation-based approach for protecting location privacy," *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 1, pp. 13–27, Jun. 2009.

[24] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," 2012, *arXiv:1212.1984*. [Online]. Available: http://arxiv.org/abs/1212.1984

[25] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*. New York, NY, USA: ACM, 2015, pp. 1298–1309.

[26] R. Lu, X. Lin, T. H. Luan, X. Liang, and X. Shen, "Pseudonym changing at social spots: An effective strategy for location privacy in VANETs," *IEEE Trans. Veh. Technol.*, vol. 61, no. 1, pp. 86–96, Jan. 2012.

[27] S. Gao, J. Ma, W. Shi, G. Zhan, and C. Sun, "TrPF: A trajectory privacy-preserving framework for participatory sensing," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 6, pp. 874–887, Jun. 2013.

[28] B. Ying, D. Makrakis, and H. T. Mouftah, "Dynamic mix-zone for location privacy in vehicular networks," *IEEE Commun. Lett.*, vol. 17, no. 8, pp. 1524–1527, Aug. 2013.

[29] B. Palanisamy and L. Liu, "Attack-resilient mix-zones over road networks: Architecture and algorithms," *IEEE Trans. Mobile Comput.*, vol. 14, no. 3, pp. 495–508, Mar. 2015.

[30] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. 1st Int. Conf. Mobile Syst., Appl. services (MobiSys)*. New York, NY, USA: ACM, 2003, pp. 31–42.

[31] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Achieving k-anonymity in privacy-aware location-based services," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2014, pp. 754–762.

[32] X. Chen and J. Pang, "Protecting query privacy in location-based services," *GeoInformatica*, vol. 18, no. 1, pp. 95–133, 2014.

[33] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in *Proc. 25th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2005, pp. 620–629.

[34] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: Query processing for location services without compromising privacy," in *Proc. 32nd Int. Conf. Very large data bases*, 2006, pp. 763–774.

[35] T. Xu and Y. Cai, "Exploring historical location data for anonymity preservation in location-based services," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, Apr. 2008, pp. 547–555.

[36] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with privacygrid," in *Proc. Proc. 17th Int. Conf. World Wide Web (WWW)*. New York, NY, USA: ACM, 2008, pp. 237–246.

[37] A. R. Beresford, "Location privacy in ubiquitous computing," Dept. Comput. Lab., Univ. Cambridge, Cambridge, U.K., Tech. Rep. UCAM-CL-TR-612, 2005.

[38] B. Niu, X. Zhu, Q. Li, J. Chen, and H. Li, "A novel attack to spatial cloaking schemes in location-based services," *Future Gener. Comput. Syst.*, vol. 49, pp. 125–132, Aug. 2015.

[39] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous location-based queries in distributed mobile systems," in *Proc. 16th Int. Conf. World Wide Web (WWW)*, 2007, pp. 371–380.

[40] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "MobiHide: A mobilea peer-to-peer system for anonymous location-based queries," in *Proc. Int. Symp. Spatial Temporal Databases*. Berlin, Germany: Springer, 2007, pp. 221–238.

[41] C.-Y. Chow, M. F. Mokbel, and X. Liu, "Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments," *GeoInformatica*, vol. 15, no. 2, pp. 351–380, 2011.

[42] Y. Che, Q. Yang, and X. Hong, "A dual-active spatial cloaking algorithm for location privacy preserving in mobile peer-to-peer networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2012, pp. 2098–2102.

[43] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz, "Analysis of the clustering properties of the Hilbert space-filling curve," *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 1, pp. 124–141, 2001.

[44] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003.

[45] S. Zakhary, M. Radenkovic, and A. Benslimane, "Efficient location privacy-aware forwarding in opportunistic mobile networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 2, pp. 893–906, Feb. 2014.

[46] Z. Zhai, "A P2P spatial-cloaking algorithm without exposing the collaborators for LBS," in *Proc. 4th Int. Conf. Inf. Sci. Cloud Comput. PoS(ISCC)*, vol. 264. Trieste, Italy: Sissa Medialab, Feb. 2016, p. 003.

[47] M. Ghaffari, N. Ghadiri, M. H. Manshaei, and M. S. Lahijani, "P^4QS: A peer-to-peer privacy preserving query service for location-based mobile applications," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 9458–9469, 2017.

[48] H. Jin and P. Papadimitratos, "Resilient privacy protection for location-based services through decentralization," *ACM Trans. Privacy Secur.*, vol. 22, no. 4, pp. 1–36, Sep. 2019.

[49] S. Zhang, K.-K.-R. Choo, Q. Liu, and G. Wang, "Enhancing privacy through uniform grid and caching in location-based services," *Future Gener. Comput. Syst.*, vol. 86, pp. 881–892, Sep. 2018.

[50] R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, and J.-P. Hubaux, "Hiding in the mobile crowd: LocationPrivacy through collaboration," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 3, pp. 266–279, Jun. 2014.

[51] R. Gupta and U. P. Rao, "Achieving location privacy through CAST in location based services," *J. Commun. Netw.*, vol. 19, no. 3, pp. 239–249, 2017.

[52] T. Peng, Q. Liu, D. Meng, and G. Wang, "Collaborative trajectory privacy preserving scheme in location-based services," *Inf. Sci.*, vol. 387, pp. 165–179, May 2017.

[53] T. Dargahi, M. Ambrosin, M. Conti, and N. Asokan, "ABAKA: A novel attribute-based k-anonymous collaborative solution for LBSs," *Comput. Commun.*, vol. 85, pp. 1–13, Jul. 2016.

[54] Z. Haitao, Z. Lei, F. Weimiao, and M. Chunguang, "A users collaborative scheme for location and query privacy," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2016, pp. 383–390.

[55] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Proc. Int. Workshop Peer-to-Peer Syst.* Berlin, Germany: Springer, 2002, pp. 53–65.

[56] P. J. Leach, M. Mealling, and R. Salz, *A Universally Unique Identifier (UUID) URN Namespace*, Standard RFC 4122, 2005.

[57] *Generator-IAPG*. Accessed: 2019. [Online]. Available: https://iapg.jade-hs.de/personen/brinkhoff/generator/

[58] G. K. Zipf, *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Cambridge, MA, USA: Addison-Wesley, 1949.

SHENGCHAO LIU received the B.Sc. degree in computer science from Jilin University, China, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, China. His research interests include network measurement and network security.

JESSIE HUI WANG received the Ph.D. degree in information engineering from The Chinese University of Hong Kong, in 2007. She is currently an Associate Professor with Tsinghua University. Her research interests include traffic engineering, cloud computing, network measurement, and the Internet economics.

JILONG WANG (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, in 2000. He is currently a Professor with Tsinghua University. His research interests include network measurement, network security, and SDN.

QIANLI ZHANG received the Ph.D. degree from the Department of Electronic Engineering, Tsinghua University, in 2007. He is currently an Associate Professor with Tsinghua University. His research interests include network security and network architecture.