Dong K, Gu T, Tao XP *et al.* Complete bipartite anonymity for location privacy. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 29(6): 1094–1110 Nov. 2014. DOI 10.1007/s11390-014-1493-y

Complete Bipartite Anonymity for Location Privacy

Kai Dong[1,2] (董 恺), Tao Gu[3] (顾 涛), *Senior Member*, *IEEE*, *Member*, *ACM*
Xian-Ping Tao[1,2,*] (陶先平), *Member*, *CCF*, *IEEE*, and Jian Lv[1,2] (吕 建), *Fellow*, *CCF*, *Member*, *ACM*

[1] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China*

[2] *Institute of Computer Software, Nanjing University, Nanjing 210046, China*

[3] *School of Computer Science and Information Technology, RMIT University, Melbourne, Victoria 3001, Australia*

E-mail: kaidong@smail.nju.edu.cn; tao.gu@rmit.edu.au; {txp, lj}@nju.edu.cn

Received October 28, 2013; revised April 18, 2014.

Abstract Users are vulnerable to privacy risks when providing their location information to location-based services (LBS). Existing work sacrifices the quality of LBS by degrading spatial and temporal accuracy for ensuring user privacy. In this paper, we propose a novel approach, Complete Bipartite Anonymity (CBA), aiming to achieve both user privacy and quality of service. The theoretical basis of CBA is that: if the bipartite graph of k nearby users' paths can be transformed into a complete bipartite graph, then these users achieve k-anonymity since the set of "end points connecting to a specific start point in a graph" is an equivalence class. To achieve CBA, we design a Collaborative Path Confusion (CPC) protocol which enables nearby users to discover and authenticate each other without knowing their real identities or accurate locations, predict the encounter location using users' moving pattern information, and generate fake traces obfuscating the real ones. We evaluate CBA using a real-world dataset, and compare its privacy performance with existing path confusion approach. The results show that CBA enhances location privacy by increasing the chance for a user confusing his/her path with others by 4 to 16 times in low user density areas. We also demonstrate that CBA is secure under the trace identification attack.

Keywords location privacy, k-anonymity, path confusion, query obfuscation, complete bipartite anonymity

1 Introduction

With the advances of global positioning system (GPS) and mobile devices, people can easily obtain their location information, and access a wide range of location-based services (LBS). To access LBS, users have to reveal their locations to location-based service providers (LSP). This poses significant privacy risks since users' location information may be disclosed.

Two different approaches have been proposed to preserve location privacy. Spatial and temporal cloaking[1] uses location blurring in which an accurate location is blurred into an area and all the requests from this area within a certain period of time are managed together to achieve anonymity. Some other studies use the same concept in different settings such as [2-4]. Another approach[5] achieves user anonymity by leveraging the concept of mix zone, which has been widely adopted in [6-8]. A mix zone exists if there are enough users located in the same place at the same time. Since no location information is reported when users are in a mix zone, their traces are "mixed". As such, the LSP cannot distinguish a user either when he/she is in a mix zone or after he/she moves out of the zone.

However, many LBSs such as GeoLife[9] and Facemap① rely on an accurate, continuous, and real-time stream of location information to provide high quality of service (QoS). It remains a challenging issue to balance the privacy with functionality. The aforementioned techniques do not guarantee the desired QoS. The cloaking-based techniques degrade the spatial accuracy and increase the delay in reporting users' locations; and the mix zone based techniques temporarily prevent users from reporting their locations in the zone area, resulting in the loss of their accurate locations.

This paper extends our previous work[10], and uses complete bipartite anonymity (CBA) to achieve both privacy and functionality (QoS) for LBS. The intuition behind CBA is simple. We group k nearby users into a specific region, named CBA zone, to ensure that the users enter or leave the region at the same time. Our idea is to confuse the paths of these users by connecting

Regular Paper

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61373011, 91318301, and 61321491.

*Corresponding Author

① http://zen-mobi.com/, Sept. 2014.

©2014 Springer Science + Business Media, LLC & Science Press, China

their traces with fake ones. In a CBA zone, a user not only reports his/her real trace, but also generates fake traces connecting to the traces of all other users. All the traces in a CBA zone compose a complete bipartite graph where each entry or exit point is a vertex and there exists a query trace for each pair of (*entry*, *exit*) (i.e., edge). In this case, we say this group of users satisfies complete bipartite anonymity, since the probability of tracing a user from his/her entry to his/her real exit is $1/k$. For this user group, the functionality is also achieved since every user is able to query LBS with his/her accurate, continuous, and real-time location information.

Although this idea works in principle, developing a realistic CBA scheme is non-trivial. The first question is that how nearby users can find each other to create a CBA zone as far as privacy is concerned. The LSP may not be trustworthy, thereby user privacy can be violated if a CBA zone is generated based on the information provided by the LSP. In this work, we propose a collaborative path confusion mechanism to enable nearby users to work together to generate a CBA zone without involving any trusted entities. Second, we use fake traces for path confusion[6]. At its entry point, each user generates the fake traces to the exit points of all other users. This means each user has to know in advance the exit points. To achieve this, we design a local prediction engine in which each user can predict these exit points. Third, how to ensure the fake traces resemble the real ones is a difficult task. Recent studies have demonstrated that fake traces can be distinguished from realistic user move patterns, e.g., Peddinti *et al.*[11] presented a classification attack that can identify up to 93.67% of real user trips from a dataset with 5 times fake user trips. To address this problem, we propose a cloaked obfuscation method which removes the characteristics in real move patterns to make it indistinguishable from fake ones.

The rest of the paper is organized as follows. We first discuss the related work in Section 2. We then describe the idea of CBA in Section 3, and the detailed scheme of CBA in Section 4. Section 5 presents the evaluation, and Section 6 concludes the paper.

2 Background and Related Work

In this section, we define what we mean by "location privacy", and survey the existing techniques preserving location privacy.

2.1 Understanding Location Privacy

According to Machanavajjhala *et al.*[12], "the key to defining privacy is to model knowledge of the attacker". To better understand users' location privacy in LBS, we must first look at how a user's location is disclosed. To access LBS, users have to reveal their locations to LSP. This poses significant privacy risks since the LSP may be untrustworthy and disclose users' information.

Anonymous communication techniques have been proposed to preserve location privacy. In these techniques, user ID is typically omitted and the network address is handled by mechanisms such as onion routing[13] to ensure user anonymity. However, revealing anonymous positional information still poses new problems[1]. The anonymous locations can be classified by their temporal and spatial relationships, to discover traces of different users[14], much like a trial of breadcrumbs left by Hansel and Gretel in the fairy tales. An adversary may mount an identification attack to identify and track a subject if the breadcrumb trial contains some "identifying locations". For example, if the LBS is invoked at the time when a user is in the garage, the location coordinates can be mapped to the address of the owner of the residence.

In this paper, location privacy refers to users' ability to avoid being tracked or identified by the aforementioned attack.

2.2 *k*-Anonymity and Spatial Cloaking

Much work has been done by leveraging the concept of k-anonymity to preserve users' location privacy. k-anonymity is originally proposed by Sweeney[15] to protect sensitive information from being disclosed. A table satisfies k-anonymity if every record is indistinguishable from at least $k-1$ other records with respect to every set of quasi-identifiers. To achieve k-anonymity, a generalization function is often used[16] to obscure the quasi-identifiers. If an aggregation of k reports of different individuals satisfies k-anonymity, the probability of identifying an individual will be theoretically $1/k$.

Gruteser and Grunwald[1] proposed spatial and temporal cloaking to preserve location privacy. In this technique, all the requests (from at least k different users) from an area within a certain period of time are managed together as an anonymity set to achieve k-anonymity. This idea of spatial cloaking has been widely adopted, e.g., in [2-4, 17].

In these approaches, there exists a trade-off between functionality and privacy, i.e., to ensure privacy, they sacrifice the granularity of location information. As a result, users cannot access accurate LBSs.

2.3 Mix Zone and Path Confusion

Path confusion[6] addresses the trade-off between location privacy and the QoS of LBS. In path confusion, two users change their pseudonyms when they cross their paths to become indistinguishable. This "cross"

of their paths serves as a small mix zone, where multiple users achieve anonymity when they occupy the same place at the same time[5].

Different algorithms are developed to implement path confusion. Hoh *et al.*[7] proposed path cloaking, which ensures two users pass a same place at different time, can incorporate a delay to expose this location to the LBS at the same time. Meyerowitz and Choudhury[14] proposed a solution named CacheCloak to cache service results. When a user requests location data, the CacheCloak server either returns cached data or obtains new data from the location-based service, thus a user's location is surrounded and mixed by other users' paths.

In these approaches, a trusted third party (TTP) is required to notify nearby users' existence and confuse their paths. However, the TTP may cause a single point of failure problem. Furthermore, in real life, we are not convinced that there exists such a TTP which is really trusted by all the users. Hence, the solution that relies on a single entity to collect and store all information from users may not work in reality.

2.4 Collaborative Location Privacy

Several approaches have been proposed to address the above limitation by providing a peer-to-peer (P2P) method to achieve collaborative location privacy. Shokri *et al.*[18] introduced MobiCrowd to increase privacy without assuming third party servers. In MobiCrowd, each user buffers context information he/she obtained until it expires; and this information is spread to nearby users through Wi-Fi ad hoc connection. Hence a user does not need to access the LBS if some other user sends him/her the requested information. Christin *et al.*[19] proposed a similar collaborative path hiding mechanism for participatory sensing applications. The main difference is that when users encounter each other, they exchange part of their sensed information; and when a user reaches his/her destination, he/she reports all his/her buffered information to the server. Thus the server can obtain the sensed information, and in the meanwhile the users' historical paths are mixed.

The existed P2P approaches have their limitations, since they cannot provide real time context information to the users.

2.5 Query Obfuscation

A different technique to preserve location privacy is to use dummies or fake queries. Kido *et al.*[20] proposed the concept of dummies to achieve k-anonymity when user density is low. Some recent researches use different data mining methods to generate dummies. Krumm[21] proposed a probabilistic model, and Shaker *et al.*[22] proposed a statistical clustering technique.

There are two limitations in these techniques. The first limitation is the cost — users have to query $k-1$ extra times to achieve k-anonymity, and the LSP has to process and respond to all these fake queries. Although the overhead may be acceptable in the realm of bits, the cost will be too high for real world services[5]. Second, recent studies have demonstrated that fake traces can be distinguished from realistic user move patterns, e.g., Peddinti *et al.*[11] presented a classification attack that can identify up to 93.67% of real user trips from a dataset with 5 times fake user trips.

2.6 Complete Bipartite Anonymity

In our previous work[10], we proposed CBA to achieve both privacy and functionality. For a certain area, the users' paths compose a bipartite graph, where each path contains a start point and an end point. CBA generates fake traces for these users, and turns the bipartite trace graph to a complete bipartite graph. CBA ensures user anonymity since the set of "end points connecting to a specific start point" is an equivalence class.

This work extends our previous work by enhancing CBA from both privacy and functionality. We now assume the confusion server is untrustworthy, and redesign the Collaborative Path Confusion protocol in Section 4. We propose a symmetric key exchange protocol in Subsection 4.1, leveraging on Wi-Fi Direct for the authentication between nearby users. Now users' privacy is ensured even if the confusion server performs some strong attacks like the man-in-the-middle attack. We also design a new prediction engine in CBA in Subsection 4.2, implemented by using the Google Map APIs. Nearby users now can predict where they meet each other more precisely. Experiments also show that users now achieve higher privacy and functionality.

3 Complete Bipartite Anonymity: The Principle

In this section, we first define CBA, and then use an example to show how CBA achieves both privacy and QoS. The terms and notations used later are listed in Table 1.

3.1 Definitions

Based on the breadcrumb effect, a user's queries can be classified and linked to form a trace.

Definition 1 (Trace). *A trace $\mathcal{T}$ is composed of a sequence of coordinates,*

$$\mathcal{S}_c = \big((x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\big),$$

Table 1. Notations Used

Term	Definition
$\mathcal{Z}$	Zone
$\mathcal{G}$	Graph
$\mathcal{T}$	Trace
$\mathcal{S}_c$	Sequence of coordinates
$\mathcal{S}_t$	Sequence of time stamps
$l(x, y)$	Location (coordinates)
t	Time stamp
d	Moving direction
o	Destination direction
f	Query frequency

and a sequence of corresponding time stamps

$$\mathcal{S}_t = (t_0, t_1, \ldots, t_n).$$

A user has a trace $\mathcal{T}$ means that, at each time $t_i \in \mathcal{S}_t$, this user reports $(x_i, y_i) \in \mathcal{S}_c$ as his/her location.

Definition 2 (Real Trace). *If all the coordinates $(x_i, y_i) \in \mathcal{S}_c$ that a user reports are his/her real locations, his/her trace $\mathcal{T}$ is a real trace.*

Definition 3 (Obfuscating Trace). *Otherwise, $\mathcal{T}$ is an obfuscating trace.*

For a group of users $\mathcal{U}$, each user has a different trace. The distance between any two traces can be measured by two thresholds: the spatial distance dis_{S} and the temporal distance dis_{T}. Based on these thresholds, we give the following definition.

Definition 4 (Nearby Trace). *Two traces $\mathcal{T}_1, \mathcal{T}_2$ are nearby traces if and only if they satisfy that*:

$$\begin{aligned}&\forall x_1, y_1, t_1, (x_1, y_1) \in \mathcal{S}'_{c_1} \wedge t_1 \in \mathcal{S}'_{t_1}\\ &\exists x_2, y_2, t_2, (x_2, y_2) \in \mathcal{S}'_{c_2} \wedge t_2 \in \mathcal{S}'_{t_2}\\ &\quad \rightarrow |(x_1, y_1) - (x_2, y_2)| < dis_{\mathrm{S}} \wedge |t_1 - t_2| < dis_{\mathrm{T}}.\end{aligned}$$

Definition 5 (Zone). *For a user group $\mathcal{U}$, each user $u_i \in \mathcal{U}$ has a trace $\mathcal{T}_{u_i}$. For each trace $\mathcal{T}_{u_i}$, if we can find a sub-trace $\mathcal{T}'_{u_i} = (\mathcal{S}'_c, \mathcal{S}'_t)$, where $\mathcal{S}'_c \subseteq \mathcal{S}_c \wedge \mathcal{S}'_t \subseteq \mathcal{S}_t$, satisfying any two of these sub-traces are nearby traces, there exists a smallest zone $\mathcal{Z}$ which covers all these sub-traces.*

Definition 6 (Trace Graph). *In a zone $\mathcal{Z}$, taking the first and the last coordinate and the time stamp of each trace as vertices, and taking all the traces (including both the real traces and the obfuscating traces) as edges, we get the trace graph $\mathcal{G}_{\mathcal{Z}}$ of $\mathcal{Z}$.*

Definition 7 (Complete Bipartite Anonymity). *If the trace graph of a zone is a complete bipartite graph, we name this zone a CBA zone. The user group in a CBA zone satisfies complete bipartite anonymity, and we name it a CBA group.*

As illustrated in Fig.1, in a CBA zone, a group of users satisfying CBA generate obfuscating traces to confuse real traces of each other, ensuring user privacy. Meanwhile, each user is still able to query LBS with his/her accurate, continuous and real-time location information, achieving a high QoS.

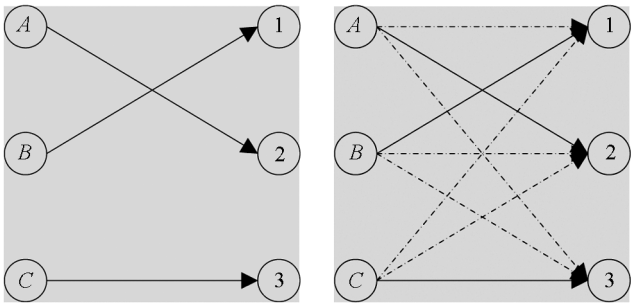


Fig.1. Complete bipartite anonymity. In a CBA zone, (a) presents the real trace of user A, B and C, respectively, represented by the solid line; (b) presents a complete bipartite graph, where obfuscating traces are generated, represented by the dotted line.

3.2 Illustration Example

We now use a two-user example to illustrate how CBA works in principle. Suppose users A and B access LBS anonymously, they may be traced and identified since they update their accurate locations continuously. When they drive close in an area, their queries form two nearby traces in a zone $\mathcal{Z}$. To generate a CBA zone from $\mathcal{Z}$, user A and user B apply the Collaborative Path Confusion protocol. First they discover each other with the help of a confusion server, apply a key exchange protocol to establish a symmetric key, and then predict their exit points using a local prediction engine. An obfuscating trace will be generated to connect one's entry to the other's exit. To resemble their obfuscating traces to the real ones, a cloaked obfuscation technique is used to remove the characteristics of their move patterns. In the zone area, their queries form a complete bipartite trace graph, thus $\mathcal{Z}$ is a CBA zone, and they form a 2-anonymity CBA group. After both users leave zone $\mathcal{Z}$, they are still untraceable. As shown in Fig.2, the grey area represents a CBA zone created by user A and user B. They become indistinguishable after they pass this CBA zone. In the following sections, we describe the detailed design of CBA.

4 Collaborative Path Confusion

The Collaborative Path Confusion (CPC) protocol enables nearby users to confuse their paths collaboratively. It involves three elements: the LBS server, the mobile user, and the confusion server. The confu-

sion server aggregates users' locations to notify nearby users' existence and provides a communication channel for these users. In our design, we do not assume the confusion server to be a trustworthy entity, and we will discuss related security issues in Subsection 4.2.

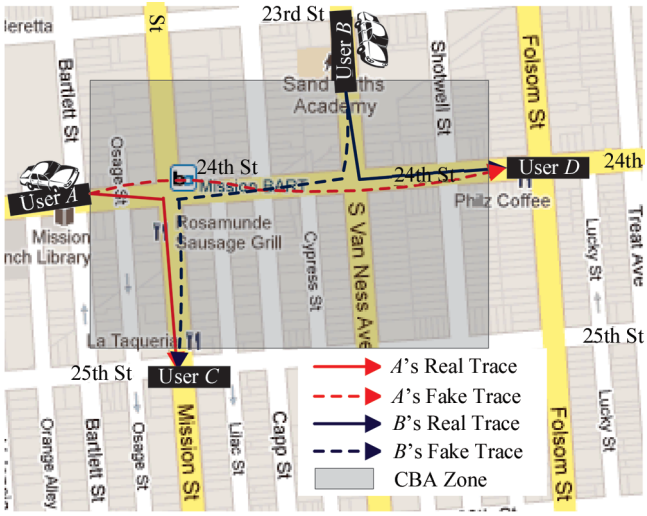


Fig.2. Example of 2-anonymity CBA zone (user A and user B each generates a fake trace to confuse each other's real paths).

Fig.3 gives an overview of how CPC works in a two-user scenario. A mobile user (i.e., A) queries an LBS server with his/her accurate location information continuously in real time. In the meanwhile, he/she generates a tile ID by using a public *location generalization* function, and reports this tile ID to the confusion server. The confusion server selects users sharing the same tile ID by using a *matching* function, and notifies these users the existence of each other. With this server, nearby users (e.g., B) exchange their moving patterns, including their locations, directions, and speeds by using the *information exchange* function. Based on these information, they predict their real paths by using a local *cross prediction* function, and use the *trace generation* function to generate the obfuscating traces. CPC ensures user anonymity since all these paths are confused, and it also ensures the quality of the LBS since users report their accurate locations.

We now describe the details for the aforementioned methods (i.e., *location generalization*, *matching*, *information exchange*, *cross prediction* and *trace generation*) in the following subsections.

4.1 Location Generalization and Matching

In CPC, the confusion server collects users' locations, and notifies users if they come close to each other. In case that the confusion server is not trustworthy, users only report their generalized locations with pseudonyms, described as follows.

By using GPS, a user A obtains his/her current location which is represented by a GPS coordinate with latitude and longitude, e.g., $(+00.123\,456, -01.234\,567)$. We then generalize the coordinate to an area with a radius of 100+ meters, which is reasonable for building

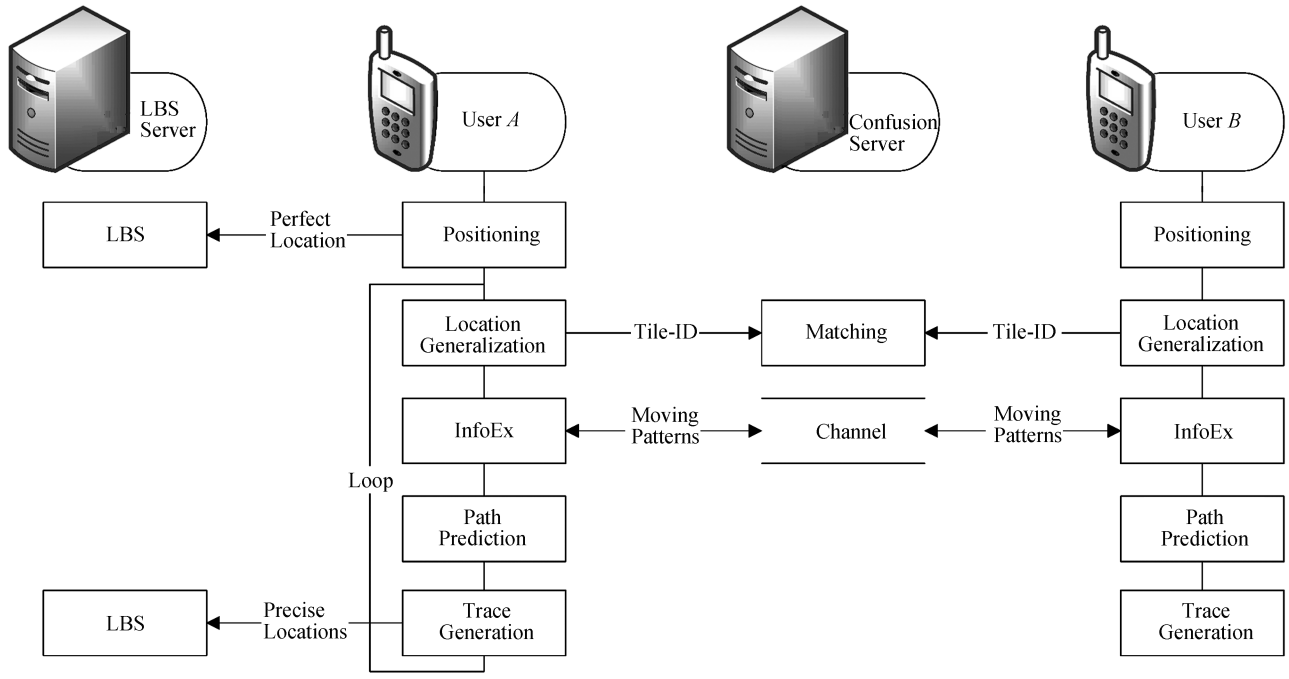


Fig.3. Overview of the CPC protocol.

a CBA zone. This can be done by rounding off the coordinate to the third decimal (the third decimal represents an accuracy of 110 m to 160 m), i.e., $(+00.123, -01.235)$. A tile ID is then generated by appending the longitude bits to the latitude bits (a sign bit is added), i.e., A's tile ID is 100123001235.

We define the *matching* method to find nearby users for any given user. Finding all the users sharing a same tile ID can be implemented by an SQL operation in the database (table T) maintained by the confusion server.

"SELECT $t_2.id$ WHERE $t_2.tile\_id = t_1.tile\_id$ (SELECT $t_1.id$)".

Since nearby users may be located in two neighboring tiles, the *matching* method returns all the nearby users within the same tile, as well as the users in the next tile.

4.2 Information Exchange

In CPC, users require other users' information including location, time, speed, and direction to predict their paths. The information is transmitted via the communication channel provided by the confusion server. The transmitted information may be revealed by attacks such as eavesdropping if the confusion server is untrustworthy. Simple key exchange protocols such as Diffie-Hellman key exchange may not secure this process since the confusion server may masquerade as a normal user or perform the man-in-the-middle attack to obtain the shared symmetric key.

To address this problem, we design a symmetric key exchange protocol implemented in the *information exchange* method to secure all the information exchange among nearby users. Users can first apply the Diffie-Hellman (D-H) key exchange protocol to exchange a symmetric key which may not be secure, then use Wi-Fi Direct to discover each other and authenticate the symmetric key. Our symmetric key exchange protocol operates in four phases: D-H key exchange, peer discovering, authentication, and encryption and transfer, as illustrated in Fig.4.

D-H Key Exchange. Two nearby users (i.e., A and B) who wish to exchange information will need to establish a symmetric key k_i via the confusion server applying the D-H key exchange protocol. To do this, they first negotiate a public prime number p and a public base g. One user, say A, then chooses a secret integer a and sends a message m_A to B where $m_A = g^a \bmod p$; correspondingly, B chooses a secret integer b and sends $m_B = g^b \bmod p$ to A. Next, A computes $s_A = m_B^a \bmod p$ and B computes $s_B = m_A^b \bmod p$. Since $s_A = m_B^a \bmod p$ is equal to $m_A^b \bmod p = s_B$, A and B now share a common secret key generator $s = s_A = s_B$ for computing a symmetric key k_i. k_i may not be secure since the D-H key exchange protocol does not provide user authentication.

Adversary Model. We suppose two adversary models. A weak adversary can perform a masquerade attack, where the adversary masquerades as user A to obtain B's accurate location and his/her moving pa-

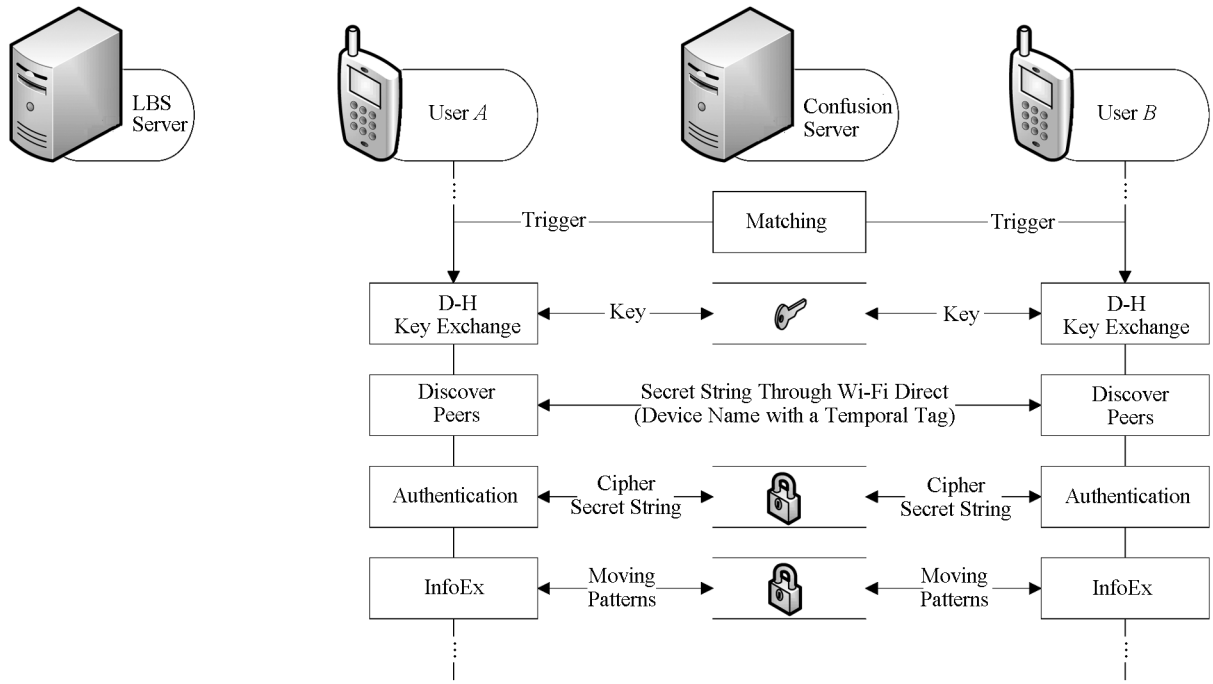


Fig.4. Symmetric key exchange protocol.

tterns, or vice versa. A strong adversary (e.g., when the confusion server is compromised) can perform the man-in-the-middle attack (even if A and B are both real users) and establish two distinct D-H key exchanges (one with A and the other with B) to obtain both their accurate locations. In CPC, we use Wi-Fi Direct to authenticate each user and defend these two attacks, which will be described in the following phases.

Peer Discovering. To discover the existence of nearby users, we use the Wi-Fi Direct protocol since it is widely available in many smartphones. Wi-Fi Direct provides an API, the *discoverPeers*() function for this purpose. To do so, user A generates a string $A'|t_A$ consisting of his/her pseudonym A' and a time tag t_A, and then uses k_i to encrypt this string and moderates his/her device name as cypher text $A'' = Enc_{k_i}(A'|t_A)$. After that, A calls the *discoverPeers*() function. Meanwhile, user B sets his/her device name as $B'' = Enc_{k_i}(B'|t_B)$ and also calls the *discoverPeers*() function. At the end of the *peer discovering* phase, user A finds a nearby peer B'', and user B finds a nearby peer A''.

Authentication. To make k_i secure, A and B decrypt his/her peer's device name to get its pseudonym respectively. In this case, A is expected to get B', and B gets A' to authenticate each other. The weak adversary will not be able to masquerade as a fake user unless it controls a device near the target user.

We now prove that the strong adversary will not be able to perform the man-in-the-middle attack as well. When the adversary establishes two distinct D-H key exchanges, it may masquerade as either a real user (i.e., A' or B') or a fake user (i.e., C). In the case of a fake user C, it will be recognized since A and B do not discover a peer with the device name $Enc_k(C|t)$. In the case of a real user A' or B', it shares a key k_1 with A and another key k_2 with B. This means users A and B do not share the same key, thus A cannot use k_1 to decrypt $B'' = Enc_{k_2}(B'|t_B)$, and vice versa for B. As a result, the adversary can be recognized.

Encryption and Transfer. Using k_i, A and B can encrypt their messages. They exchange information including accurate location, time, speed, moving direction, and destination direction. The moving direction (i.e., the direction that a user is currently heading) can be obtained by the mobile phone from its accelerometer readings. The destination direction is set by a user.

4.3 Cross Prediction

After information exchange among nearby users, they predict where they will meet each other. This is done by the *cross prediction* method consisting of the following two steps. First, it applies a prediction engine to compute an area where two users are present at the same time, and it then refines the area with a road map.

Prediction Engine (*Named JointCache*). A method $\mathcal{F}_Z$ is called by both users A and B to generate three areas.

$$\mathcal{F}_Z : \mathcal{L}, \mathcal{D}, \mathcal{O} \to \mathcal{Z}.$$

It takes inputs of current locations $l \in \mathcal{L}$, moving directions $d \in \mathcal{D}$ and orientations (destination directions) $o \in \mathcal{O}$ from both users, and outputs three areas $z_i \in \mathcal{Z}$.

As illustrated in Fig.5, users A and B move approaching to each other. d_A, o_A, and d_B form an triangle area, and we name this area A's *cache zone*, similar for B. The two cache zones have an overlap area (i.e., an quadrangle area), named the *joint zone*. The joint zone is where the two users' paths will intersect each other. Both the two cache zones form an area named the CBA *zone* where we can find a complete bipartite graph with four vertices.

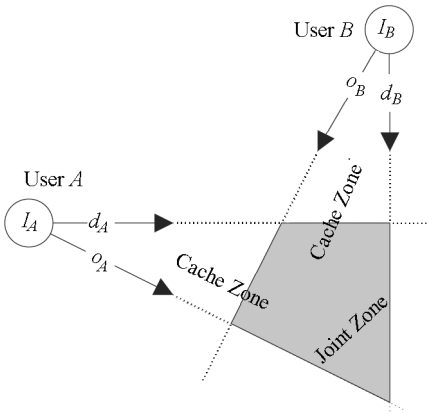


Fig.5. Joint zone and cache zone generation. The grey area represents the joint zone generated by user A and user B, and the white area represents the cache zones.

Refinement. With the knowledge of a local map, $\mathcal{F}_Z$ can be improved. We first describe the points of interest (POIs). A continuous area z_i is composed of infinite location points, and only a definite number of these location points can be inputs of certain LBS. For any z_i, we use Google Maps API[②] and find a set of location points $\{l_i\}$, satisfying that 1) l_i is a point of a road in Google Maps, 2) the discrete location points $\mathcal{L}_i$ are sufficiently dense to ensure accurate QoS, and 3) l_i is as sparse as possible to have a minimum i. These location points are POIs in *JointCache*.

To decide the POIs, a user first generates a set of discrete location points $\{\mathcal{L}_i\}$ from a continuous area $\mathcal{A}$. Let

[②] https://developers.google.com/maps/, Sept. 2014.

$$\mathcal{A} = \{L_{(x,y)} | x_{\min} \leqslant x \leqslant x_{\max} \wedge y_{\min} \leqslant y \leqslant y_{\max}\},$$

where (x, y) represents the coordinates. Then the user calculates

$$\{\mathcal{L}_i\} = \{L_{(x,y)} | L_{(x,y)} \in \mathcal{A} \wedge x \bmod r = 0 \wedge y \bmod r = 0\},$$

where r indicates the radius of an area, which is precise enough to meet accurate QoS. The area $\mathcal{A}$ can be divided into several grids, the side length of which equals to r, and each point in $\{\mathcal{L}_i\}$ is the center of a grid. For any points within $\mathcal{A}$, we can find a point in $\{\mathcal{L}_i\}$ such that the distance between these two points is less than $r/\sqrt{2}$. Note that even in very severe situations, when the LBS requires very accurate locations, the distance can be larger than 10 meters due to the accuracy of GPS positioning. Based on this understanding, *JointCache* uses the set of discrete location points $\{\mathcal{L}_i\}$ to represent the continuous area $\mathcal{A}$ when accessing LBS.

From $\{\mathcal{L}_i\}$ we choose the points satisfying the following conditions as POIs: 1) There must be at least one road, which can be obtained from Google Maps API, passing through the grid which contains the point. 2) In the grid, the centerline of this road must be longer than $r/2$. 3) This road is reachable to the current location of the user in the road graph. Fig.6(a) illustrates how to decide the POIs and Fig.6(b) shows an example.

We define the POIs laying on the border of the joint zone as vertices of the trace graph, one of which must be the entrance point of the user to the joint zone, while the others are candidate exit points as outcomes $\mathcal{C}$ of the prediction.

4.4 Trace Generation

Having the candidate exit points, a *trace generation* method $\mathcal{F}_T$ is called by both users A and B to generate all the traces.

$$\mathcal{F}_T : \mathcal{L}_f, \mathcal{C}, t, f \to \{\mathcal{T}\}.$$

This method takes the start location $\mathcal{L}_s \in \mathcal{L}_f$, the end location $\mathcal{L}_e \in \mathcal{C}$, and the corresponding time stamps t_s, t_e, and a reporting frequency f as inputs, and outputs a trace $\mathcal{T}_i = (\mathcal{S}_{i_c}, \mathcal{S}_{i_t})$. Here, the frequency means the frequency of a user accessing a certain LBS (or reporting his/her locations to the LBS). This frequency can be identifying information since two users with different reporting frequencies can easily be distinguished. In our method, we require the users A and B change their frequencies to a similar one. The frequency f can be different for different LBSs to ensure functionality.

We use Google Maps API to implement $\mathcal{F}_{\text{TraceGen}}$. The API takes the source and the destination pair $(\mathcal{L}_s, \mathcal{L}_e)$ as inputs, and generates the shortest path between them. The locations in $\mathcal{S}_c$ are evenly distributed in the shortest path, and the time stamps in $\mathcal{S}'_t$ satisfy:

$$\forall i, t_s < t_i \leqslant t_e \wedge t_i = t_{i-1} + 1/f.$$

Trace Identification Problem. One of the most significant challenges in the query obfuscation based techniques is how to generate fake traces indistinguishable from real user move patterns. Peddinti *et al.*[11] presented two types of attacks depending upon whether a short-term query history is available. When the history is available, a classification attack can be performed by using machine learning such as the support vector machine (SVM) classifier which can be trained with the user training data and the fake query training data generated from known user trips. In the absence of history, a trip correlation attack can be performed based on two metrics — distance and average speed. For the security parameter $k = 5$, i.e., four fake traces will be generated to obfuscate one real trace, the classification

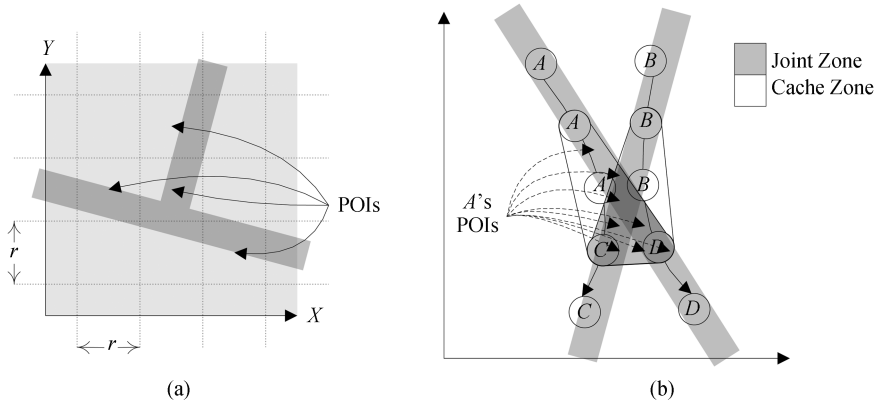


Fig.6. Deciding the POIs. (a) *JointCache* outputs the grey area, then narrows it down to roads with the knowledge of a local map. POIs are spread on these roads. (b) Example of POI decision.

attacks can identify up to 93.67% of user trips, with only 2.02% of fake trips misclassified, and the trip correlation attacks can increase the user query identification probability from 20% to 40%.

Cloaked Obfuscation. We address the trace identification problem from a new perspective. Instead of generating fake traces resemble to the real ones, we remove the characteristics in real user move patterns to make it resemble to the fake ones while the accuracy of the location information is still guaranteed. We name this method "cloaked obfuscation".

Before we introduce the details of cloaked obfuscation, we first need to define one special type of obfuscating traces — the cloaking trace.

For any real trace, which is composed of real locations and real time stamps

$$\mathcal{T} = \big(((x_0, y_0), t_0), ((x_1, y_1), t_1), \ldots, ((x_n, y_n), t_n)\big),$$

the user can generate a corresponding cloaking trace. The first step is to use the prediction engine with the input of the real start point to predict the end point $((x'_n, y'_n), t'_n)$. The second step is to call the trace generation method $\mathcal{F}_T$ to connect the real start point and the predicted end point, and generate an obfuscating trace,

$$\mathcal{T}' = \big(((x_0, y_0), t_0), ((x'_1, y'_1), t'_1), \ldots, ((x'_n, y'_n), t'_n)\big).$$

This cloaking trace has the following features.

1) The distance between the cloaking trace and the real trace is small. If the prediction is accurate and the method $\mathcal{F}_T$ outputs a path that the user really moves through, for every point in the real trace, there exists a point in the cloaking trace satisfying that the spatial distance dis_{S} and the temporal distance dis_{T} of these two points are small enough. This means the user can access LBS with the cloaking trace, while QoS is still guaranteed.

2) The cloaking trace and the other obfuscating traces can be indistinguishable. Since the cloaking trace is generated using the same function $\mathcal{F}_T$ as normal obfuscating traces, they are indistinguishable under attacks based on data mining techniques without extra knowledge on the user's real trace. We prove the security of the cloaked obfuscation method by reducing it to that of the mix zone (in Subsection 5.4).

An example is shown in Fig.7. The solid line represents the user A's real trace, and the dotted line represents A's fake trace. In Fig.7(a), using trace identification, the real trace and the fake trace can be distinguished. Fig.7(b) shows how this problem is solved by CBA. Every time user A accesses LBS in a CBA zone, he/she sends only fake queries with his/her fake locations (represented as $F1$ and $F2$). Since $F1$ and $F2$ are generated with the same algorithm, with no other background knowledge, they cannot be distinguished using machine learning techniques.

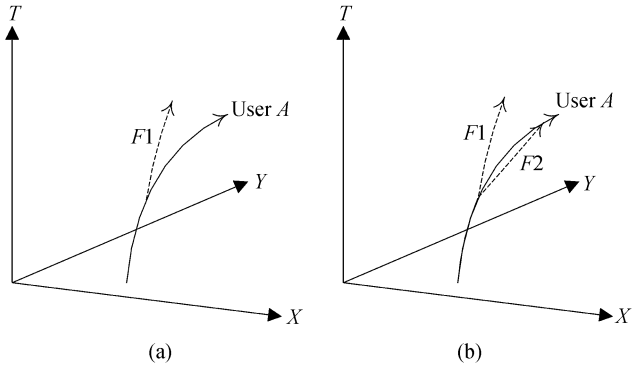


Fig.7. Real trace, obfuscating trace, and the predicted cloaking trace. The solid lines indicate A's real traces, and the dotted lines indicate the fake traces. In (a), using machine learning, A's real trace and fake trace $F1$ (obfuscating trace) can be distinguished based on his/her move patterns. In (b), user A generates a fake trace $F2$ (cloaking trace) to approximate his/her real trace — in this case, $F1$ and $F2$ are indistinguishable.

4.5 CPC for Multiple Users

In the situation where there are multiple users in a CBA zone, i.e., k users nearby, we now illustrate how to create a k-anonymity CBA zone with CPC. When k users are close by, we can find a sub-trace for each user where all these sub-traces are nearby traces, and a zone $\mathcal{Z}$ which concludes all these sub-traces. Each pair of these k users can create a 2-anonymity CBA zone $\mathcal{Z}_i$, where $\mathcal{Z}_i \subseteq \mathcal{Z}$. If each possible pair combination of users creates a CBA zone, we have $C(k, 2)$ obfuscating traces in $\mathcal{Z}$ and any pair of (*entry*, *exit*) is connected. Thus $\mathcal{G}_{\mathcal{Z}}$ is a complete bipartite graph and $\mathcal{Z}$ is a k-anonymity CBA zone.

Fig.8(a) depicts an example of 3-anonymity CBA zone. For users A, B, and C, each pair combination generates a 2-anonymity CBA zone, i.e., A and B generate zone I, B and C generate zone II, A and C generate zone III. Combining these three zones, we obtain a 3-anonymity CBA zone, represented as the large circle. In contrast, if A and C are not close enough, they will not be able to generate a 3-anonymity CBA zone, as illustrated in Fig.8(b).

In the crowded areas of a city, users have too many chances to meet others. With k increasing, the number of obfuscating traces ($C(k, 2)$) can be very large. It may be costly in terms of resources used to generate, send and serve the obfuscating queries. We address this issue by temporarily restricting a user from calling the *information exchange* method.

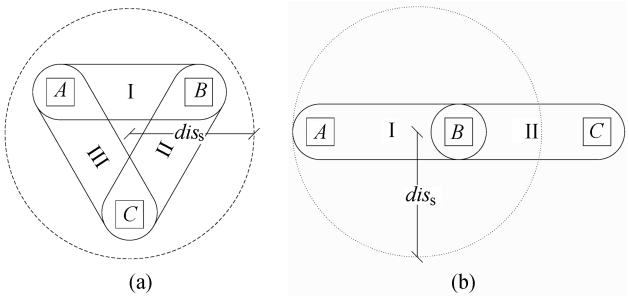


Fig.8. CBA zones with multiple users. In (a), users A, B, C are nearby users. They can generate a 3-anonymity CBA zone which is composed of three 2-anonymity CBA zones. In (b), since users A, C are not close enough, they cannot generate a 3-anonymity CBA zone.

5 Evaluations

We evaluate the CBA scheme from the following aspects.

• Can CBA guarantee privacy and functionality at the same time?

• Can the CPC protocol adapt to real applications?

• Is the prediction engine accurate?

• Is the obfuscation process secure under various attacks?

5.1 Evaluation of CBA Performance

In this subsection, we evaluate the performance of CBA using real taxi cab traces[23]. The cab traces are collected in the Cabspotting project③, which tracks the location of the cabs in the San Francisco Bay area using on-board GPS devices for 2 071 530 seconds (i.e., 24 days approximately). Each cab reports its location to the server at various time intervals (e.g., 10 seconds, 8 minutes, 1 hour). The distribution of the time intervals based on collection frequency is shown in Fig.9. Each cab's reports are aggregated to form a mobility trace and saved in a separate ASCII file which contains many lines of (*latitude, longitude, occupancy, time*), where the latitude and the longitude are in decimal degrees, the occupancy shows if a cab has a fare (1 = occupied, 0 = free) and the time is in UNIX epoch format. The dataset contains 11 219 955 records of 536 cabs in total.

5.1.1 Parameter Setting

In the implementation of CBA, there are three key parameters in generating CBA zones: the speed δ_ϕ and the turning angle δ_ψ of users, and the query frequency f. We set these parameters as follows. Fig.10(a) shows the CDF of the speed for all the users and only the frequent querying users, respectively. We set δ_ϕ to 36 km/h since most of the users drive slower than this speed. Fig.10(b) shows the CDF of the turning angle between any two records when a user is making a turn. We observe that it is unlikely that users turn at an angle larger than $\pi/2$ (even if a user makes a "U" turn, his/her turning angle between two records is not π), thus we set δ_ψ to $\pi/2$. Fig.10(c) shows the CDF of query intervals. We set f to 1/60 Hz since 80% of the queries are sent within 60 seconds.

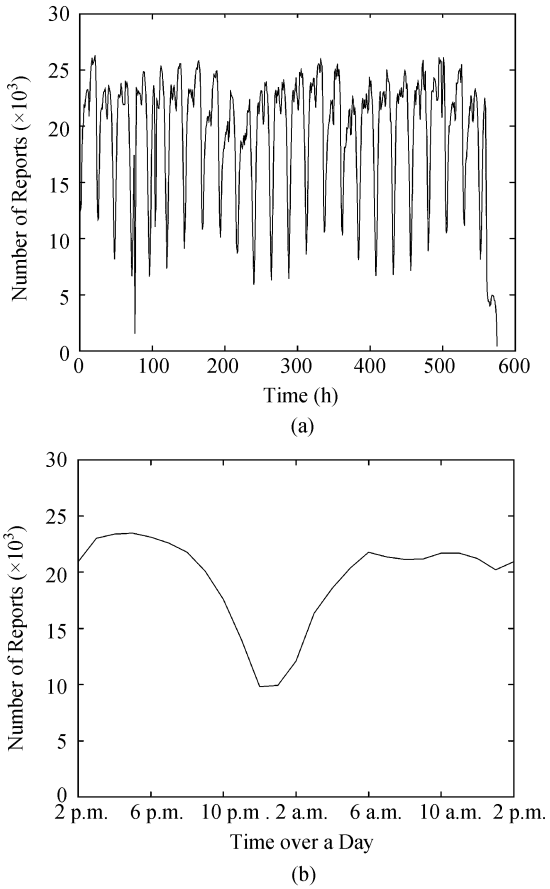


Fig.9. Features of the cabspotting dataset. (a) Distribution over 24 days. (b) Distribution in a day.

5.1.2 System Performance

Among the existing techniques, two approaches — the path confusion approach[6] and the query obfuscation approach[22] — can preserve privacy while guaranteeing the LBS functionality. We compare the performance of CBA with these two approaches, and report the results in the subsections below.

Waiting Time. We use waiting time t_w to indicate the time interval when a user is not covered by a CBA. It is defined as the time interval between a user leaving

③http://cabspotting.org/, Oct. 2014.

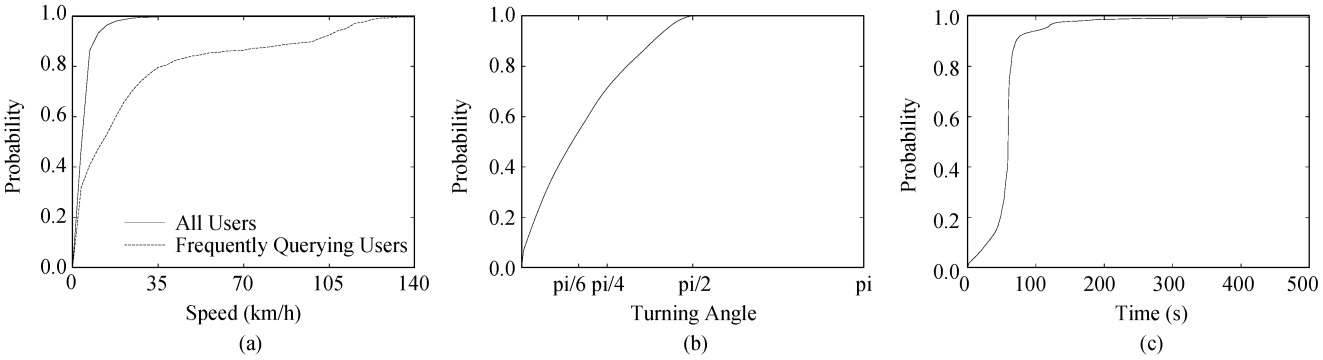


Fig.10. CDF of different parameter settings. (a) Speed δ_ϕ. (b) Turning angle δ_ψ. (c) Query frequency f.

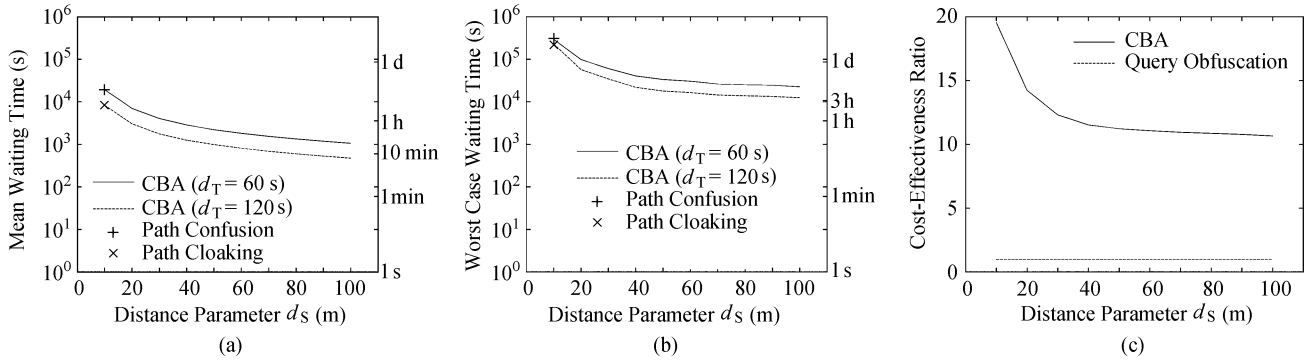


Fig.11. System performance comparison. (a) Comparison of mean waiting time between CBA and path confusion. (b) Comparison of worst case waiting time between CBA and path confusion. (c) Comparison of cost-effectiveness ratio between CBA and query obfuscation. The distance parameter d_S and the time parameter d_T indicate the upper bounds of the CBA zone in the space dimension and the time dimension.

a CBA zone and joining another CBA zone. The longer the waiting time a user has, the more likely the user can be identified. In the path confusion or mix zone approach, users' location can be protected when two users are presented in the same place at the same time. However, in the low user density areas of a city, people may not meet each other very frequently, resulting in long waiting time. CBA does not require that users have to meet each other. Instead, users in a CBA zone can confuse their paths with each other using obfuscating traces.

We compare the waiting time between CBA and path confusion. The results in Fig.11(a) and Fig.11(b) show that CBA increases the chance for a user to join in an anonymity group by about 10 times. Fig.11(a) shows that on average a user in CBA creates a CBA zone every 10 minutes, as compared to more than two hours in path confusion. Fig.11(b) shows that even in the worst case, a user is still able to create a CBA zone every three hours, and this is in comparison to longer than a day in path confusion. Both figures show that the waiting time decreases with a larger CBA zone. However, a larger CBA zone may increase the cost of resources spent to generate obfuscating queries, which we evaluate in the next experiment.

Query obfuscation does not have this same-place-same-time problem since a user can generate fake queries whenever he/she wants. CBA achieves higher anonymity degree since in low user density areas, it is easier to perform a classification attack to identify the obfuscating queries[11].

Cost-Effectiveness Ratio. To indicate the effectiveness of resources used in CBA, we define cost-effectiveness ratio as follows.

$$r = \frac{k}{\sum_u \mathcal{W}_{\mathcal{T}} / \sum_u \mathcal{W}_{\mathcal{T}_R}},$$

where k represents the anonymity degree (the k-anonymity parameter), and $\mathcal{W}$ represents the resource consumed (quantified as the number of queries), $\mathcal{T}$ represents a trace generated by user u, and $\mathcal{T}_R$ represents his/her real trace. To achieve the same anonymity degree, less resources used result in a higher cost-effectiveness ratio.

In this experiment, we compare CBA with query obfuscation in terms of r, as shown in Fig.11(c). From the figure, we observe that the cost-effectiveness ratio of CBA is much higher than that of query obfuscation. For query obfuscation, the cost-effectiveness ratio is constant: $r_{\text{Query Obfucation}} = 1$, since users should ge-

nerate $k-1$ fake queries for every real query to achieve k anonymity. Specifically, when $k=1$, we have $r_{\text{NoProtection}}=1$. For CBA, fake queries are generated only in CBA zones, thus only a small proportion (i.e., $1/m$, where $m>1$) of queries are fake queries. The cost-effectiveness ratio for this user can be calculated as follows.

$$r_{\text{CBAzone}}=\frac{k}{\mathcal{W}_{\mathcal{T}_O}+\mathcal{W}_{\mathcal{T}_P}/\mathcal{W}_{\mathcal{T}_P}}=\frac{k}{1+1/m}=\frac{m\times k}{m+1}.$$

Fig.11(c) also shows that the cost-effectiveness ratio decreases with a larger CBA zone, due to more fake queries generated. We find that even when the diameter of a CBA zone is up to 100 meters, the cost-effectiveness ratio of CBA still achieves 10, i.e., to achieve the same anonymity degree, CBA only consumes at most 10% of the resources consumed in query obfuscation. As mentioned earlier in the previous section, there exists a trade-off between the waiting time and the cost-effectiveness ratio. According to our experimental results, we find that setting the spatial distance parameter dis_{S} (the upper bound of the diameter of CBA zones) to 100 meters achieves the best trade-off.

Path confusion does not consume additional resources since it does not generate fake queries. Compared with path confusion, CBA consumes at most 10% more resources while solving the same-place same-time problem, and achieves a higher anonymity degree, which is measured in the following subsection.

5.1.3 Privacy Performance

We now evaluate the privacy performance of CBA under the de-anonymizing and re-identification attack.

With the knowledge of user queries, an attacker may mount an identification attack to identify a target. In this case, the spatial and temporal information of a query will be analyzed by an attacker to track an anonymous user. If any of these locations in the trace can be linked with a certain identity, the attacker knows this user's real identity with high confidence.

Attacker Model. To simulate an identification attack, we must first consider how an attacker can link a location with a certain identity. Let $p_u(lat, long)$ represent the probability that an attacker knows user u's location in the latitude and longitude coordinate $(lat, long)$. We have $p_u(lat_i,\ long_i)=1$, if location $(lat_i, long_i)$ is an identifying location for user u. The dataset we used does not contain drivers' identifying location information due to privacy reason. Thus, we introduce two models to simulate how an attacker identifies a user. In both models, we assume the LSP and the confusion server are untrustworthy, and hence the attackers may know all the information the LSP or the confusion server obtained.

Designation Model. From the dataset, we find every cab occasionally stops querying or stops moving at different locations for various periods of time. For a user u, we treat a location as an identifying location in either the following situations: u stops querying for more than 10^4 seconds (about 3 hours); or u stops moving but keeps querying for more than 10^3 seconds (about 17 minutes). Based on this criteria, we designate the identifying locations for each cab. Within the 24-day trips of 536 cabs, there are in total 376 896 times that a cab enters an identifying location, and on average, each cab is identified for every 50 minutes.

Iteration Model. The more queries sent to the LSP, the more likely the user can be identified. In the second model, we use an even distribution model to simulate the trend that the probability of identifying a user is increased with the number of queries sent by a user. In this model, no specific locations are treated as identifying locations. Let $\mathcal{P}_i$ denote the probability of identifying a user who queries i times ($p=\mathcal{P}_1$ is a constant).

$$\mathcal{P}_i=1-(1-\mathcal{P}_{i-1})\times(1-p)=1-(1-p)^i.$$

Privacy Metrics. Hol *et al.*[7] used information theoretic metrics to measure the uncertainty or confusion in tracking. For any point on the trace, *tracking uncertainty* is defined as $\mathcal{H}=-\sum p_i\times\log_2 p_i$, where p_i denotes the probability that location sample i belongs to the vehicle currently tracked. Lower values of $\mathcal{H}$ indicate more certainty or lower privacy. The tracking confidence $\mathcal{C}$ on attacker's trial can be calculated as $(1-\mathcal{H})$. This privacy metrics is also used in [14, 24].

For each identifying location model proposed, we measure the degree of privacy as the time that an attacker can correctly follow a trace, i.e., the trend that the attacker's uncertainty $\mathcal{H}$ goes with the user's online time t. For a given user, we have a sequence of time stamps $\mathcal{T}=\{t_0,t_1,\ldots,t_n\}$, and at each time stamp $t_i\in\mathcal{T}$, this user sends a query to the server.

In model $\mathcal{A}$, the identifying locations are simulated as designated locations. Suppose a user u locates at an identifying location at time t_m, the attacker is able to identify u, and hence the location entropy for this user $\mathcal{H}_u(t_m)=0$. Starting from this identifying location, u will be tracked until he/she crosses his/her path with another user v at time t_n. At t_n, the location entropy for both u and v increases. In this example, we have:

$$\mathcal{H}_u(t_i)=\begin{cases}0, & \text{if } m\leqslant i<n,\\ 1, & \text{if } i=n.\end{cases}$$

In model $\mathcal{B}$, we choose a random query of a user u generated at time t_r, and assume the track uncertainty $\mathcal{H}_u(t_r)=0$. User u will be tracked until he/she crosses

his/her path with another user v at time t_m. At t_n, the location entropy for both u and v increases, and $\mathcal{H}_u(t_n) = 1$. Starting from t_n, the track uncertainty increases with every query sent by u until u crosses his/her path with some other user again at time t_n. In this example, we have:

$$\mathcal{H}_u(t_i) = \begin{cases} 0, & \text{if } r \leqslant i < m, \\ 1, & \text{if } i = m, \\ -\log_2 0.5 \times (1-p)^{i-m}, & \text{if } m < i < n. \end{cases}$$

The probabilities of identifying a user based on different numbers of queries are shown in Fig.12.

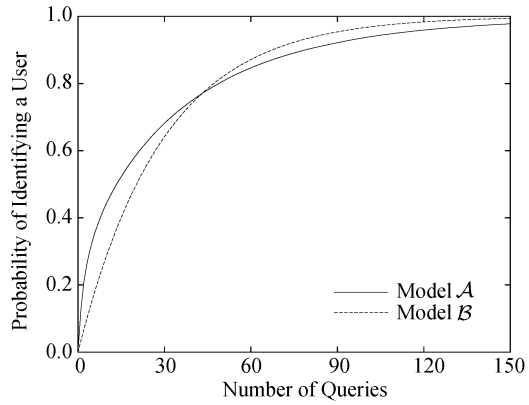


Fig.12. Identification attacker model.

Results and Analysis. For each user in the dataset, we calculate the location entropy using both attacker models. The results comparing CBA with the path confusion approach are shown in Fig.13. In all the cases, CBA has much better privacy performance than the path confusion approach. In the worst case, when the user density is very low (shown in Fig.13(b)), CBA achieves certain anonymity degree ($\mathcal{H} \approx 2$) while the path confusion approach cannot provide any protection.

5.2 Evaluation of the Key Exchange Protocol

CBA uses a symmetric key exchange protocol to encrypt information exchanged between nearby users. The time consumed in authentication may affect the performance (i.e., a long period of time in the authentication phase may cause nearby users unable to confuse their paths in time before they move away). We evaluate the time cost in this procedure.

In this experiment, we have three volunteers, each carrying a SAMSUNG Galaxy Nexus smartphone with Android 4.2.1 installed, moving around in a campus area. Their locations are recorded every 10 seconds, and reported to a confusion server. Whenever two of them are coming close to each other (i.e., within 200 meters), the server notifies both devices. Then each device uses Wi-Fi Direct to discover each other for authentication, and records the time consumed in this phase. This experiment lasts for two hours.

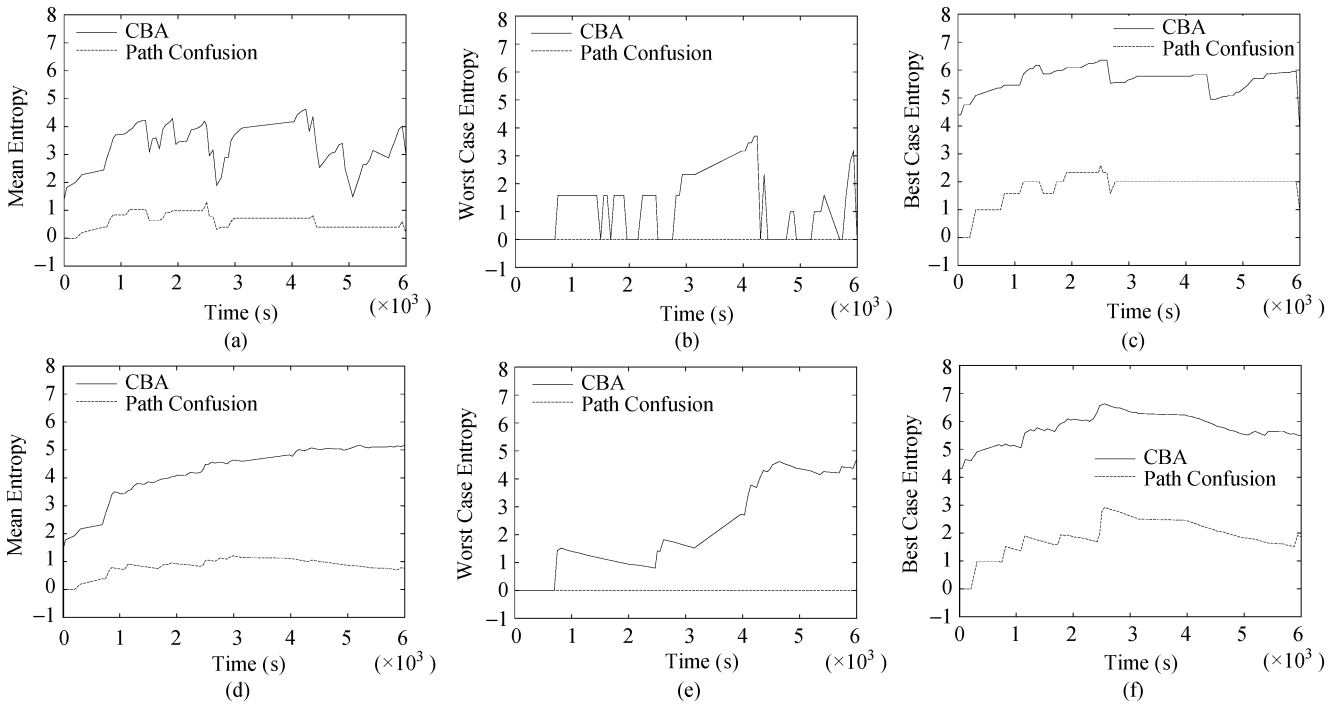


Fig.13. Privacy entropy in different cases in both models: comparison between CBA and path confusion. (a) Mean entropy over time in model $\mathcal{A}$. (b) Worst case entropy in model $\mathcal{A}$. (c) Best case entropy in model $\mathcal{A}$. (d) Mean entropy over time in model $\mathcal{B}$. (e) Worst case entropy in model $\mathcal{B}$. (f) Best case entropy in model $\mathcal{B}$.

Figs. 14 and 15 show the results within the distance range of 130 meters. The authentication works fine with a distance of 120 meters, but the failure rate increases to 0.4 with a distance of 130 meters. The authentication phase can be divided into two steps — the Wi-Fi discovery step and the setup step. The Wi-Fi discovery step in the authentication phase is more time consuming. It requires 2 to 12 seconds depending on the distance between two smartphones, as shown in Fig.14(a). The setup step requires only 0.1 to 0.12 seconds and does not depend on the distance, as shown in Fig.14(b). Next, we focus our analysis on the Wi-Fi discovery step.

By analyzing different instances of records, we find that most discoveries last for 2 to 3 seconds. However there are cases where the discovery lasts for more than 10 seconds or even longer. These records increase the time consumed, and the proportion of such records increases with distance. We use a simple quit strategy to avoid users waiting for too long within a single discovery — if no peer is found after starting a discovery for 4 seconds, Wi-Fi is reset and the discovery is re-initiated. Using this strategy, the average time consumed is reduced to 3 seconds as shown in Fig.15(a). Note that the real line represents the average time consumed, where the time wasted in these instances is taken into account. The quit rate increases with distance as shown in Fig.15(b). When the distance is between 120 and 130 meters, the quit rate is between 0.7 and 0.8.

5.3 Evaluation of Prediction Engine

We use an open source context simulator Siafu④ to evaluate our prediction engine under realistic conditions. The reason that we use a simulator to generate data instead of using real traces like the cabspotting dataset[23] is that our prediction engine requires the users' orientation to an input, which cannot be found in most real traces dataset. On the other hand, the simulator Siafu provides all the moving pattern information of the simulated users including location, time, moving direction, destination (orientation), speed, etc.

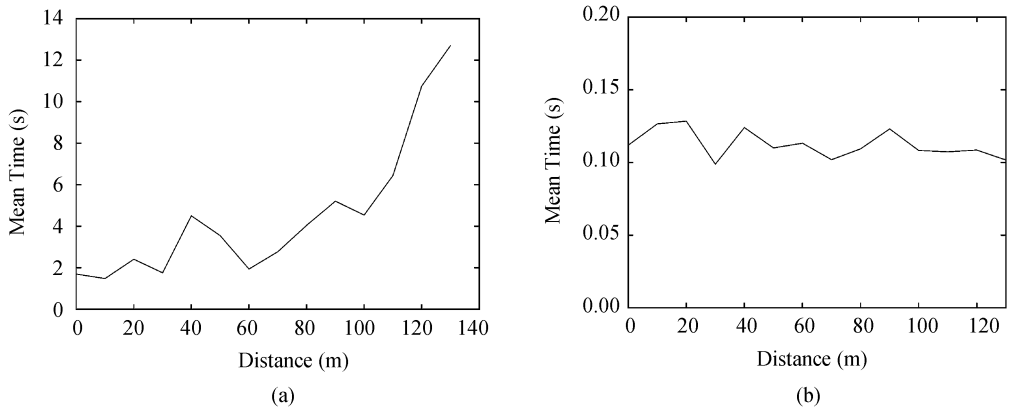


Fig.14. Time consumed in the authentication phase. (a) Average time consumed in the Wi-Fi discovery step with different distances. (b) Average time consumed in the setup step (setup and other operations).

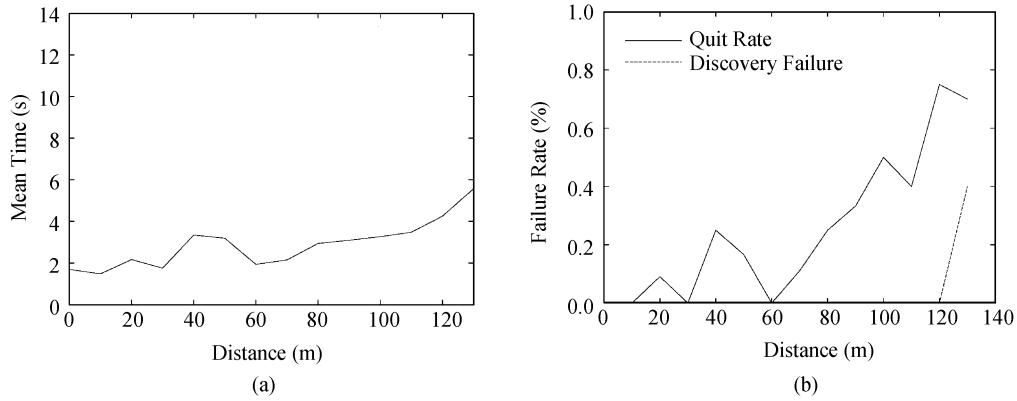


Fig.15. Time consumed in the authentication phase by using the quit strategy. (a) Average time consumed in the Wi-Fi discovery step with different distances using the quit strategy. (b) Discovery failure rate and quit rate using the quit strategy.

④http://siafusimulator.sourceforge.net/, Sept. 2014.

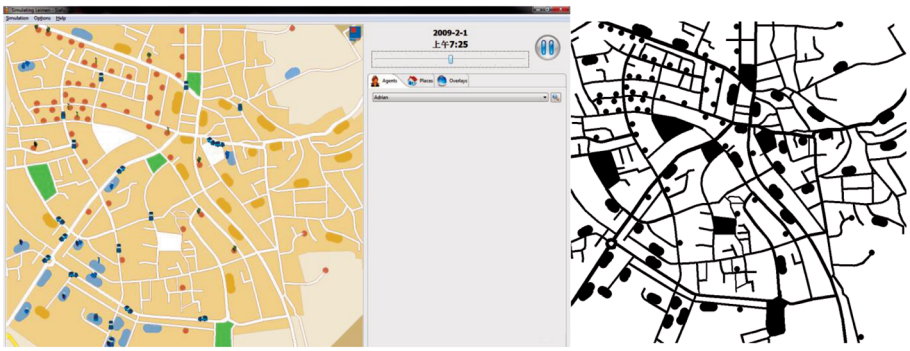


Fig.16. Context simulator Siafu and the city of Leimen.

Our simulation is based on an existing simulation scenario of Leimen[⑤]. In this scenario, people in Leimen lead a simple life. They wake up in the morning, go to office for work (by walk or car), then go back home, or go for parties after work. The same pattern repeats the next day. In our evaluation, we focus on the people who are driving, and treat them as potential users of CBA. The simulation runs continuously, and we record the location information and the moving pattern information for all the users (i.e., cars) for every 10 seconds, and mark each record with the user's name. The trace is recorded, and then loaded into our prediction engine, simulating a real-time stream of location updates from users.

We focus on cases in which the real traces of two users are close to each other, i.e., we can find a pair of records, each from a different trace, where the spatial distance dis_{S} is below a threshold θ_{S}, and the temporal distance dis_{T} is below a threshold θ_{T}. Since the recording interval is 10 seconds, we set θ_{T} to 10 seconds. Assuming different θ_{S}, we randomly select 1 000 cases, and use our prediction engine to calculate the CBA zone. After we get the CBA zone area, we match it to the Leimen map as shown in Fig.16 and obtain the POIs (as we can do by using the Google Maps API) on the border of the CBA zone as the candidate exit points of the CBA zone. The results are shown in Fig.17. Our prediction engine has a success rate of more than 82% on predicting the border of the CBA zone (a successful prediction means that the user goes through the joint zone), and has a success rate of more than 93% on predicting the candidate exit points (a successful prediction means that one of the candidate exit points is the real exit point). The overall success rate of our prediction engine is above 78%.

5.4 Evaluation of Cloaked Obfuscation

As discussed in Subsection 4.4, under a trace identification attack, an attacker distinguishes fake traces from

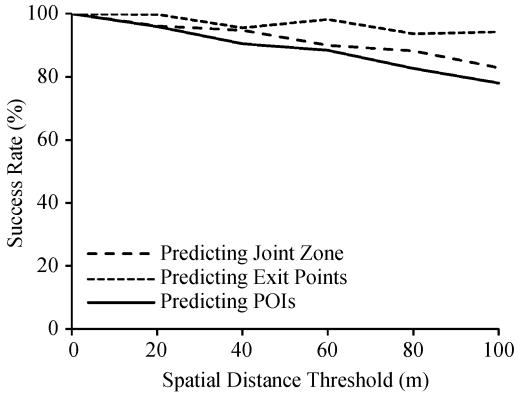


Fig.17. Success rate of the prediction engine.

user move patterns. In CBA, users access LBS with only real locations when they are out of CBA zones. In case that the LSP is untrustworthy, an attacker may know a user's historical location information from the LSP, and can perform a classification attack to identify this user. In this subsection, we demonstrate that the security of CBA reduces to that of the mix zone approach.

Security Assumption of Identifying Mix Zones. Let $u, v \in \mathcal{U}$ be chosen at random in a mix zone $\mathcal{Z}$, $\mathcal{T}_1$, $\mathcal{T}_2$ be the traces of them after they pass $\mathcal{Z}$. We define the security assumption of identifying mix zones as that no machine learning algorithm $\mathcal{M}$ can distinguish u's trace $\mathcal{T}_u$ from $\mathcal{T}_1$ and $\mathcal{T}_2$ with non-negligible advantage. The advantage of $\mathcal{M}$ is $|P(\mathcal{T}_u = \mathcal{T}_1) - P(\mathcal{T}_u = \mathcal{T}_2)|$.

Attacker Model. To simulate a classification attack, suppose an attacker knows all the information the LSP knows, the attacker may generate fake queries based on historical information. Then he/she chooses a classifier and trains the classifier with both the user data and the fake query data. If a classifier $\mathcal{C}$ has a classification accuracy $\mathcal{P}_C$, the probability of distinguishing a fake trace in a CBA zone is $\mathcal{P}_C$.

Privacy Analysis. We now conduct experiments to analyze if CBA is secure under the classification attack.

⑤ http://siafusimulator.sourceforge.net/?what=simulations&simtitle=leimen/, Sept. 2014.

Similar to the privacy metrics in Subsection 5.1.3, we use threshold $\mathcal{H}$ to indicate privacy entropy, which is defined as follows.

$$\mathcal{H} = -\sum P \times \log_2 P = -\log_2 \mathcal{P}_C.$$

We use various classifiers available in Weka, such as naive bayes, support vector machines, AD trees, J48 trees. Using these classifiers, the probability of identifying the real traces ranges from 0.5 to 0.57 when $k = 2$. To this aspect, CBA is secure. Proving the security of CBA is constrained by the classifier used. Instead we prove that the security of CBA in the above attacker model can be reduced to the hardness of the security assumption of identifying mix zones.

Proof of Security. Suppose there exists a machine learning algorithm $\mathcal{N}$ that can distinguish users u, v who pass the same CBA zone with more than negligible advantage σ. For the same users u, v, assume the CBA zone to be a mix zone, we have their traces before they enter this mix zone and the traces after they leave this mix zone. We use the algorithm $\mathcal{F}_{\text{TraceGen}}$ (described in Subsection 4.4) to generate four traces connecting their entries and their exits to this mix zone. According to Definition 1, the mix zone with these four traces is a CBA zone. Then we have composition algorithm $\mathcal{M} = C_{\mathcal{F}_{\text{TraceGen}}}\mathcal{N}$, which can distinguish u's trace $\mathcal{T}_u$ from $\mathcal{T}_1$ and $\mathcal{T}_2$ with more than a negligible advantage σ. □

Discussion on Location Accuracy. Querying LBS with only fake locations decreases QoS. We conduct experiments on this factor, and the results are shown in Fig.18. We observe that even in the worst case, the distance between the predicted queries and the real queries is less than 10 meters, which is similar to the accuracy provided by commercial GPS. We believe that the price is reasonable to trade off privacy.

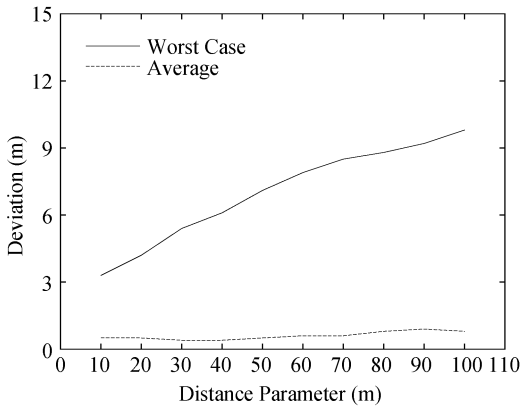


Fig.18. Distance between the predicted queries and the real queries.

6 Conclusions

This paper presents a novel CBA scheme to balance user privacy with the functionality for location-based services. We proposed an collaborative path confusion mechanism for nearby users to generate a CBA zone, the local prediction algorithm for each user to generate the obfuscating traces for path confusion, and the cloaked obfuscation method to prevent the trace identification attacks. Using a real-world dataset, we demonstrated that CBA outperforms the path confusion approach and the query obfuscation approach in terms of robustness and resources consumed.

We have proven that the security of CBA reduces to that of the mix zone approach. However, temporal and spatial relations exist between a user's traces before entering a CBA zone and after leaving a CBA zone. By analyzing such relations, an attacker may still be able to launch attacks using advanced machine learning techniques, resulting in the decreases in the anonymity degree of CBA. For our future work, we plan to improve the robustness of CBA against such attack. To address this problem, users can apply the same cloaked obfuscation method to remove the characteristics of real user move patterns along his/her trace outside CBA zones.

References

[1] Gruteser M, Grunwald D. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. the 1st International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*, May 2003, pp.31-42.

[2] Gedik B, Liu L. Location privacy in mobile systems: A personalized anonymization model. In *Proc. the 25th International Conference on Distributed Computing Systems* (*ICDCS 2005*), June 2005, pp.620-629.

[3] Mokbel M, Chow C, Aref W. The new Casper: Query processing for location services without compromising privacy. In *Proc. the 32nd International Conference on Very Large Data Bases* (*VLDB 2006*), Sept. 2006, pp.763-774.

[4] Kalnis P, Ghinita G, Mouratidis K, Papadias D. Preventing location-based identity inference in anonymous spatial queries. *IEEE Transactions on Knowledge and Data Engineering* (*TKDE*), 2007, 19(12): 1719-1733.

[5] Beresford A, Stajano F. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2003, 2(1): 46-55.

[6] Hoh B, Gruteser M. Protecting location privacy through path confusion. In *Proc. the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks* (*SECURECOMM 2005*), September 2005, pp.194-205.

[7] Hoh B, Gruteser M, Xiong H, Alrabady A. Preserving privacy in GPS traces via uncertainty-aware path cloaking. In *Proc. the 14th International Conference on Computer and Communications Security* (*CCS 2007*), October 29-November 2, 2007, pp.161-171.

[8] Palanisamy B, Liu L. MobiMix: Protecting location privacy with mix-zones over road networks. In *Proc. the 27th International Conference on Data Engineering* (*ICDE 2011*), April 2011, pp.494-505.

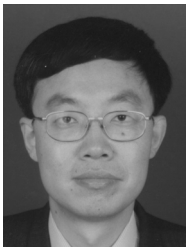
[9] Zheng Y, Chen Y, Xie X, Ma W. Geolife2.0: A location-based

social networking service. In *Proc. the 10th International Conference on Mobile Data Management: Systems, Services and Middleware* (*MDM 2009*), May 2009, pp.357-358.
[10] Dong K, Gu T, Tao X, Lu J. Complete bipartite anonymity: Confusing anonymous mobility traces for location privacy. In *Proc. the 18th International Conference on Parallel and Distributed Systems* (*ICPADS 2012*), December 2012, pp.205-212.
[11] Peddinti S, Saxena N. On the limitations of query obfuscation techniques for location privacy. In *Proc. the 13th International Conference on Ubiquitous Computing* (*UbiComp 2011*), September 2011, pp.187-196.
[12] Machanavajjhala A, Gehrke J, Götz M. Data publishing against realistic adversaries. *Proc. the VLDB Endowment*, 2009, 2(1): 790-801.
[13] Goldschlag D, Reed M, Syverson P. Onion routing. *Communications of the ACM*, 1999, 42(2): 39-41.
[14] Meyerowitz J, Choudhury R. Hiding stars with fireworks: Location privacy through camouflage. In *Proc. the 15th Annual International Conference on Mobile Computing and Networking* (*MobiCom 2009*), September 2009, pp.345-356.
[15] Sweeney L. *k*-anonymity: A model for protecting privacy. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, 2002, 10(5): 557-570.
[16] Sweeney L. Achieving *k*-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems*, 2002, 10(5): 571-588.
[17] Hashem T, Kulik L. "Don't trust anyone": Privacy protection for location-based services. *Pervasive and Mobile Computing*, 2011, 7(1): 44-59.
[18] Shokri R, Papadimitratos P, Theodorakopoulos G, Hubaux J. Collaborative location privacy. In *Proc. the 8th International Conference on Mobile Adhoc and Sensor Systems* (*MASS 2011*), Oct. 2011, pp.500-509.
[19] Christin D, Guillemet J, Reinhardt A, Hollick M, Kanhere S. Privacy-preserving collaborative path hiding for participatory sensing applications. In *Proc. the 8th International Conference on Mobile Adhoc and Sensor Systems* (*MASS 2011*), Oct. 2011, pp.341-350.
[20] Kido H, Yanagisawa Y, Satoh T. An anonymous communication technique using dummies for location-based services. In *Proc. the 3rd International Conference on Pervasive Services* (*ICPS 2005*), July 2005, pp.88-97.
[21] Krumm J. Realistic driving trips for location privacy. In *Proc. the 7th International Conference on Pervasive Computing*, March 2009, pp.25-41.
[22] Shankar P, Ganapathy V, Iftode L. Privately querying location-based services with SybilQuery. In *Proc. the 11th International Conference on Ubiquitous Computing* (*UbiComp 2009*), September 30-October 3, 2009, pp.31-40.
[23] Piorkowski M, Sarafijanovoc-Djukic N, Grossglauser M. A parsimonious model of mobile partitioned networks with clustering. In *Proc. the 1st International Conference on Communication Systems and Networks* (*COMSNETS 2009*), Jan. 2009, pp.1-10.
[24] Bindschaedler L, Jadliwala M, Bilogrevic I, Aad I, Ginzboorg P, Niemi V, Hubaux JP. Track me if you can: On the effectiveness of context-based identifier changes in deployed mobile networks. In *Proc. the 19th Network and Distributed System Security Symposium (NDSS 2012)*, February 2012.

Kai Dong received his B.S. and M.S. degrees in computer science from Nanjing University in 2007 and 2010, respectively. He is currently a Ph.D. candidate in the Department of Computer Science at Nanjing University. His research interests include privacy preservation, mobile and pervasive computing.

Tao Gu received his B.S. degree from Huazhong University of Science and Technology, and M.S. degree from Nanyang Technological University, Singapore, and Ph.D. degree in computer science from National University of Singapore. He is currently an associate professor in the School of Computer Science and Information Technology at RMIT University, Melbourne. His research interests include mobile and pervasive computing, wireless sensor networks, distributed network systems, sensor data analytics, cyber physical system, Internet of Things, and online social networks. He is a senior member of IEEE and a member of ACM.

Xian-Ping Tao received his M.S. and Ph.D. degrees in computer science from Nanjing University in 1994 and 2001, respectively. He is currently a professor in the Department of Computer Science at Nanjing University. His research interests include software agent, middleware system, Internetware methodology, and pervasive computing. He is a member of CCF and IEEE.



Jian Lv received his B.S., M.S., and Ph.D. degrees in computer science from Nanjing University in 1982, 1984, and 1988, respectively. He is currently a professor in the Department of Computer Science at Nanjing University. He is also the director of the State Key Laboratory for Novel Software Technology and the vice director of the Institute of Software Technology at Nanjing University. His research interests include programming methodology, pervasive computing, software agent, and middleware. He is a fellow of CCF and a member of ACM.