Geoinformatica
DOI 10.1007/s10707-013-0192-0

Protecting query privacy in location-based services

Xihui Chen · Jun Pang

Received: 30 May 2013 / Revised: 28 August 2013 / Accepted: 24 September 2013

Abstract The popularity of location-based services (LBSs) leads to severe concerns on users' privacy. With the fast growth of Internet applications such as online social networks, more user information becomes available to the attackers, which allows them to construct new contextual information. This gives rise to new challenges for user privacy protection and often requires improvements on the existing privacy-preserving methods. In this paper, we classify contextual information related to LBS query privacy and focus on two types of contexts—*user profiles* and *query dependency*: user profiles have not been deeply studied in LBS query privacy protection, while we are the first to show the impact of query dependency on users' query privacy. More specifically, we present a general framework to enable the attackers to compute a distribution on users with respect to issuing an observed request. The framework can model attackers with different contextual information. We take user profiles and query dependency as examples to illustrate the implementation of the framework and their impact on users' query privacy. Our framework subsequently allows us to show the insufficiency of existing query privacy metrics, e.g., *k*-anonymity, and propose several new metrics. In the end, we develop new generalisation algorithms to compute regions satisfying users' privacy requirements

This article is a revised and extended version of our two conference papers [7, 8].

Xihui Chen is supported by an AFR PhD grant from the National Research Fund, Luxembourg.

X. Chen (✉)
Interdisciplinary Centre for Security, Reliability and Trust,
University of Luxembourg,
6 rue Richard Coudenhove-Kalergi, 1359 Luxembourg, Luxembourg
e-mail: xihui.chen@uni.lu

J. Pang
Faculty of Science, Technology and Communication,
University of Luxembourg,
6 rue Richard Coudenhove-Kalergi, 1359 Luxembourg, Luxembourg
e-mail: jun.pang@uni.lu

Published online: 18 October 2013

expressed in these metrics. By experiments, our metrics and algorithms are shown to be effective and efficient for practical usage.

Keywords Location-based services · Query privacy · Anonymity · Measurement

1 Introduction

Location-based services (LBSs) are services customised according to users' locations. In the last fifteen years, LBSs have endured a great growth, especially after GPS-enabled devices such as smartphones became popular. An LBS request contains the issuer's location and a *query*—the type of information of interest, for instance, 'where the nearest gas stations are'. In spite of the great convenience brought to users' daily life, LBSs lead to severe privacy concerns especially in cases where LBS providers are not considered trustworthy and share users' request with the attackers. In the literature, two major privacy concerns in LBSs have been studied—*location privacy* and *query privacy* [22]—in terms of the types of sensitive information. The former is related to the disclosure of users' exact locations while query privacy, the focus of our paper, concerns the disclosure of queries.

The basic idea to protect users' query privacy in LBSs is to break the link between user identities and requests [4]. However, in the context of LBSs, only removing or replacing identities with pseudonyms has been proved insufficient. Locations contained in requests can still reveal issuers' identities, since attackers can acquire users' locations through a number of methods, e.g., triangulating mobile phones' signals and localising users' access points to the Internet. In such cases, users' spatial and temporal information serves as *quasi-identifiers*. Anonymisation techniques from other research areas such as sensitive data release [22] are thus introduced, including k-anonymity and its different extensions (e.g., ℓ-diversity and t-closeness [29, 30]). Locations or time are replaced with regions or periods so that a certain number of users (at least k for k-anonymity) share the same quasi-identifier with the real issuer. The calculation of the regions or periods is termed as *generalisation* or *cloaking*. Since in practice LBS providers are usually required to offer immediate responses, throughout the paper, temporal generalisation is out of the scope. We call a request *generalised* if the location is generalised and the user identity is removed.

However, when the adversary has access to additional information, new privacy risks will emerge. For instance, "outlier" attacks are found on some existing generalisation algorithms when their implementation is made public [33]. Some users are eliminated from the set of potential issuers as the algorithms cannot output the same generalised region to all of them when they issue the request. Such information is classified as *contextual information* in the literature and the privacy in LBSs related to contextual information has been recognised as *context-aware privacy* [37]. Many types of contextual information have been studied so far. For example, Shin et al. [42, 43] study user profiles and propose metrics based on k-anonymity by restricting levels of similarity among users in terms of their profiles. Cheng et al. [10] propose the concept of *historical k-anonymity* against attacks where the adversary learns a trace of associated requests, e.g., issued by the same user.

Our motivations The research on context-aware privacy usually follows a two-step approach. It starts with identifying a type of contextual information and demonstrating its impact on users' privacy and then it proceeds with developing specific privacy protection mechanisms. There are a few restrictions with this line of research. First, the privacy concern of contextual information is usually illustrated in a possibilistic way with a focus on whether the impact exists or not. In spite of the claim that attackers can learn more information about issuers, it is not clear what changes have been exactly made on the attackers' knowledge by this piece of contextual information. Second, variants of contextual information are studied independently. Although new privacy properties are defined and based on them privacy protection mechanisms are proposed, they are only effective for the identified contextual information. In particular, with the fast development of information and communication techniques, new contextual information will always be identified. We need a framework to uniformly capture contextual information. Moreover, we need to define new generic privacy metrics for users to express their privacy requirements precisely and design (generalisation) algorithms to support such requirements.

Our contributions We summarise our main contributions in this paper as follows to address the above identified research questions:

1. We develop a formal framework for context-aware query privacy analysis. In particular, this framework gives a probabilistic interpretation of the attacks on query privacy. With this interpretation, we show how query privacy can be breached form the adversary's viewpoint. We take the contextual information—*user profiles* as an example to illustrate the instantiation of our framework.
2. Within the formal framework, we propose a systematic classification of contextual information related to query privacy, according to whether the information changes over time—*static* and *dynamic*.
3. Along with the development of LBSs, new contextual information can be collected and abused by the attackers. We identify a new type of contextual information—*query dependency* and study its impact on query privacy within our framework.
4. Our probabilistic framework allows us to define several new metrics to measure users' query privacy from different perspectives. Moreover, these metrics enable users to express their privacy requirements effectively. In order to compute generalised regions that satisfy users' requirements, we propose new generalisation algorithms which are efficient and can ensure a good quality of the responses from the LBS providers.
5. Through experiments, we show that our framework can significantly improve the adversary's analysis of users' query privacy. In particular, we validate that query dependency can effectively reduce the uncertainty of the adversary about the real issuer of a request. Furthermore, we show that our generalisation algorithms are efficient to meet the demands on real-time responses and can generate regions satisfying privacy requirements when various contextual information are considered. We also show that the different strengths of our metrics can help a user choose a right metric to achieve a balance between privacy and the quality of service delivered by different LBS providers.

Structure of the paper In Section 2, we discuss the related work. We define our formal framework and the adversary model in Section 3. In Section 4 we take user profiles as an example of contextual information to show the application of our framework. We formally study the impact of query dependency on query privacy in Section 5 by incorporating it into our framework. Our query privacy metrics are defined in Section 6 and our new generalisation algorithms are presented in Section 7. Section 8 summarises the experimental results and the corresponding analysis. We conclude the paper in Section 9.

2 Related work

2.1 Query privacy and request generalisation

Protecting users' query privacy is essentially to prevent the adversary from learning their issued queries. A number of techniques have been proposed in the literature to protect query privacy and they can be classified into three main groups—*obfuscation* [28, 49], *generalisation* [22, 48] and *cryptographic transformation* [17]. The methods of obfuscation forge dummy requests with different queries such that the real queries are hidden in the dummy ones. The idea of generalisation is to hide the real issuer in a number of users such that he is indistinguishable from the others from the view of the adversary. The concept of k-anonymity has been extensively studied in this group. In the methods exploring cryptographic transformation, users' queries are encrypted and remain secret for LBS providers so as to offer strong privacy protection. All of these methods introduce extra processing overhead. For instance, Ghinita et al. [17] build a protocol based on computational private information retrieval (cPIR)—their protocol requires one extra round of communication between users and LBS providers and imposes additional computation overheads on both sides due to the encryption of queries and decryption of responses. In this paper we focus on request generalisation and use it to protect query privacy with respect to contextual information.

The notion of k-anonymity was originally proposed by Samarati and Sweeney in the field of database privacy [39]. The idea of k-anonymity is to guarantee that a database entry's identifier is indistinguishable from other $k-1$ entries. However, this method does not always work. For instance, the fact that an HIV carrier is hidden in k carriers does not protect his infection of the virus. Further research has been done to fix this problem [29]. In the context of privacy in LBSs, k-anonymity was first studied by Gruteser and Grunwald [22]. Its purpose is to compute a region containing at least other $k-1$ users (i.e., *area generalisation*) and replace the issuer's location with it. Because of its simplicity, k-anonymity has been studied and refined in many ways. For instance, Tan et al. define information leakage to measure the amount of revealed location information in spatial cloaking, which quantifies the balance between privacy and performance [46]. Xue et al. [48] introduce the concept of *location diversity* to ensure generalised regions to contain at least ℓ semantic locations (e.g., schools). However, deeper understanding of k-anonymity reveals its drawbacks in preserving location privacy. Shokri et al. analyse the effectiveness of k-anonymity in protecting location privacy in different scenarios in terms of adversaries' background information [45], i.e., *real-time location information*, *statistical information*

Fig. 1 A centralised framework of LBSs

and *no information*. They show its flaws which the adversary can exploit to infer users' current locations and conclude that spatial cloaking (e.g., *k*-anonymity) is only effective for protecting query privacy. As a consequence, in this work we use area generalisation *only* to protect query privacy.

The generalisation of LBS requests is usually implemented in two ways—*centralised* and *distributed*. A centralised structure (depicted in Fig. 1) relies on a trusted agent, the *anonymiser*, to collect users' requests and anonymise them before sending them to LBS providers. However, in a distributed implementation users cooperate with each other to construct a generalised region [18, 40]. The centralised framework is easy to implement and well-studied in the literature while the distributed framework requires more communications between collaborators and security analysis, e.g., with respect to *insiders*, is not well studied. In the centralised framework, normally it is assumed that the communication channels between users and the anonymiser are secure while the ones between the anonymiser and the LBS provider are public.

2.2 Context-aware privacy analysis

The effectiveness of area generalisation can be compromised when the adversary has access to auxiliary contextual information. In fact, area generalisation guaranteeing *k*-anonymity is proposed to protect query privacy against the adversary who has users' real-time locations in their knowledge. Mascetti et al. [33] identify the 'outlier' attack on some generalisation algorithms if the adversary learns their implementations. *k*-anonymity is violated because the algorithms cannot ensure that all the potential issuers have the same generalised area as the real issuer. Shokri et al. [44] use mobility patterns modelled as Markov chains of location transition and propose a probabilistic framework to de-anonymise generalised traces. Personal information (e.g., gender, job, salary) has been becoming more accessible on the Internet, e.g., due to online social networks such as Facebook and LinkedIn, and can also serve as a type of contextual information which we call *user profiles*. Shin et al. [42, 43] identify the concern of query privacy caused by user profiles and propose metrics based on *k*-anonymity by restricting similarity levels between users in terms of their profiles.

The contextual information (e.g., user profiles and generalisation algorithms) mentioned above is irrelevant to users' past LBS requests. Actually LBS requests can also be explored by the adversary to refine his guess on the issuers. Two types of LBS requests have been studied in the literature—*associated requests* [4, 10, 13] and *recurrent requests* [38]. Requests are associated once they are recognised as issued by a same (anonymous) user, which can be achieved for example by

multi-target tracking techniques [24] or probabilistic reasoning [44]. By calculating the intersection of all associated requests' anonymity sets the adversary can reduce the number of possible issuers. To handle such privacy threats, Cheng et al. [10], Bettini et al. [4] introduce *historical k-anonymity*, which is then extended for continuous LBSs by Dewri et al. [13]. Historical k-anonymity aims to guarantee that associated requests share at least k fixed users in the generalised regions. Requests are recurrent when they are issued at the same time. When multiple recurrent requests contain the same query and region, the protection of query privacy offered by spatial cloaking (e.g., k-anonymity) will be degraded [38]. For instance, in the extreme case, when all users in a region send an identical query, no user has query privacy. Riboni et al. [38] identify the threat and make use of t-closeness to guarantee that the distance between the distribution over the queries from an issuer's generalised region and that of the whole region is below a threshold. Dewri et al. [14] identify a scenario in continuous LBSs which has both associated and recurrent requests. They propose m-invariance to ensure that in addition to k fixed users shared by the associated requests, at least m different queries are generated from each generalised region.

2.3 Area generalisation algorithms

The first generalisation algorithm called IntervalCloaking is designed by Gruteser and Grunwald [22]. Their idea is to partition a region into quadrants with equal area. If the quadrant where the issuer is located contains less than k users, then the original region is returned. Otherwise, the quadrant with the issuer is taken as input for the next iteration. The algorithm CliqueCloak [16] is proposed by Gedik and Liu in which regions are generalised based on the users who have issued queries rather than all potential issuers. The major improvement is that this algorithm enables users to specify their personal privacy requirements by choosing different values for k. Mokbel et al. [11, 34] design the algorithm Casper which employs a quadtree to store the two-dimensional space. The root node represents the whole area and each of other nodes represent a quadrant region of its parent node. The generalisation algorithm starts from the leaf node which contains the issuer and iteratively traverses backwards to the root until a region with more than k users is found. Another algorithm nnASR [27] simply finds the nearest k users to the issuer and returns the region containing these users as the anonymising spatial region.

The above algorithms suffer from a particular attack called "outlier problem" [3], where the attackers have the generalisation algorithms and users' spatial distribution as part of their knowledge. An algorithm against this attack needs to ensure that for any user in the anonymity set it returns the same region. Kalnis et al. design the first algorithm called hilbASR that does not suffer from the outlier problem [27]. The algorithm exploits the Hilbert space filling curve to store users in a total order based on their locations. The curve is then partitioned into blocks with k users. The block with the issuer is returned as the generalised region. Mascetti et al. propose two algorithms, dichotomicPoints and grid, which are also secure against the outlier problem [33]. The former iteratively partitions the region into two blocks until less than $2k$ users are located in the region while the latter draws a grid over the two-dimensional space so that each cell contains k users and returns the cell with the issuer. Because of the simplicity of implementation and the relatively smaller area of

the generalised regions, we adopt and extend these two algorithms in our algorithm design (see Section 7). The area of generalised regions is usually used to measure the quality of the LBS response, as smaller regions lead to more accurate results and less communication overhead.

3 Our framework

In this section, we present our framework for query privacy analysis in LBSs. This framework allows us to precisely specify relevant components and attacks on query privacy with various contextual information.

3.1 Mobile users

We assume a set of users who subscribe an LBS and use the service frequently during their movements. Let $\mathcal{U}$ be the set of such users. We use $\mathcal{L}$ to denote the set of all possible positions where a user can issue a request. The accuracy of any position $\ell \in \mathcal{L}$ is determined by the positioning devices used. We represent time as a totally ordered discrete set $\mathcal{T}$, whose granularity, e.g., minutes or seconds, is decided by the LBS provider. The function $whereis : \mathcal{U} \times \mathcal{T} \rightarrow \mathcal{L}$ gives the exact position of a user at a given time. Thus, for any time $t \in \mathcal{T}$, users' *spatial distribution* is $dis_t = \{\langle u, whereis(u, t)\rangle | u \in \mathcal{U}\}$. Suppose the set of queries (e.g., the nearest gas station) supported by LBS providers is represented by $\mathcal{Q}$. An LBS request is then in the form of $\langle u, \ell, t, q\rangle \in \mathcal{U} \times \mathcal{L} \times \mathcal{T} \times \mathcal{Q}$, where $\ell = whereris(u, t)$.

3.2 Request generalisation algorithms

Given a request $\langle u, \ell, t, q\rangle$, the anonymising server (*anonymiser*) will remove the issuer's identity (i.e., u) and replace his location (i.e., ℓ) with an area to protect his query privacy. We only consider *spatial generalisation* in this paper as in LBSs users require instant responses. Let $2^{\mathcal{L}}$ be the power set of $\mathcal{L}$ and then the set of all possible generalised regions can be denoted by $\mathcal{R} \subseteq 2^{\mathcal{L}}$. Given $\langle u, \ell, t, q\rangle$, the anonymising server outputs a *generalised request* in the form of $\langle r, t, q\rangle$, where $r \in \mathcal{R}$ is the generalised area and $\ell \in r$. The generalisation algorithm of the anonymiser can thus be represented as a function $f : \mathcal{U} \times \mathcal{L} \times T \times \mathcal{Q} \rightarrow \mathcal{R} \times \mathcal{T} \times \mathcal{Q}$. We use function *query* to obtain the query of a (generalised) request (i.e., $query(\langle u, \ell, t, q\rangle) = q$ and $query(\langle r, t, q\rangle) = q$).

The generalisation algorithm also takes users' privacy requirements as part of its input. In our framework, a privacy requirement is represented by a pair—a chosen privacy metric and the corresponding value (e.g., $\langle \mathsf{k-anonymity}, 5\rangle$). We use $req(\langle u, \ell, t, q\rangle)$ to represent u's privacy requirement on request $\langle u, \ell, t, q\rangle$.

3.3 The adversary

Privacy risks and countermeasures should be categorised according to the adversary's model and goals [4]. For query privacy, the adversary's goal is obviously to associate issuers to their queries while the model should be defined in terms of his *knowledge* and *attack(s)* [43].

The knowledge of an adversary can be interpreted as the contextual information that he has access to. We denote by $\mathcal{C}_t$ his collection of contextual information at time t. In this paper, we assume that some contextual information is inherently contained in $\mathcal{C}_t$.

The adversary knows the deployed request generalisation algorithm (i.e., f) and users' spatial distributions before the current time t (i.e., $\mathcal{D}_t = (dis_{t_1}, \ldots, dis_{t_n})$ where $t_n = t$ and $\forall_{1 \le i < n} t_i < t_{i+1}$). This is a common assumption used in the literature and it makes a strong adversary which allows us to analyse query privacy in the worst cases. The availability of dis_t enables the adversary to obtain the set of users located in any region r at time t, which is denoted as $u\ell(r, t)$.

Given a generalised request $\langle r, t, q\rangle$ and $\mathcal{C}_t$, the objective of an attack performed by the adversary on query privacy is to track the request's issuer. In most of the cases, it is not practical for the adversary to be completely sure of the issuer. Uncertainty is thus inevitable. We use a *posterior probability distribution* over users to capture his certainty and quantify the expected correctness of his attack. Let variable U be the issuer of $\langle r, t, q\rangle$. For any user $u \in \mathcal{U}$, his probability to issue the request $\langle r, t, q\rangle$ from the view of the adversary with $\mathcal{C}_t$ can be represented as $p(U = u|\langle r, t, q\rangle, \mathcal{C}_t)$. In the following, we give one method the adversary can adopt to calculate the distribution. Through the Bayesian rule we have equation:

$$\begin{aligned} p(U = u|\langle r, t, q\rangle, \mathcal{C}_t) &= \frac{p(\langle r, t, q\rangle|u, \mathcal{C}_t)}{p(\langle r, t, q\rangle, \mathcal{C}_t)} \\ &= \frac{p(\langle r, t, q\rangle|u, \mathcal{C}_t) \cdot p(u|\mathcal{C}_t) \cdot p(\mathcal{C}_t)}{\sum_{u'} p(\langle r, t, q\rangle|u', \mathcal{C}_t) \cdot p(u'|\mathcal{C}_t) \cdot p(\mathcal{C}_t)}. \end{aligned}$$

In the above equation, there are three new distributions. The distribution $p(\mathcal{C}_t)$ measures the probability of the adversary having learned the collection of contextual information $\mathcal{C}_t$. It is difficult to evaluate its value. Whereas, since it appears in both the numerator and the denominator, we can eliminate it from the formula. The distribution $p(u|\mathcal{C}_t)$ represents the probability for user u to issue a request at time t based on the contextual information $\mathcal{C}_t$. As we have no information about the distribution, it can be assumed as uniform according to the principle of maximum entropy [25, 26], which leads to $p(u|\mathcal{C}_t) = p(u'|\mathcal{C}_t)$ $(\forall u' \in \mathcal{U})$. Thus, the target posterior distribution can be further simplified as:

$$p(U = u|\langle r, t, q\rangle, \mathcal{C}_t) = \frac{p(\langle r, t, q\rangle|u, \mathcal{C}_t)}{\sum_{u' \in \mathcal{U}} p(\langle r, t, q\rangle|u', \mathcal{C}_t)}. \tag{1}$$

The probability $p(\langle r, t, q\rangle|u, \mathcal{C}_t)$ indicates the likelihood that 'if user u generates a request at time t then the request will be generalised as $\langle r, t, q\rangle$'. This is actually a joint probability of the following two probabilities. The first is the probability that user u issues the request $\langle u, whereis(u, t), t, q\rangle$ when he sends a request at t. It can also be formulated as the probability that u chooses query q at time t to consult the LBS provider. We call this probability the a priori *probability* of user u. The second probability is the likelihood that the area generalisation algorithm outputs a region r for $whereis(u, t)$. We use $p_u(q|\mathcal{C}_t)$ and $p(f(\langle u, whereis(u, t), t, q\rangle) = \langle r, t, q\rangle)$

Table 1 Notations

$\mathcal{U}$	Set of users
$\mathcal{T}$	Set of time instances
$\mathcal{L}$	Set of locations
$\mathcal{R}$	Set of possible generalised regions
$q \in \mathcal{Q}$	A query supported by the LBS
$\langle u, \ell, t, q\rangle$	A request issued by u at position ℓ at time t
$\langle r, t, q\rangle$	A generalised request
$whereis(u, t)$	Position of user u at time t
$f(\langle u, \ell, t, q\rangle)$	An algorithm computing generalised queries
dis_t	Spatial distribution of users in $\mathcal{U}$ at time t
$u\ell(r, t)$	Set of users located in region r at time t
$req(\langle u, \ell, t, q\rangle)$	User u's privacy requirement on $\langle u, \ell, t, q\rangle$
$query(\langle r, t, q\rangle)$	The query of $\langle r, t, q\rangle$

to represent these two probabilities, respectively. Based on the above discussion, formally we have

$$p(\langle r, t, q\rangle | u, \mathcal{C}_t) = p_u(q|\mathcal{C}_t) \cdot p(f(\langle u, whereis(u, t), t, q\rangle) = \langle r, t, q\rangle). \tag{2}$$

We assume that the generalisation algorithms mentioned in this paper are *deterministic*. In other words, there is always a unique generalised request corresponding to each LBS request, which leads to $p(f(\langle u, whereis(u, t), t, q\rangle) = \langle r, t, q\rangle)$ being either 1 or 0. Furthermore, given an LBS request and a generalised request, the value of this probability is available to the adversary as generalisation algorithms are public. Therefore, the key of query privacy analysis is to calculate $p_u(q|\mathcal{C}_t)$ for any query $q \in \mathcal{Q}$.

The calculation of $p_u(q|\mathcal{C}_t)$ depends on $\mathcal{C}_t$, i.e., the available contextual information. In the following discussion, we give the instantiations of our framework when two different types of contextual information are added into the adversary's knowledge, i.e., user profiles (see Section 4) and query dependency (see Section 5). In this way, we not only show that our framework can handle the contextual information that has been studied (i.e., user profiles), but also demonstrate that it is generic to cope with new context (i.e., query dependency). The important notations are summarised in Table 1.

3.4 Classifying contextual information

From the above discussion, we can see that the adversary can learn new knowledge along with time and we should explicitly distinguish some contextual information at different time. For instance, the contextual information about users' spatial distributions (i.e., $\mathcal{D}_t$) records the sequence of the snapshots of mobile users' locations up to time t and this knowledge keeps growing with time. However, we also notice that certain contextual information remains stable over time such as user mobility patterns and user profiles.

According to this observation, we classify contextual information into two classes—*static* and *dynamic*. Formally they can be defined as follows:

Definition 1 (Static & dynamic context) Let $\varphi_t \in \mathcal{C}_t$ be the value of a type of contextual information at time t. We say that the contextual information is static

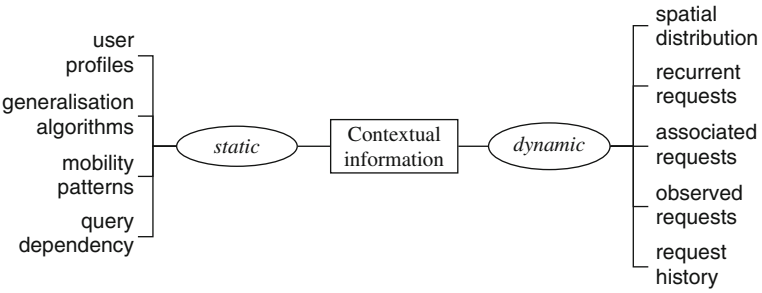


Fig. 2 A classification of contextual information

if and only if for any two time points t and t' in $\mathcal{T}$, $\varphi_t = \varphi_{t'}$. Otherwise, the contextual information is dynamic.

Note that in practice the above definition can be relaxed. When a type of contextual information keeps stable for a sufficiently long period, we can also consider it as static. For instance, a user profile can be interpreted as static even though the user may change his job as switching jobs is not frequent.

In Fig. 2, we classify the contextual information mentioned in this paper. To attack query privacy, the adversary usually combines different contextual information. For instance, when associated requests are explored [4, 10, 13], request generalisation algorithms and users' real-time spatial distribution are also part of the adversary's knowledge.

4 Privacy analysis based on user profiles

In this section, we demonstrate the implementation of our framework when user profiles are explored by the adversary. Although user profiles and their impact on query privacy have been discussed by Shin et al. [42], they do not describe precisely how to exploit user profiles and quantify the amount of benefits gained by the adversary. On the contrary, with our framework we can formally define the attack and use a posterior probability distribution to describe the adversary's knowledge about the issuer.

As discussed in Section 3, given a generalised request $\langle r, t, q\rangle$ the key of query privacy analysis is to compute users' a priori probabilities, e.g., $p_u(q|\mathcal{C}_t)$. Before presenting the calculation, we start with formulating the adversary' knowledge. User profiles are associated with a set of attributes, e.g., contact attributes (zip codes, addresses), descriptive attributes (age, nationalities, jobs) and preference attributes (hobbies, moving patterns) [42]. The values of these attributes can be categorical (e.g., nationality) or numerical (e.g., salary, age). Let $\langle a_1 : \mathcal{A}_1, \ldots, a_m : \mathcal{A}_m\rangle$ be the list of the attributes where a_i is the name of the attribute and $\mathcal{A}_i$ is its domain. The profile of user u can be represented as a tuple of values each of which corresponds to an attribute. Let $\Phi_u = \langle \alpha_1, \ldots, \alpha_m\rangle \in \mathcal{A}_1 \times \ldots \times \mathcal{A}_m$ be the tuple where α_i is the value

of a_i and denoted by $\Phi_u^{a_i}$. Thus the contextual information learnt by the adversary at time t can be represented as the following:

$$\mathcal{C}_t = \{\mathcal{D}_t,\ f, \{\Phi_u | u \in \mathcal{U}\}\}.$$

Our main idea to calculate $p_u(q|\mathcal{C}_t)$ is to compute the relevance of user u's profile to each query and compare the relevance to q with those to other queries. Given an attribute a_i, we can discretise its domain $\mathcal{A}_i$ into intervals if it is numerical or divide the domain into sub-categories if it is discrete. For instance, the domain of attribute address can be categorised in terms of districts while the numerical values of salary can be discretised into three intervals, such as '$\leq$1000', '1000–5000' and '$\geq$5000'. Note that the intervals are mutually exclusive and their union is equal to the original domain.

With the list of the intervals, we can transform the value of an attribute into a vector of binary values based on which interval or category it belongs to. Suppose $\mathcal{A}_i$ is divided into the list of intervals $(\mathcal{A}_i^1, \ldots, \mathcal{A}_i^k)$ where for any $1 \leq x, y \leq k$, $\mathcal{A}_i^x \cap \mathcal{A}_i^y$ and $\cup_{1 \leq j \leq k} \mathcal{A}_i^j = \mathcal{A}_i$. Let $\mathbf{\Phi}_u^{a_i}$ be the vector of $\Phi_u^{a_i}$ and $[\mathbf{\Phi}_u^{a_i}]_j$ be the jth value. Thus, we have

$$[\mathbf{\Phi}_u^{a_i}]_j = \begin{cases} 1 & \text{if } \Phi_u^{a_i} \in \mathcal{A}_i^j; \\ 0 & \text{if } \Phi_u^{a_i} \notin \mathcal{A}_i^j. \end{cases}$$

If a user has a salary of 3000 euros, then $\mathbf{\Phi}_u^{\text{salary}} = [0\ 1\ 0]$.

Each query $q \in \mathcal{Q}$ has a set of related attributes that determines whether it is likely for a user to issue the query q. Furthermore, for a given related attribute, its value decides the amount of likelihood. For instance, for the query asking for expensive hotels, the associated attributes should include salary, jobs and age while gender is irrelevant. Among them, a salary is much more relevant than age and moreover, a salary of more than 5000 euros is much more important than one of less than 1000 euros. Therefore, we introduce a relevance vector for each attribute to express the relation between attributes' values and queries. Let $W_q^{a_i} = [w_1 \ldots w_n]$ be the relevance vector of query q of attribute a_i. For any $u \in \mathcal{U}$ and $q \in \mathcal{Q}$, the relevance value of user u's profile to query q can be calculated as follows:

$$v_u(q) = \sum_{i \leq m} \mathbf{\Phi}_u^{a_i} \cdot [W_q^{a_i}]^T$$

where $[W_q^{a_i}]^T$ is the transpose of $W_q^{a_i}$. Suppose the relevance vector of attribute salary to a five-star hotel is [0 0.2 0.6] then $v_u(q) = [0\ 1\ 0] \cdot [0\ 0.2\ 0.6]^T = 0.2$. Finally, we can calculated u's a priori probability $p_u(q|\mathcal{C}_t)$ as follows:

$$p_u(q|\mathcal{C}_t) = \frac{v_u(q)}{\sum_{q' \in \mathcal{Q}} v_u(q')}.$$

As users are independent from each other to decide next queries to issue and user profiles are the only additional information in $\mathcal{C}_t$ to the inherent contexts, for the sake of simplicity we use $p_u(q|\mathcal{P}_u)$ to replace $p_u(q|\mathcal{C}_t)$ when there is no confusion from the context.

5 Privacy analysis based on query dependency

In this section, we identify a new type of contextual information—*query dependency* and present how to incorporate it into our framework.

Since the first commercial LBSs launched in 1996, LBSs have evolved from simple single-target finder to diverse, proactive and multi-target services [2]. However, due to the lack of user privacy protection, especially at the beginning, LBS providers accumulate a large amount of users' historical requests. What makes the situation worse is the shift of LBS providers from telecom operators (who were viewed as trusted entities) to open businesses such as Google Latitude, Foursquare, and MyTracks. This increases the risk of potential misuse of the accumulated records due to the new sensitive information derived from them.

The dependency between queries is one type of such sensitive but personal information. It is contained in users' requests because of users' preference in arranging their daily activities [21]. This preference leads to a repetitive pattern in their requests. For instance, a user often posts a check-in of a coffee house after lunch. The fact that users' frequent queries are usually restricted to a small set makes the extraction of query dependency more precise.

Users' query dependency can be abused and becomes a potential risk to users' query privacy. As far as we know, we are the first to explore *query dependency* for query privacy protection. We illustrate this by a simple example.

Example 1 Bob issues a request about the nearest night clubs in a 2-anonymous region with Alice being the other user. Suppose the adversary has also learnt that Alice just issued a query about the nearest clinics and Bob queried about bars. As it is not common to ask clubs after clinics compared to bars, the adversary can infer that Bob is more probable to issue the request about night clubs. In this example, even if Alice and Bob share a similar profile, the dependency between queries obviously breaks 2-anonymity for all users in the region who are supposed to be equally likely to issue the request.

In the rest of this section, we start with a formal definition of the adversary's knowledge and then give an approach to derive dependency between queries for a user from his request history. Then we propose a method for the adversary to breach users' query privacy by exploring query dependency.

5.1 Updating the adversary's knowledge

Besides the contextual information considered in Section 4, there are two new types of contextual information added—*request history* and *observed request traces*.

As we have mentioned before, LBS providers have collected users' request history. For each user u, we assume that the adversary has a user u's request history for a sufficiently long period. We use a sequence to denote the requests of user u collected by the adversary, i.e., $\mathcal{H}_u = (\langle u, \ell_1, t_1, q_1\rangle, \ldots, \langle u, \ell_n, t_n, q_n\rangle)$ $(\forall_{1\le i\le n-1} t_i < t_{i+1})$. The ith request in $\mathcal{H}_u$ is represented by $\mathcal{H}_u(i)$. We call this sequence *user request history*, whose length is denoted as $len(\mathcal{H}_u)$. For the sake of simplicity, we assume that $\mathcal{H}_u$

is complete, namely there do not exist any requests that are issued by u during the period but are not included in $\mathcal{H}_u$.

Another assumption is that the adversary has access to the public channel which transmits generalised requests. This means that the adversary can obtain any generalised requests from users. We denote this contextual information by a sequence of generalised requests in the chronologically ascending order. Up to time t, the sequence of observed requests is $\mathcal{O}_t = (\langle r_1, t_1, q_1\rangle, \ldots, \langle r_n, t_n, q_n\rangle)$ ($\forall_{1\leq i\leq n} t_i < t_{i+1}$ and $t_n < t$)). For the sake of simplicity, we do not consider recurrent queries, i.e., those elements in $\mathcal{O}_t$ with the same time-stamps. Furthermore, for each request in $\mathcal{O}_t$, the adversary calculates its anonymity set, i.e., the users located in the generalised region. Thus, for each user, the adversary can maintain a sequence of generalised requests, whose anonymity sets contain this user. We call this sequence an *observed request trace* and denote the one of user u up to t as $\mathcal{O}_{u,t}$ whose length is $len(\mathcal{O}_{u,t})$. Obviously with time passing, a user's observed request trace keeps growing. The difference between $\mathcal{H}_u$ and $\mathcal{O}_{u,t}$ is that the adversary is certain about the issuer of each request in $\mathcal{H}_u$ but uncertain about the issuers of the requests in $\mathcal{O}_{u,t}$.

To summarise, the knowledge of the adversary can be formulated as the following:

$$\mathcal{C}_t = \{\mathcal{D}_t,\ f, \{\Phi_u | u \in \mathcal{U}\}, \mathcal{O}_t, \{\mathcal{H}_u | u \in \mathcal{U}\}\}.$$

5.2 Deriving query dependency

Query dependency can be used to predict a user's next query based on his past queries. However, when a user has no past queries or the past queries have little impact on his future queries, we need to consider users' a priori *preference* on queries.

Query dependency We model query dependency with the assumption that the query that a user will issue next can only be affected by the last query that the user has issued (i.e., the Markov property). For a pair of queries q_i and q_j, the dependency of query q_j on q_i can thus be expressed by the conditional probability $p_u(q_j|q_i)$.

To find dependent queries, we need to identify the successive requests. Intuitively, two requests are successive if there are no other requests between them in the request history. This simply means that $\mathcal{H}_u(i+1)$ is the successive request of $\mathcal{H}_u(i)$ for $i < len(\mathcal{H}_u)$. All the occurrence of query q_j depending on q_i can be captured by the set of successive request pairs $\mathcal{S}_{i,j} = \{(\mathcal{H}_u(k), \mathcal{H}_u(k+1)) | req(\mathcal{H}_u(k)) = q_i \wedge req(\mathcal{H}_u(k+1)) = q_j, 0 < k < len(\mathcal{H}_u)\}$. Given a request history $\mathcal{H}_u$, the adversary can derive for the user u his dependency between any pair of queries based on the sets $\mathcal{S}_{i,j}$. In this paper we make use of Lidstone's or additive smoothing [32] to ensure that there is no dependency of degree zero for q_j on q_i due to no occurrence of the pair (q_i, q_j) in the request history. Formally, let λ be the smoothing parameter which is usually set to 1. The dependency $p_u(q_j|q_i)$ is calculated as follows:

$$p_u(q_j|q_i) = \frac{|\mathcal{S}_{i,j}| + \lambda}{\sum_{q_k \in \mathcal{Q}} |\mathcal{S}_{i,k}| + |\mathcal{Q}| \cdot \lambda}.$$

A priori preference There are many cases that a query does not depend on its past queries. For example, users may issue an LBS query for the first time or accidentally for an emergent need. In such cases, the best the adversary can do is to apply users' a priori preference to find the possible issuer.

We model the a priori preference of a user u as a distribution over the set of queries indicating the probability of the user to issue a query. For query $q_i \in \mathcal{Q}$, we denote by $p_u(q_i)$ the probability that user u issues the query q_i when there is no dependence on any previous queries. It is obvious that $\sum_{q_i \in \mathcal{Q}} p_u(q_i) = 1$.

There are many sources of information reflecting users' a priori preference. Users' personal information, i.e., user profiles, have been discussed in Section 4 and shown effective in assessing users' preference on queries [42, 43]. Moreover, a user's request history also reflects his preference. Thus, we estimate a user's a priori preference by combining his request history ($\mathcal{H}_u$) and his user profile. Recall that we calculate a distribution for each user over the set of queries indicating the probability that the user issues a query based on his profile, i.e., $p_u(q_i|\mathcal{P}_u)$. Moreover, let $p_u(q_i|\mathcal{H}_u)$ be the likelihood for user u to issue q_i based on his request history. We can use the frequency of the occurrence of the query in the request history to estimate $p_u(q_i|\mathcal{H}_u)$:

$$p_u(q_i|\mathcal{H}_u) = \frac{|\{\mathcal{H}_u(k)|query(\mathcal{H}_u(k)) = q_i\}|}{len(\mathcal{H}_u)}.$$

The two distributions evaluate a user's a priori preference on next queries from two different perspectives. An agreement between them is needed. This is equivalent to aggregate expert probability judgements [5]. We use *linear opinion pool aggregation* which is empirically effective and has been widely applied in practice [1]. By assigning a weight to each distribution, i.e., $w_{\mathcal{P}}$ and $w_{\mathcal{H}}$ with $w_{\mathcal{P}} + w_{\mathcal{H}} = 1$, we can calculate $p_u(q_i)$ as follows:

$$p_u(q_i) = w_{\mathcal{P}} \cdot p_u(q_i|\mathcal{P}_u) + w_{\mathcal{H}} \cdot p_u(q_i|\mathcal{H}_u).$$

Remark 1 The way we model users' query dependency and a priori preference has some restrictions. For instance, we do not consider the influence of factors such as the time when LBS requests are issued—usually a user's behaviours on weekdays are different from weekends. By distinguishing the request history at different time periods, the impact of time can be taken into account. We have also assumed that a query is only dependent on its immediate previous query. This restriction can be lifted by considering, e.g., the last k historical queries. However, such dependency might not be as efficient and accurate as the probabilities of the form of $p_u(q_i|q_j)$. An interesting factor is the time intervals between successive requests, which may present certain patterns as well. For instance, a user may prefer to issue a request within a specific amount of time after the previous one. This leads to various probabilities for a user to issue a query when he chooses different issuing time. In Section 5.4, we take time intervals between requests as an example to illustrate how to extend our model of query dependency to capture more influencing factors.

5.3 Query privacy analysis

Recall that the purpose of the adversary is to calculate the distribution $p(U = u|\langle r, t, q\rangle, \mathcal{C}_t)$ given a generalised request $\langle r, t, q\rangle$. In the adversary's knowledge, the

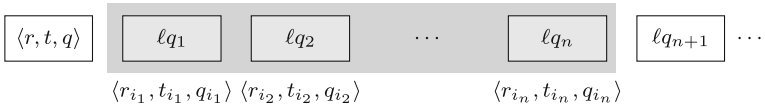


Fig. 3 A history window of n observed requests

observed request list ($\mathcal{O}_t$) is the only dynamic context besides the spatial distribution (i.e., $\mathcal{D}_t$) which is inherently contained. For the sake of simplicity, we use $p(u|\langle r, t, q\rangle, \mathcal{O}_t)$ for short to represent $p(U = u|\langle r, t, q\rangle, \mathcal{C}_t)$ whenever no confusion exists.

The key of query privacy analysis is still to calculate $p_u(q|\mathcal{C}_t)$ (i.e., $p_u(q|\mathcal{O}_t)$ for short). Due to the independence between users with respect to issuing requests, a user's requests have no influence on the next query of any other user. Thus, $p_u(q|\mathcal{O}_t) = p_u(q|\mathcal{O}_{u,t})$.

The size of $\mathcal{O}_{u,t}$ is an important factor determining the accuracy and the complexity of the calculation of $p_u(q|\mathcal{O}_{u,t})$. Recall that $\mathcal{O}_{u,t}$ consists of all the observed requests that may be issued by user u up to time t. Intuitively, the longer $\mathcal{O}_{u,t}$ is, the more computational overhead is required to obtain $p(u|\langle r, t, q\rangle, \mathcal{O}_t)$. Therefore, it is not practical to consider the complete $\mathcal{O}_{u,t}$. Instead, we fix a *history window* which consists of the latest n observed requests of user u ($n \leq len(\mathcal{O}_{u,t})$). Our problem can thus be reformulated as to compute $p^n(u|\langle r, t, q\rangle, \mathcal{O}_t)$, indicating that the distribution is based on the last n observed requests.

In Fig. 3, we show an example of a history window which contains n observed requests, $\langle r_{i_1}, t_{i_1}, q_{i_1}\rangle, \ldots, \langle r_{i_n}, t_{i_n}, q_{i_n}\rangle$ with $t_{i_j} > t_{i_{j-1}}$ ($j > 1$). Let $\ell q_j(\mathcal{O}_{u,t})$ be the jth latest observed request in $\mathcal{O}_{u,t}$, whose query is $query(\ell q_j(\mathcal{O}_{u,t})) = q_{i_j}$. In the following discussion, we simply write ℓq_j if $\mathcal{O}_{u,t}$ is clear from the context. It is obvious that ℓq_1 is the latest observed request of user u.

Once $p^n(u|\langle r, t, q\rangle, \mathcal{O}_t)$ is calculated, it is added into the adversary's knowledge. Therefore, for a past request $\langle r', t', q'\rangle$ in $\mathcal{O}_{u,t}$ ($t' < t$), the adversary has $p(u|\langle r', t', q'\rangle, \mathcal{O}'_t)$. In the sequel, we simply denote it as $p(u|\langle r', t', q'\rangle)$ in cases without confusion.

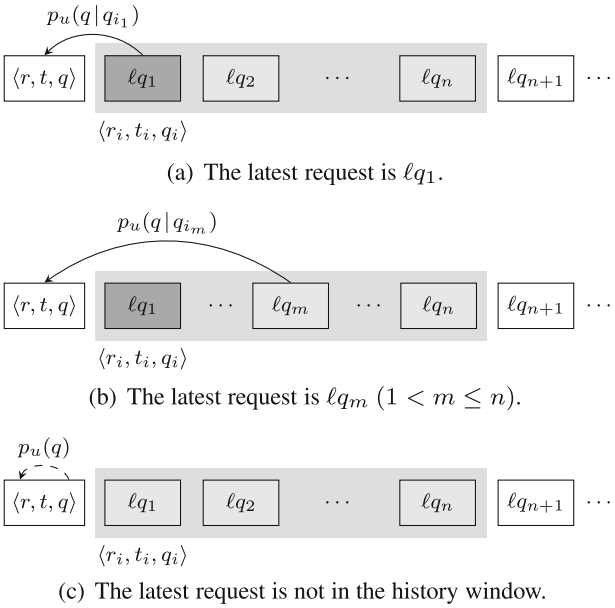
A user's latest request determines the probability distribution of his next query. Whereas, it is uncertain which is the latest in the history window. To handle this uncertainty, we distinguish three cases which are depicted in Fig. 4.

1. User u has issued both the last request (i.e., ℓq_1, see Fig. 4a) and the current request (i.e., $\langle r, t, q\rangle$). Considering query dependence, the probability of this case is

$$p_u(u|\ell q_1) \cdot p_u(q|q_{i_1}).$$

2. User u has issued the current request $\langle r, t, q\rangle$ and his latest request is ℓq_m ($1 < m \leq n$) (see Fig. 4b). The probability of ℓq_m being the latest request is the production of the probability that the last $m - 1$ requests are *not* issued by u and the probability that u has issued ℓq_m, i.e., $p(u|\ell q_m) \cdot \prod_{j=1}^{m-1}(1 - p(u|\ell q_j))$.

Fig. 4 The three cases



Considering query dependence, the probability of this case is

$$p_u(q|q_{i_m}) \cdot p(u|\ell q_m) \cdot \prod_{j=1}^{m-1}(1 - p(u|\ell q_j)).$$

3. User u did not issue any request in the history window (see Fig. 4c). In this case, we suppose that the user issued the current request according to his a priori preference, i.e., $p_u(q)$. Based on the probability that the user's latest request is outside of the history window as $\prod_{j=1}^{n}(1 - p(u|\ell q_j))$, the probability of this case is

$$p_u(q) \cdot \prod_{j=1}^{n}(1 - p(u|\ell q_j)).$$

We sum up the above three probabilities to compute the probability for user u in region r at time t to issue q when a history window of size n is considered:

$$\begin{aligned} p_u^n(q|\mathcal{O}_{u,t}) = {} & p(u|\ell q_1) \cdot p_u(q|query(\ell q_1)) \\ & + \sum_{m=2}^{n} p(u|\ell q_m) \cdot p_u(q|query(\ell q_m)) \cdot \prod_{j=1}^{m-1}(1 - p(u|\ell q_j)) \\ & + p_u(q) \cdot \prod_{j=1}^{n}(1 - p(u|\ell q_j)). \end{aligned} \tag{3}$$

We use the following example with $n = 2$ to show the calculation.

Example 2 Suppose the last two requests are $\langle r'', t'', q'' \rangle$ and $\langle r', t', q' \rangle$ with $t'' < t' < t$ in $\mathcal{O}_{u,t}$. Let $\langle r, t, q \rangle$ be an observed request. Then for user u, the probability that he issues the request is computed as follows:

$$\begin{aligned} p_u^2(q|\mathcal{O}_{u,t}) = & \; p_u(q|q') \cdot p(u|\langle r', t', q' \rangle) \\ & + \big(1 - p(u|\langle r', t', q' \rangle)\big) \cdot p(u|\langle r'', t'', q'' \rangle) \cdot p_u(q|q'') \\ & + \big(1 - p(u|\langle r', t', q' \rangle)\big) \cdot \big(1 - p(u|\langle r'', t'', q'' \rangle)\big) \cdot p_u(q). \end{aligned}$$

5.4 Handling the time intervals between requests

In this section, we study a factor that has impact on query dependency—time intervals between two successive requests.

It has been noted that not only the behaviours of a user follow certain patterns but also the amount of time between behaviours. For instance, Giannotti et al. [19, 20] study and extract the pattern of the time intervals between events in sequential pattern mining. The idea is also adopted by Chen et al. [9] for constructing and comparing user mobility profiles. Similarly, with respect to LBS requests, users can also have their preferences on the time intervals between two successive requests.

Example 3 Consider a user who is in a city centre and wants to meet his friends at a bar. He first sends an LBS request asking for nearby friends who can potentially meet together. Then the user contacts those friends and discuss with them about their availability, which takes about half an hour. Afterwards, he issues another request for nearby bars.

In the above example, the time interval between the two requests should usually be around 30 min. Suppose that the user sent another query 2 min after the first query about nearby friends. Then this query is less likely to be a query on nearby bars, compared to the situation when a query is issued about 30 min later. Therefore, query dependency should vary according to when the next query is issued.

To capture the influence of query issuing time, given two queries q_i and q_j, instead of $p_u(q_j|q_i)$ we calculate the distribution $p_u(q_j|q_i, \tau)$, where τ is the amount of time after user u issued the last request with query q_i. This distribution can be calculated based on other distributions deduced from the following equation:

$$\begin{aligned} p_u(q_j|q_i, \tau) &= \frac{p_u(\tau|q_j, q_i) \cdot p_u(q_j, q_i)}{p_u(\tau|q_i) \cdot p_u(q_i)} \\ &= \frac{p_u(\tau|q_j, q_i)}{p_u(\tau|q_i)} \cdot p_u(q_j|q_i). \end{aligned}$$

There are two new distributions in the above equation. The first one is $p_u(\tau|q)$ indicating the probability that a user issues a successive query with time interval τ after issuing query $q \in \mathcal{Q}$. The other distribution is $p_u(\tau|q_j, q_i)$ meaning the likelihood that if user u issues query q_j after q_i, then time interval between them is τ.

The time interval between requests can be considered as a random variable T. The above two distributions can thus be calculated based on the probability density

functions of T in different cases, i.e., $\hat{f}(T|q)$ and $\hat{f}(T|q_j, q_i)$. Let ϵ be the granularity of time, e.g., a second or a minute. Then given a time interval τ, we have

$$p_u(\tau|q_i) = \int_{\tau}^{\tau+\epsilon} \hat{f}(T|q_i)\mathrm{d}T; \quad p_u(\tau|q_j, q_i) = \int_{\tau}^{\tau+\epsilon} \hat{f}(T|q_j, q_i)\mathrm{d}T.$$

The problem of density estimation based on observed data has been extensively studied and some classic methods have been developed in practice, e.g., the kernel estimator [12]. In our case, the key to estimate the density function of T is to extract the corresponding set of observed samples of time intervals. Take $\hat{f}(T|q_j, q_i)$ as an example. The samples of time intervals form a multi-set which can be obtained from users' request history, e.g., $\mathcal{H}_u$. Recall that $\mathcal{S}_{i,j}$ is the set of pairs of successive requests whose queries are q_i and q_j, respectively. Then the observed set of time intervals is

$$\{t' - t|(\langle r, t, q_i\rangle, \langle r', t', q_j\rangle) \in \mathcal{S}_{i,j}\}.$$

The calculation of user u's a priori probability at time t to issue query q (i.e., Eq. 3) can thus be extended to handle the query dependency with respect to time intervals. The calculation is shown in Eq. 4:

$$\begin{aligned} p_u^n(q|\mathcal{O}_{u,t}) = {} & p(u|\ell q_1) \cdot p_u(q|query(\ell q_1), t - time(\ell q_1)) \\ & + \sum_{m=2}^{n} p(u|\ell q_m) \cdot p_u(q|query(\ell q_m), t - time(\ell q_m)) \cdot \prod_{j=1}^{m-1}(1 - p(u|\ell q_j)) \\ & + p_u(q) \cdot \prod_{j=1}^{n}(1 - p(u|\ell q_j)). \end{aligned} \tag{4}$$

6 Measuring query privacy

LBS requests are generalised to protect the issuers' query privacy. The level of query privacy offered by the generalisation algorithms should be quantified precisely. This is due to (i) the generalisation algorithm requires the evaluation so as to improve their performance; (ii) LBS users need the quantification to express their privacy requirements for their requests.

Besides k-anonymity, many privacy metrics have been proposed in the literature, such as correctness-based [44], estimation error-based [35] and feeling-based [47]. These metrics quantify query privacy from different perspectives. For instance, the feeling-based metric makes use of entropy to evaluate the average uncertainty of the adversary to guess the issuer in a given scenario (e.g., shopping mall) which is subsequently used as the privacy requirement of users. Correctness-based metrics quantify privacy as the probability of the adversary choosing the right issuer when he makes a single guess. Using our framework, we can adopt the ideas of these metrics, which leads to a diverse and comprehensive series of measurements for query privacy. In this section, we present three new metrics on query privacy and formally define them using our framework.

Inspired by anonymity degrees defined by Reiter and Rubin [36], we come up with the following two new privacy metrics—*k-approximate beyond suspicion* and *user specified innocence*. Note that user specified innocence coincides with the idea

of correctness-based metrics. Furthermore, we propose a third metric by exploring entropy.

k-approximate beyond suspicion *Beyond suspicion* means from the attacker's viewpoint the issuer cannot be more likely than other potential users in the anonymity set to issue the query. In the context of LBSs, we need to find a set of users in which users are the same likely to send a given query. This set is taken as the anonymity set whose size determines the degree of users' privacy as in k-anonymity. Let $AS : Q' \rightarrow 2^{\mathcal{U}}$ denote the anonymity set of a generalised request. The issuer of query $\langle u, whereis(u, t), t, q\rangle$ is beyond suspicious with respect to the corresponding generalised request $\langle r, t, q\rangle$ if and only if $\forall u' \in AS(\langle r, t, q\rangle)$,

$$p(u|\langle r, t, q\rangle, \mathcal{C}_t) = p(u'|\langle r, t, q\rangle, \mathcal{C}_t).$$

In practice, the number of users with the same probability to send a query is usually small, which leads to a large generalised area with a fixed k. So we relax the requirement to compute an anonymity set consisting of users with *similar probabilities* instead of the exact same probability. Let $\|p_1, p_2\|$ denote the difference between two probabilities and ϵ be the pre-defined parameter describing the largest difference allowed between similar probabilities.

Definition 2 (k-approximate beyond suspicion) Let $\langle u, whereis(u, t), t, q\rangle \in Q$ be a query and $\langle r, t, q\rangle \in Q'$ the corresponding generalised request. The issuer u is *k-approximate beyond suspicious* if

$$|\{u' \in AS(\langle r, t, q\rangle) \mid \|p(u|\langle r, t, q\rangle, \mathcal{C}_t), p(u'|\langle r, t, q\rangle, \mathcal{C}_t)\| < \epsilon\}| \geq k.$$

Different from k-anonymity, the set of users that are k-approximate beyond suspicious is computed based on the spatial distribution of users with similar probabilities rather than the original distribution involving all users. The users in an anonymity set have similar probabilities and the size of the anonymity set is larger than k. Therefore, k-approximate beyond suspicion can be seen as a generalised version of k-anonymity. If for a specific query $q \in \mathcal{Q}$, any two users have the same probability to issue it, then k-approximate beyond suspicion is equivalent to k-anonymity. For short, we use k-ABS to denote k-approximate beyond suspicion in the following discussion.

User specified innocence *Probable innocence* and *possible innocence* are proposed by Reiter and Rubin [36]. An issuer is probably innocent if from the attacker's view the issuer appears no more likely to be the originator of the query. In other words, the probability of each user in the anonymity set to be issuer should be less than 50 %. Meantime, possible innocence requires the attacker not be able to identify the issuer with a non-trivial probability. We extend these two notions into a metric with user-specified probabilities (instead of restricting to 50 % or non-trivial probability which is not clearly defined). We call the new anonymity metric *user specified innocence* where $\alpha \in [0, 1]$ is the specified probability given by the issuer. Intuitively, for a query, an issuer is α-user specified innocent, if the anonymiser generates the same region for any user in the region with the same specified value α. In other words, in the generalised region, the most probable user has a probability smaller than α.

Recall that $u\ell(r,t)$ denotes the set of users in region r at time t. It is clear that the anonymity set consists of all users in the generalised area.

Definition 3 (User specified innocence) Let $\alpha \in [0,1]$, $\langle u, whereis(u,t), t, q\rangle \in Q$ be a query and $\langle r,t,q\rangle \in Q'$ the corresponding generalised request. The issuer u is *α-user specified innocent* if for all $u' \in u\ell(r,t)$,

$$p(u'|\langle r,t,q\rangle, \mathcal{C}_t) \leq \alpha.$$

We abbreviate α-user specified innocence as α-USI.

An entropy-based metric Serjantov and Danezis [41] define an anonymity metric based on entropy and Díaz et al. [15] provide a similar metric that is normalised by the number of users in the anonymity set. The concept *entropy* of a random variable X is defined as $H(X) = -\sum_{x\in\mathcal{X}} p(x)\cdot\log p(x)$ where $\mathcal{X}$ is the domain (all possible values) of X. In our context, entropy can also be used to describe the attacker's uncertainty to identify the issuer of a generalised request. Let variable U denote the issuer of query $\langle r,t,q\rangle$. Then the uncertainty of the attacker can be expressed as

$$H(U|\langle r,t,q\rangle, \mathcal{C}_t) = -\sum_{u'\in u\ell(r,t)} p(u'|\langle r,t,q\rangle, \mathcal{C}_t)\cdot\log p(u'|\langle r,t,q\rangle, \mathcal{C}_t).$$

For a given generalised request $\langle r,t,q\rangle$ and a given value β, we say that the issuer is entropy-based anonymous with respect to the value β if all users in region r can have r as the generalised region when issuing the same query and the entropy $H(U|\langle r,t,q\rangle, \mathcal{C}_t)$ is not smaller than β.

Definition 4 (Entropy-based anonymity) Let $\beta > 0$, $\langle u, whereis(u,t), t, q\rangle \in Q$ be a query and $\langle r,t,q\rangle \in Q'$ the corresponding generalised request. The issuer u is *β-entropy based anonymous* if

$$H(U|\langle r,t,q\rangle, \mathcal{C}_t) \geq \beta.$$

For short, we call β-entropy based anonymity β-EBA.

Remark When users use these metrics to express their privacy requirements, at least three elements should be provided—a metric, the values of the parameters required by the chosen metric (e.g., k, α), and the values of the parameters used to calculation posterior probabilities (e.g., the size of history windows).

In practice it is difficult and cumbersome for a user to give exact values to the elements. First, all the metric values in requirements should be determined before requests are generalised (i.e., ex-ante) but they are defined ex-post in nature in the metric. Furthermore, users need to understand the meaning of each parameter and the corresponding implication on privacy protection. To avoid this situation, in this paper we provide a list of privacy levels, e.g., from *low* to *very high*. Each level corresponds to a setting of privacy parameters. For example, when query dependency is considered, a user's privacy requirement can be represented as $\langle$kABS, *high*$\rangle$, which is then transformed into $\langle$kABS, $(10, 0.05), (5)\rangle$. This ensures that whenever a request is successfully generalised, the region contains 10 users with similar posterior probabilities to the issuer's, after taking into account the last 5 observed requests.

Furthermore, the distance between two such users' posterior probabilities is bounded by 0.05. In practice, the transformation can be made automatic and embedded in the request generalisation process. Note that the existing works can also be adapted to determine the values, e.g., the feeling-based privacy metric [47].

7 Generalisation algorithms

In this section, we develop area generalisation algorithms to compute regions satisfying users' privacy requirements expressed in the proposed metrics in Section 6. As to find a region satisfying k-ABS is similar to compute a region satisfying k-anonymity on a given spatial distribution, we design an algorithm for k-ABS by combining the algorithm grid [33] with a clustering algorithm. For the other metrics, we design a uniform algorithm based on dichotomicPoints [33].

7.1 An algorithm for k-ABS

To find an area that satisfies k-ABS is to guarantee that at least k users in the area have similar posterior probabilities. This task can be divided into two main steps. The first is to obtain the spatial distribution of the users who have similar a priori probabilities to the issuer (e.g., $p_u(q|\mathcal{C}_t)$). The second step is to run a k-anonymity generalisation algorithm to find a region with at least k users based on the spatial distribution computed at the first step.

The first step can be transformed to the clustering problem. Given $q \in \mathcal{Q}$, we need to cluster the users in $\mathcal{U}$ such that the users with similar a priori probabilities with respect to issuing q are grouped together.

For the second step, we use algorithm grid by Mascetti et al. [33] as it generates regular regions with smaller area compared to others. A two-dimensional space is partitioned into a grid with $\lfloor \frac{N}{k} \rfloor$ cells each of which contains at least k users, where N denotes the number of users in $\mathcal{U}$. A user's position is represented by two dimensions x and y. The algorithm grid consists of two steps. First, users are ordered based on dimension x, and then on y. The ordered users are divided into $\lfloor \sqrt{\frac{N}{k}} \rfloor$ blocks of consecutive users. The block with the issuer enters the second step. The users in this block are then ordered first based on dimension y and then x. These users are also partitioned into $\lfloor \sqrt{\frac{N}{k}} \rfloor$ blocks. Then the block with the issuer is returned as the anonymity set. Details of the grid algorithm can be found in [33].

Algorithm 1 describes our algorithm for k-ABS. In general, it gives the generalised region as output which satisfies the user requirement k. Function cluster returns the cluster of users with similar probabilities to that of u with respect to query q. Then the function grid outputs a subset of *sim_users* with at least k users who are located in the rectangular region. The generalised region is computed by function region.

Note that the clustering algorithm does not have to run each time when there is a request coming to the anonymiser. As long as the spatial distribution remains static or does not have big changes, for the requests received during this period, the anonymiser just executes the clustering algorithm once and returns the cluster containing the issuer as output of function cluster. The choice of the clustering algorithms has an impact on the performance of the generalisation algorithm.

Algorithm 1 A generalisation algorithm for k-ABS.

FUNCTION: kABS
INPUT: $\langle u, whereis(u,t), t, q\rangle, dis(t), k, \mathcal{M}(q) = \{p_{u'}(q|\mathcal{C}_t)|u' \in \mathcal{U}\}$
OUTPUT: A region r that satisfies k-ABS

sim_users :=cluster$(u, q, \mathcal{M}(q))$;
AS := grid$(sim_users, dis(t), k)$;
r := region(AS)

The complexity of Algorithm 1 is the sum of those of the clustering algorithm implemented and the gird algorithm ($\mathcal{O}(\sqrt{kN}\log\sqrt{kN})$ [33]). The correctness of Algorithm 1 is stated as Theorem 1 and its proof is rather straightforward.

Theorem 1 *For any $\langle u, \ell, t, q\rangle \in Q$, Algorithm 1 computes a generalised region in which the issuer u is k-approximate beyond suspicious.*

7.2 An algorithm for α-USI and β-EBA

For privacy metrics α-USI and β-EBA, we design a uniform algorithm where users can specify which metric to use. Recall that in grid, the number of cells is predetermined by k and the number of users. Thus it is not suitable to perform area generalisation for metrics without a predefined number k. Instead we use algorithm dichotomicPoints.

The execution of dichotomicPoints involves multiple iterations in each of which users are split into two subsets. Similar to grid, positions are represented in two dimensions x and y, and users are also ordered based on their positions. However, different from grid the orders between axes are determined by the shape of intermediate regions rather than fixed beforehand. Specifically, if a region has a longer projection on dimension x, then x is used as the first order to sort the users. Otherwise, y is used as the first order. Users are then ordered based on the values of their positions on the first order axis and then the second order. Subsequently, users are partitioned into two blocks with the same or similar number of users along the first order axis. The block containing the issuer is taken into the next iteration. This process is repeated until any of the two blocks contains less than $2k$ users. This termination criterion is to ensure security against the outlier problem (see Section 2).

However, in our uniform algorithm, instead of checking the number of users, we take the satisfaction of users' privacy requirement as the termination criterion, e.g., if all users in the two blocks have a probability smaller than α.

Given a request, our uniform algorithm executes three main steps to calculate the generalised region. The first step is to update users' a priori probabilities (at time t) based on the latest contextual information $\mathcal{C}_t$. This is done by the procedure updatePriori. This step can be skipped if the evolution of the contextual information does not affect the a priori probabilities, e.g., when only user profiles are contained. In the second step, after determining the first order axis, we call function updateAS to find a smaller anonymity set. It takes a set of users and partitions them into two subsets along the first order axis, both of which should satisfy the issuer's privacy requirement and updateAS returns the one containing the issuer as the updated

Algorithm 2 The uniform generalisation algorithm for α-USI and β-EBA.

```
1: FUNCTION: uniformDP
2: INPUT: qu = ⟨u, ℓ, t, q⟩, req(qu), C_t
3: OUTPUT: Region r that satisfies req(qu)
4:
5: AS := U;
6: updatePriori(AS); \* for each u' ∈ AS, calculate p_u(q|C_t). *\
7: cont := check(AS, req(qu));
8: if cont = false then
9:    return ∅;
10: end if
11: while cont do
12:    min_x := min_{u'∈AS} whereis(u').x;
13:    min_y := min_{u'∈AS} whereis(u').y;
14:    max_x := max_{u'∈AS} whereis(u').x;
15:    max_y := max_{u'∈AS} whereis(u').y;
16:    if (max_x − min_x) ≥ (max_y − min_x) then
17:       first := x;
18:       second := y;
19:    else
20:       first := y;
21:       second := x;
22:    end if
23:    AS' = updateAS(AS, req(qu), first);
24:    if AS' = AS then
25:       AS' = updateAS(AS, req(qu), second);
26:    end if
27:    if AS' ≠ AS then
28:       cont := true;
29:    else
30:       cont := false;
31:    end if
32: end while
33:
34: updateContext(C_t);
35: return region(AS);
```

anonymity set. When it is not possible to partition users along the first order axis, i.e., one of the two blocks generalised by any partition fails the issuer's requirement, the second order axis will be tried. If both tries have failed, updateAS simply returns the original set, which means no possible partition can be made with respect to the privacy requirement. In this situation, the whole algorithm terminates. Otherwise, the new set of users returned by updateAS is taken into the next iteration. Last, if the request can be generalised, then we should update the contextual information to include the generalised request, e.g., the observed request lists (i.e., $\mathcal{O}_t$, $\mathcal{O}_{u,t}$). This is done by calling the function updateContext whose implementation is determined by the exploited contextual information.

Algorithm 2 describes the uniform algorithm in detail. The function check(AS, $req(qu)$) calculates the normalised a priori probability of each user in AS. Then the function takes the resulted normalised probabilities as the users' posterior probabilities and check whether they satisfy the requirement $req(qu)$. The boolean variable *cont* is used to decide whether the algorithm should continue. It is set to false when the set of users in $\mathcal{U}$ does not satisfy the requirement (line 7) or when AS cannot be partitioned furthermore (line 30). The former case means that the requirement $req(qu)$ is set too hight to be satisfied and the algorithm should immediately terminate while the latter case indicates that the generalised region is found. The anonymity set AS is represented as a two-dimensional array. After ordering users in AS, $AS[i]$ consists of all users whose positions have the same value on the first order axis. We use len($order$) to denote the size of AS in the dimension denoted by *order*. For instance, in Fig. 5a, axis x is the first order axis and $AS[3]$ has three users with the same x values. Moreover, len($first$) is 6.

The function updateAS shown in Algorithm 3 is critical for our algorithm uniformDP. It takes as input a set of users and outputs a subset that satisfies the issuer's privacy requirement $req(qu)$. It first orders the users and then divides them into two subsets with the same number of users along the first order axis (indicated by the variable *order*). This operation is implemented by the function mid(AS, *order*) which returns the middle user's index in the first dimension of AS. If

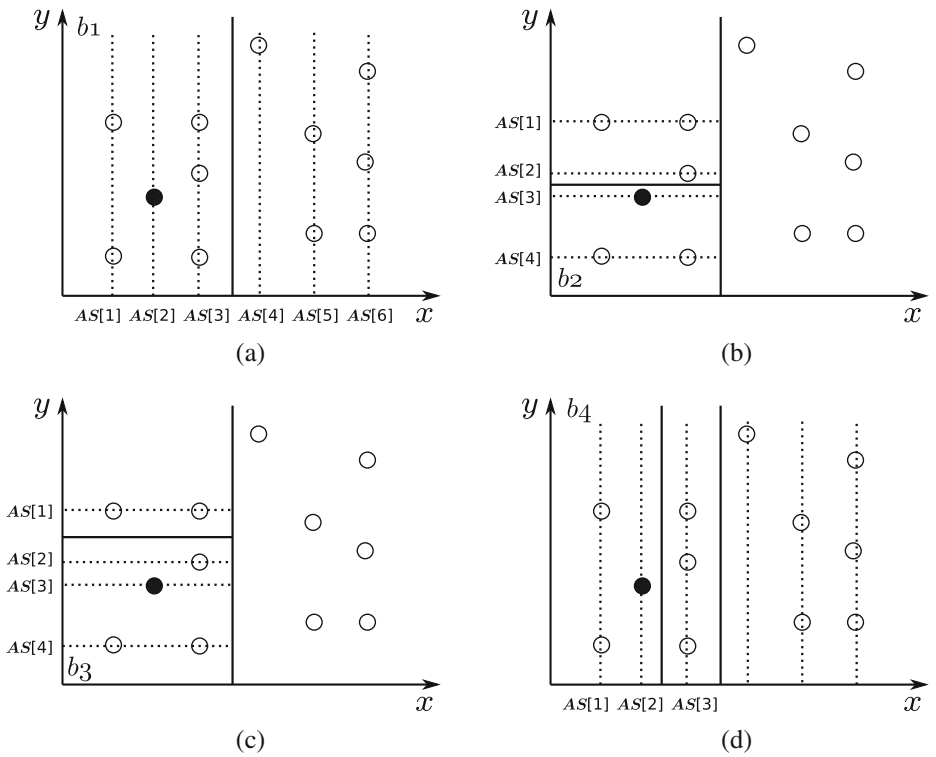


Fig. 5 An example execution of our algorithm uniformDP

Algorithm 3 The function updateAS.

```
1: FUNCTION: updateAS
2: INPUT: AS, req(qu), order
3: OUTPUT: AS' ⊆ AS that contains u and satisfies req(qu)
4:
5: AS := reorder(AS, order);
6: i := mid(AS, order);
7: if check(left(i), req(qu)) ∧ check(right(i), req(qu)) then
8:    AS := part(i, u);
9: else
10:   found := false;
11:   j := 0;
12:   while j ≤ len(order) ∧ ¬found do
13:      if check(left(j), req(qu)) ∧ check(right(j), req(qu)) then
14:         found := true;
15:         AS := part(j, u);
16:      else
17:         j := j + 1;
18:      end if
19:   end while
20: end if
21: return AS;
```

both of the two subsets satisfy $req(qu)$, then the one containing the issuer is returned (implemented by function part(i, u)). Otherwise, an iterative process is started. In jth iteration, the users are partitioned into two sets one of which contains the users in $AS[1], \ldots, AS[j]$ (denoted by left(j)) and the other contains the rest (denoted by right(j)). These two sets are checked against the privacy requirement $req(qu)$. If both left(j) and right(j) satisfy $req(qu)$, the one with issuer u is returned by part(j, u). If there are no partitions feasible after len$(order)$ iterations, the original set of users is returned.

An example execution of Algorithm 2 is shown in Fig. 5. The issuer is represented as a black dot. In Fig. 5a the users are first partitioned into two parts from the middle. Assume both parts satisfy $req(qu)$, so the set b_1 is returned as the anonymity set AS for the next iteration. As b_1's projection on axis y is longer, the first order is set to axis y (Fig. 5b). If after dividing the users from the middle, the set b_2 does not satisfy $req(qu)$. Thus, the users are partitioned from $AS[1]$ to $AS[4]$ (Fig. 5c). Suppose no partitions are feasible. The first order axis is then switched to axis x. Function updateAS is called again to find a partition along axis x (Fig. 5d).

We can see Algorithm 2 iterates for a number of times. In each iteration, some users are removed from the previous anonymity set. Operations such as partition and requirement check are time-linear in the size of the anonymity set. The number of iterations is logarithmic in the number of the users. So in the worst case, the time complexity of Algorithm 2 is $\mathcal{O}(N \log N)$, where N denotes the number of all users in $\mathcal{U}$. The correctness of Algorithm 2 is stated as Theorem 1.

Theorem 2 *For any query $\langle u, \ell, t, q\rangle$, Algorithm 2 computes a generalised region that satisfies the issuer u's privacy requirement $req(\langle u, whereis(u, t), t, q\rangle)$.*

Proof By Definitions 3 and 4, Algorithm 2 computes a region r for a query $\langle u, whereis(u,t), t, q\rangle$ that satisfies a constraint related to the issuer's posterior probability and the entropy about the issuer. We take α-USI as an example to show the correctness of our algorithm and the proofs of the other two are analogous.

By Definition 3, we have to prove the posterior probability of each user $u' \in u\ell(r,t)$ is smaller than α, i.e., $p(u'|\langle r,t,q\rangle, \mathcal{C}_t) \leq \alpha$. According to Eqs. 1 and 2, we need to prove for any $u' \in u\ell(r,t)$ (1) $f(\langle u', whereis(u',t), t, q\rangle) = \langle r,t,q\rangle$ and (2) his normalised a priori probability over those of all users in region r should be smaller than α, i.e.,

$$\frac{p(q|u', \mathcal{C}_t)}{\sum_{u'' \in \mathcal{U}} p(q|u'', \mathcal{C}_t)} \leq \alpha. \tag{5}$$

Let u' be any user in the generalised region r of Algorithm 2. Let AS_j and AS'_j be the values of AS in the jth iteration of Algorithm 2 of u and u', respectively. To prove (1), we show that $AS_j = AS'$ by induction on the number of iterations, i.e., j.

Induction basis: Initially, we suppose that $\mathcal{U}$ satisfied the requirement. Then we have $AS_1 = AS'_1$.

Induction step: Assume at jth iteration $AS_j = AS'_j$. We have to show that the algorithm either terminates with AS_j and AS'_j, or enters the next iteration with $AS_{j+1} = AS'_{j+1}$. The equality that $AS_j = AS'_j$ is followed by that $\mathsf{mid}(AS_j, order) = \mathsf{mid}(AS'_j, order)$. There are three possible executions.

Case 1 if $\mathsf{left}(i)$ and $\mathsf{right}(i)$ of AS_j and AS'_j satisfy the requirements (line 7 of Algorithm 3), the part containing the issuer is returned. Thus AS_{j+1} contains u as well as all other users in $u\ell(r,t)$, including u'. Thus, $AS_{j+1} = AS'_{j+1}$.

Case 2 if the check at line 7 of Algorithm 3 fails, then the algorithm switches to find from the beginning the first feasible partition. Suppose the partition is made at the position x for AS_j. Then x is also the right position for AS'_j as $AS_j = AS'_j$. Because of the similar reason in the previous possible execution, the same subset is set to AS_{j+1} and AS'_{j+1}. Thus, $AS_{j+1} = AS'_{j+1}$.

Case 3 if there are no possible partitions, Algorithm 3 returns AS_{j+1} and AS'_{j+1} in both cases. Then the first order is changed and Algorithm 3 is called again. If one of the first two execution is taken, with the analysis above, we have $AS_{j+1} = AS'_{j+1}$. Otherwise, Algorithm 2 terminates with $\mathsf{region}(AS_j)$ and $\mathsf{region}(AS'_j)$ which are equal.

We proceed with (2). Recall that the function $\mathsf{check}(AS, req(qu))$ returns true for metric α-USI only if Eq. 5 holds for each user in AS because it takes users' normalised a priori probabilities as their posterior probabilities. At the line 5 of Algorithm 2, we set AS to the original user set $\mathcal{U}$ and the algorithm continues only if the function $\mathsf{check}(\mathcal{U}, req(qu)$ returns true. Otherwise, it is impossible to return a region satisfying the requirement. The set AS is only reassigned to another set when a partition is made (line 8 or line 15 in Algorithm 3). For the two sets by the partition check all returns true and the one containing the issuer is assigned to AS. Thus, it is guaranteed that for each user $u' \in u\ell(r,t)$, Eq. 5 holds. □

8 Experimental results

We conduct experiments to evaluate our work from two aspects. First, we test the effectiveness of our framework in terms of the changes of issuers' posterior probabilities. In this way, we illustrate that users' personal profiles and request histories do cause privacy risks. Second, we implement our algorithms presented in Section 7 and with the experimental results we show and compare the characteristics of our new metrics proposed in Section 6.

To perform the experiments, we construct two sample datasets to simulate the spatial distributions of a collection of mobile users (*mobility dataset*) and their issued requests during movements (*request dataset*). We generate the mobility dataset using the moving object generator [6] and it consists of the trajectories of 38,500 users in a period with 50 discrete time points. We compose a series of request datasets corresponding to different numbers of *active users*. A user is called active if he subscribes certain LBSs and would issue requests during the period. Given a number of active users, we simulate a trace of requests for each of them according to his query dependency and his a priori preference on queries. Note that throughout the experiments, we do not distinguish users' a priori preferences from the a priori probabilities computed based on user profiles. This is because they are both static and a priori probabilities have already been considered in the calculation of a priori preferences. We assume 6 types of queries for users to choose. This makes users' a priori preference around 17 % on average. As we mentioned, our purpose is to evaluate the privacy risk incurred by contextual information and the effectiveness of the algorithms. Thus we assume that users' query dependency is available and generate it by a random procedure. Users' a priori preference is assessed in a similar way.

Our simulation is implemented with Java and run on a Linux laptop with 2.67 Ghz Intel Core (TM) and 4GB memory.

8.1 Impact of contextual information

We validate the effectiveness of our framework by checking if it can increase the likelihood of the adversary to correctly identify issuers by obtaining more contextual information. Given a generalised request, we can use the issuer's posterior probability as the measurement of the correctness of the adversary's attack on query privacy [44]. If a type of contextual information can help breach users' query privacy, then issuers will have larger posterior probabilities than those computed without the information on average. The main idea of our validation is to check whether the framework can capture this increase.

In our experiments, we construct three attack scenarios where k-anonymity spatial generalisation is deployed. In the first scenario, the adversary only learns the inherent contextual information while in the other two scenarios, users' a priori preferences and request histories are added sequentially to the adversary's knowledge. We denote the corresponding contextual information by $\mathcal{C}_t^{\mathsf{basic}}$, $\mathcal{C}_t^{\mathsf{pf}}$ and $\mathcal{C}_t^{\mathsf{dep}}$, respectively.

We define *correctness increase ratio* (CIR), and use it to quantify the increase of issuers' posterior probabilities when more contextual information is explored. Specifically, it is computed as the ratio of the increase over the posterior probabilities calculated without considering the contextual information. In this paper, we consider

two CIRs, i.e., Δp_{pf} and Δp_{dep}. For a generalised request $\langle r, t, q\rangle$ issued by u, they can be calculated as follows:

$$\Delta p_{\mathsf{pf}} = \frac{p(u|\langle r, t, q\rangle, \mathcal{C}_t^{\mathsf{pf}}) - p(u|\langle r, t, q\rangle, \mathcal{C}_t^{\mathsf{basic}})}{p(u|\langle r, t, q\rangle, \mathcal{C}_t^{\mathsf{basic}})}$$

where

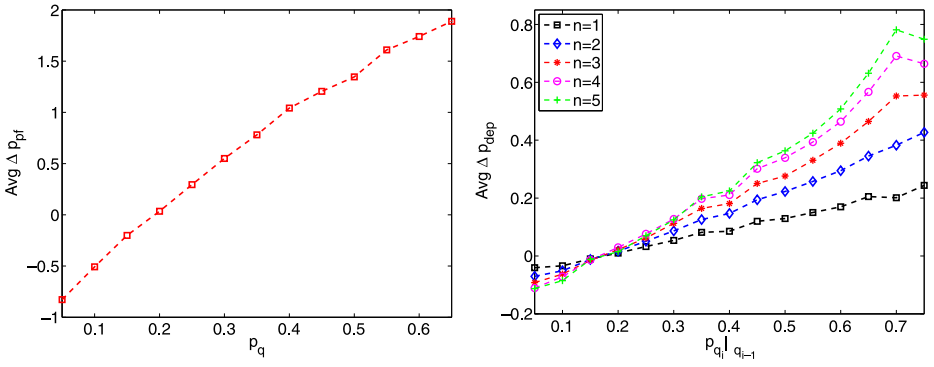
$$p(u|\langle r, t, q\rangle, \mathcal{C}_t^{\mathsf{basic}}) = \frac{1}{|\{u \in \mathcal{U}|whereis(u, t) \in r\}|},$$

and similarly,

$$\Delta p_{\mathsf{dep}} = \frac{p(u|\langle r, t, q\rangle, \mathcal{C}_t^{\mathsf{dep}}) - p(u|\langle r, t, q\rangle, \mathcal{C}_t^{\mathsf{pf}})}{p(u|\langle r, t, q\rangle, \mathcal{C}_t^{\mathsf{pf}})}.$$

In Fig. 6, we show how the correctness increase ratio changes with issuers' a priori preferences and the dependency between the last two queries. With respect to query dependency, we also illustrate the impact of the history window sizes (see Fig. 6b). The results are obtained by a simulation with 8,000 requests. We divide the requests into clusters according to the a priori preference or query dependency of the issuers when sending the requests. Specifically, we set $p_q = 0.05 \cdot cid$ where cid ($1 \leq cid \leq 20$) is the identifier of a cluster to be the maximum value of issuers' a priori preference allowed in the cluster cid. For example, if $p_q = 0.15$, the issuer of any request in the cluster has an a priori preference between 0.1 and 0.15 with respect to the issued query. Similarly, we define $p_{q_i|q_{i-1}} = 0.05 \cdot cid$ to represent the maximum query dependency allowed in cluster cid when the issuers issue the queries. Figure 6 depicts the average Δp_{pf} and Δp_{dep} of the generalised requests in each cluster satisfying k-anonymity with $k = 10$ and with 2.6 % of the users being active.

We observe that the curves in Fig. 6a and b follow two similar patterns. First, the CIR increases monotonically when ρ grows. Second, the average correctness increase ratio reaches 0 when the a priori preferences and query dependency fall into the interval between 0.15 and 0.2. This is due to the fact that users' average a priori



(a) Δp_{pf} vs. $p_u(q)$. (b) Δp_{dep} vs. $p(q_i \,|\, q_{i-1})$ and n.

Fig. 6 Impact of user profiles and query dependency on Δp

preference on each type of queries ($p_u(q_i)$) is around 17 %. With regard to Δp_{pf}, the issuer with an a priori preference of 0.17 will eliminate his difference from the other users in the same region as the average of their a priori preferences is also close to 0.17. For Δp_{dep}, the little difference between $p_u(q_i|q_{i-1})$ and $p_u(q_i)$ eliminates the influence of query dependency.

We can see that Δp_{dep} is also sensitive to the size of history windows in Fig. 6b. Larger history windows lead to bigger correctness increase ratios when the dependency between the last two queries (i.e., $p_u(q_i|q_{i-1})$) is bigger than 0.17. For instance, for the requests with query dependency between 0.3 and 0.4, the average value of Δp_{dep} increases by 0.051, 0.036, 0.031 when n grows from 1 to 2, from 2 to 3, from 3 to 4, respectively. By more experiments with larger n, we can show that bigger window sizes do not necessarily lead to more privacy leakage. For instance, when n is set to 5, the average increase of CIR is 0.029 which is almost the same as the case of $n = 4$.

From the above discussion, we can conclude that if a user issues a query with a large preference value or high dependency on the last queries, he will have less privacy if the adversary adopts our framework. This also shows that our framework is useful to increase the likelihood of attackers to correctly learn the real issuers although we have negative CIRs when users issue queries independently from their profiles or last queries. This is because in most of the cases, users' behaviour should be consistent with their profiles and past habits.

Beside the size of history windows, the number of active users has impact on Δp_{dep} as well. It decreases when there are more active users issuing LBS requests, but the influence becomes smaller with larger history windows. Figure 7 shows that the average Δp_{dep} decreases by 30 %, 24 % 19 % and 18 % for $n = 1, 2, 3$ and 4, respectively, when the percentage of active users increases from 2.5 % to 7.5 %. This is because more active users lead to more observed requests added into users' observed request traces and mixed with users' real requests, while bigger history windows have larger chances to include users' real requests.

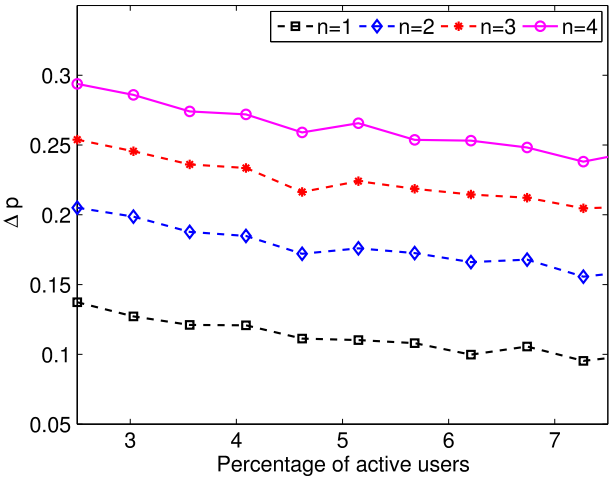


Fig. 7 Δp vs. #active users and n

8.2 Effectiveness of the new privacy metrics

In this section we discuss the features of our privacy metrics in terms of (1) area of the generalised regions and (2) issuers' posterior probabilities. To compare the metrics presented in Section 6, we define a normalised value *norm*: $norm = k$ for query-dependent k-ABS while $norm = 2^{\beta}$ for β-EBA and $norm = \frac{1}{\alpha}$ for α-USI. In the following experiments, we take $\mathcal{C}_t^{\mathsf{dep}}$ as the knowledge of the adversary due to its large coverage of contextual information.

Experiment setting We set the percentage of active users to 2.6 % and use the first 1,000 requests after 8,000 requests have been observed. Each number shown in the following discussion is an average of the 1,000 samples.

Recall that in the generalisation algorithm kABS (see Algorithm 1) we make use of a clustering algorithm to calculate the set of users with similar a priori probabilities. Clustering has been extensively studied in the literature and a number of clustering algorithms have been proposed to satisfy different properties, e.g., density-based and distribution-based [23]. In the case of generalising LBS requests, the chosen clustering algorithm should satisfy at least two properties. First, the clustering algorithm should be efficient because LBS responses need to be sent back to uses in real time. Second, we need a strict partitioning clustering algorithm as each user should belong to exact one cluster.

In the implementation of kABS, we use the K-means clustering algorithm [31]. This is mainly due to its linear time complexity with the number of users. Its main idea is to choose K centroids, one for each cluster. In our algorithm, the K centroids are selected randomly among the users. Then each user is associated to the nearest centroid according the difference between their a priori probabilities, which results in K clusters. The centroids of these K clusters are updated as the new centroids based on which all users re-calculate their centroids to associate. The process continues until the centroids remain unchanged between two consecutive iterations. In our case, K is selected and fixed by the anonymiser. In fact, it defines the 'similarity' in the definition of k-ABS in Section 6, i.e., ϵ. The larger K is, the smaller ϵ becomes.

In order to determine a proper value of K, we run our kABS algorithm by assigning different values to K. In Fig. 8, we show the changes of the average distance between any two users' a priori probabilities in the calculated clusters and the area of the generalised regions along with K. It can be seen that a larger K enables users to have closer a priori probabilities but leads to larger generalised areas. In addition, the area increases faster than the decrease of the distance. Considering the relatively small generalised regions and the similarity between users in the resulted clusters, we set K to 10 in the following experiments.

Impact of history window sizes From the above discussion, we learn that users will have less query privacy when larger history windows are used in our framework. Figure 9 shows how issuers' posterior probabilities and the area of generalised regions change according to the normalised value *norm* and the history window size n. Note that when $n = 0$, the generalisation algorithm only considers users' a priori preference.

For k-ABS, issuers' posterior probabilities are about $\frac{1}{k}$ as the generalised regions have at least k users with similar posterior probabilities. However, after taking a

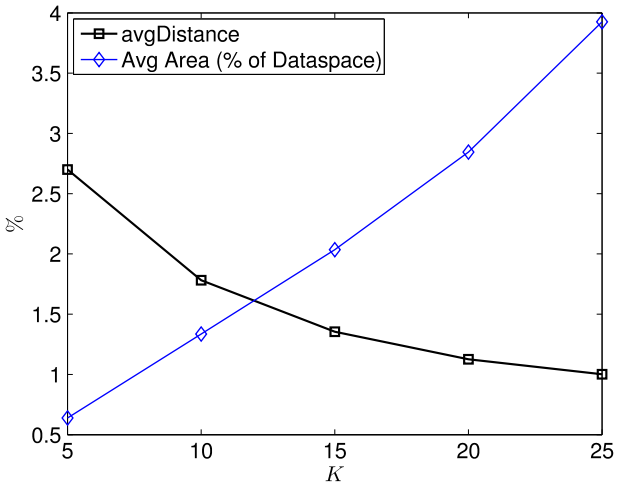


Fig. 8 The impact of K

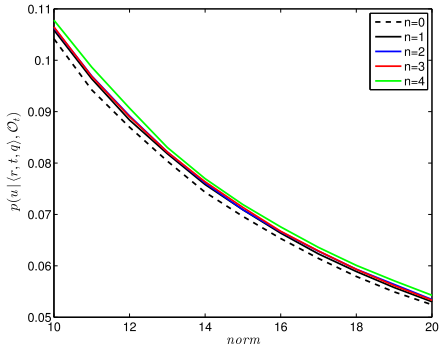
closer look, we can find that a larger n leads to a larger distance to $\frac{1}{k}$. This is because larger history windows make the issuers' posterior probabilities more different from the others, which in turn makes it more difficult to find users with similar posterior probabilities. This also explains why the generalised regions become larger with larger history windows as shown in Fig. 9b.

For α-USI, issuers' posterior probabilities are always below $\frac{1}{norm}$, which satisfies its definition (see Fig. 9c). Moreover, issuers' posterior probabilities become larger when more historical observed requests are explored. However, the area of generalised regions differs little between different history window sizes (see Fig. 9d). This is because the increase of the posterior probabilities is too small to initiate the computation of a new region.

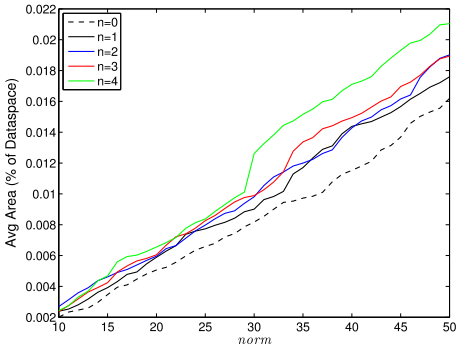
For β-EBA, issuers' posterior probabilities can remain almost unchanged in some segments of the curves. The projection of the middle point of such a segment on axis *norm* has an logarithm of integer, such as 16 and 32 (see in Fig. 9e). Similar to k-ABS, larger history windows increase the issuers' posterior probabilities, which leads to smaller entropy. This can be seen from Fig. 9f where the generalised regions of larger n double their sizes earlier than the regions of smaller n.

We can also observe from Fig. 9 that for the same value of *norm*, although the metric β-EBA cannot always ensure issuers' posterior probabilities as close to $\frac{1}{k}$ as k-ABS, the area of generalised regions is about ten times smaller than that of k-ABS and only half of that of α-USI. Since bigger regions lead to worse quality of service, this indicates that a balance between privacy protection and quality of services needs to be considered in practice.

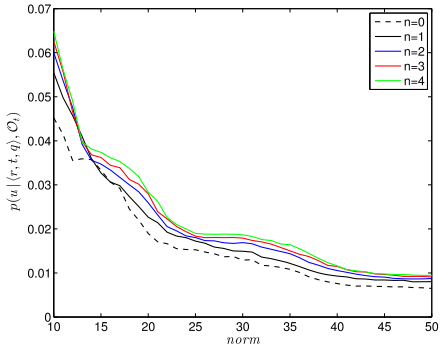
Impact of query dependency The protection of issuers' privacy varies with issuers' query dependency. Figure 10 plots posterior probabilities and average area of generalised regions for issuers with different levels of query dependency. The results are collected with the history window size $n = 3$. Our general observation is that issuers with larger dependencies have bigger posterior probabilities and larger generalised regions.



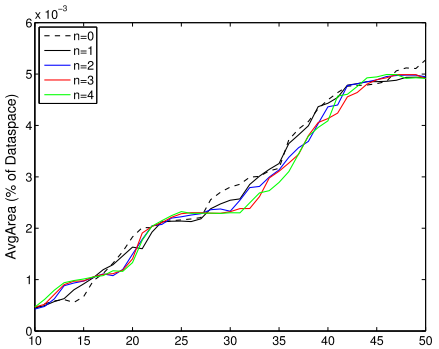
(a) $p(u \mid \langle r, t, q \rangle, \mathcal{O}_t)$ vs. n (k-ABS).



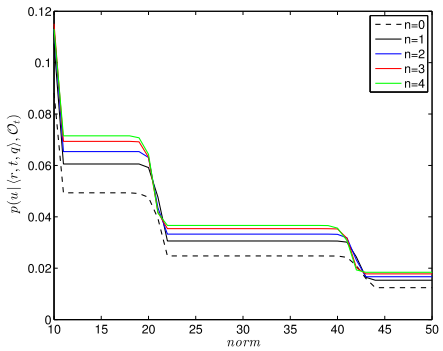
(b) Average area vs. n (k-ABS).



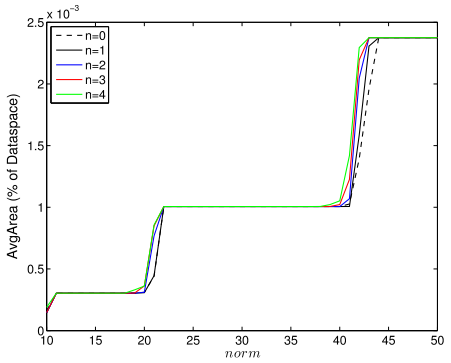
(c) $p(u \mid \langle r, t, q \rangle, \mathcal{O}_t)$ vs. n (α-USI).



(d) Average area vs. n (α-USI).



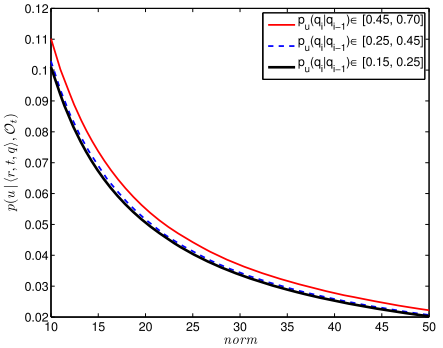
(e) $p(u \mid \langle r, t, q \rangle, \mathcal{O}_t)$ vs. n (β-EBA).



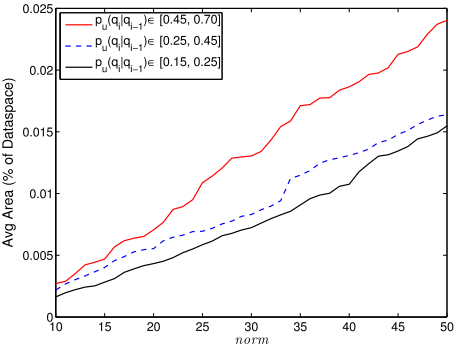
(f) Average area vs. n (β-EBA).

Fig. 9 Impact of history window size n

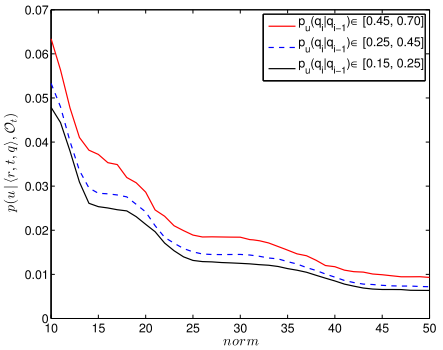
Table 2 summarises the corresponding average increases (in percentage) for issuers with high ($\geq$0.45) and medium (0.25–0.45) dependencies, when compared with those with low dependencies ($\leq$0.25). The table shows that posterior probabilities of the issuers, when β-EBA is used, are more sensitive to the degree of dependency (43.1 % increase for high-level dependency), while the generalised regions are more



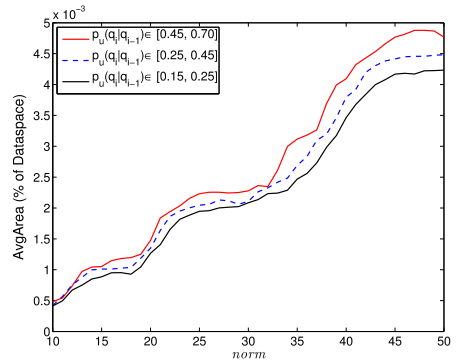
(a) $p(u \mid \langle r, t, q \rangle, \mathcal{O}_t)$ vs. $p(q_i \mid q_{i-1})$ (k-ABS).



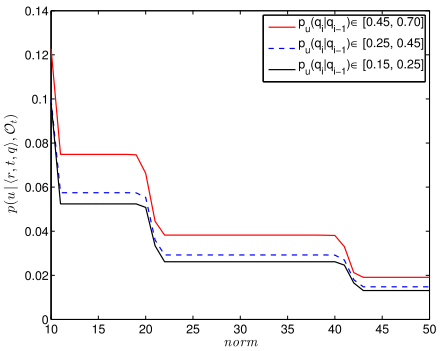
(b) Average area vs. $p(q_i \mid q_{i-1})$ (k-ABS).



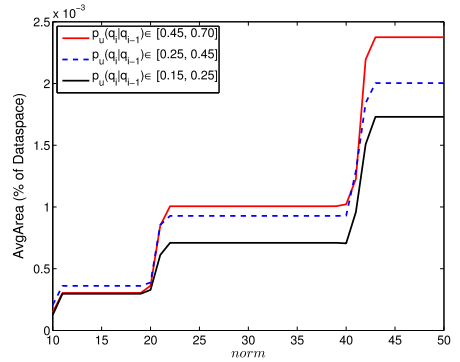
(c) $p(u \mid \langle r, t, q \rangle, \mathcal{O}_t)$ vs. $p(q_i \mid q_{i-1})$ (α-USI).



(d) Average area vs. $p(q_i \mid q_{i-1})$ (α-USI).



(e) $p(u \mid \langle r, t, q \rangle, \mathcal{O}_t)$ vs. $p(q_i \mid q_{i-1})$ (β-EBA).



(f) Average area vs. $p(q_i \mid q_{i-1})$ (β-EBA).

Fig. 10 Impact of dependency $p(q_i|q_{i-1})$

sensitive to dependency (62.9 % increase for high-level dependency) when k-ABS is used.

Performance of the proposed generalisation algorithm In Fig. 11, we present the performance of our generalisation algorithms to deal with users' various requirements. For the sake of comparison, we show in Fig. 11 the performance of the algorithms

Table 2 Increases in posterior probabilities and average area of generalised regions

	k-ABS		β-EBA		α-USI	
	Medium (%)	High (%)	Medium (%)	High (%)	Medium (%)	High (%)
Posterior prob.	2.1	9.5	11.1	43.1	11.9	40.0
Avg area	21.3	62.9	23.3	30.1	10.7	19.1

when contextual information is set to $\mathcal{C}_t^{\mathsf{pf}}$ and $\mathcal{C}_t^{\mathsf{dep}}$, respectively. The computation time recorded is the average time per request based on executions with the same 100 requests.

As discussed in Section 7, it is necessary to update the status of each user when dynamic contextual information is explored. For instance, observed request traces and the corresponding posterior probabilities have to be updated for each request when $\mathcal{C}_t^{\mathsf{dep}}$ is used. This is time-consuming, especially when the initial region is huge and contains a large number of users. In our implementation, we reduce the computation overhead by restricting the size of initial regions. The number of users located in an initial region is fixed as ten times as many as what users require for. For instance, for k-ABS, if $k = 10$, then we first call k-anonymity generalisation algorithm to get an initial region with 100 users. As the generalisation algorithm is deterministic, which means for any user in a generalised region, it always returns the same region. Thus, our implementation does not have the "outlier" problem [33].

From Fig. 11, we can see that the computation time increases as *norm* gets bigger. This is because the algorithm has to consider larger initial regions and more users are involved in the calculation of dependency-based posterior probabilities. For β-EBA and α-USI, about 20 ms are needed when $norm = 50$, while k-ABS requires more time (around 35 ms) as the K-means clustering algorithm is executed first to find similar users. When compared to the original algorithms, the computation time

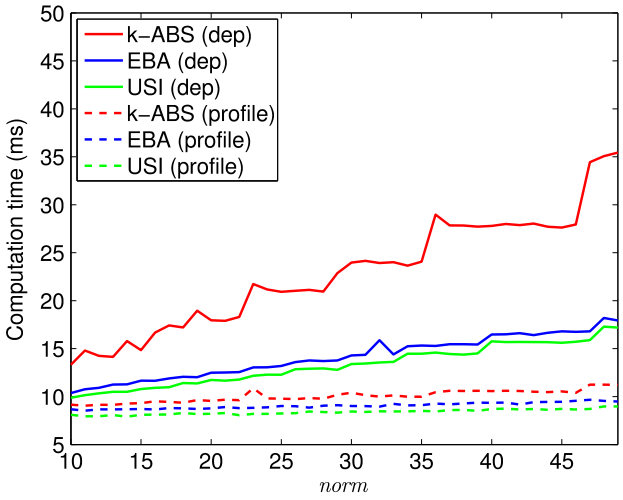


Fig. 11 Average computational time (history window $n = 3$)

increases by about two times for β-EBA and α-USI while it is about four times for k-ABS when $norm = 50$.

There are some ways to improve the efficiency of our implementation. For instance, we can use better data structures to maintain users' status. We can expect that with a powerful anonymiser our algorithms are efficient enough to handle concurrent requests and give real-time responses.

9 Conclusion

In this paper, we have developed a formal framework for query privacy analysis exploring contextual information. In the framework, we systematically categorise contextual information and propose a probabilistic way to model the adversary's attacks on query privacy. Specifically, we use a posterior probability distribution to describe the knowledge learnt by the adversary about the issuers after the analysis. This interpretation allows us to define new metrics for query privacy from different perspectives, which also facilitate users to flexibly and precisely express their privacy requirement.

We took two types of contextual information to exemplify the application of our framework. One application focuses on user profiles while the other one is further extended with contextual information—query dependency, which has not been investigated in the literature. To protect query privacy we have designed new spatial generalisation algorithms to generalise requests which can satisfy users' privacy requirements in various metrics.

Through experiments, we have shown (1) our framework is effective to increase the correctness of the adversary's guess on real issuers; (2) the newly identified query dependency does cause privacy leakage about users' queries; (3) the proposed metrics are effective to protect users' query privacy; and (4) the generalisation algorithms are efficient.

For experiments, we made use of simulated datasets about users' movements and request traces due to the lack of real-life data with respect to LBSs. This causes some difficulties for us to test the impact of time intervals between requests. As part of our future work, we want to check whether we can collect and use users' logs in Geo-social networks in order to have a more comprehensive validation of our work.

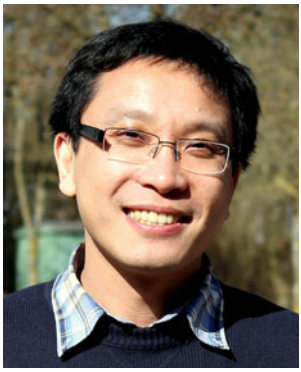
References

1. Ariely D, Au WT, Bender RH, Budescu DV, Dietz CB, Gu H, Wallsten TS, Zauberman G (2000) The effects of averaging subjective probability estimates between and within judges. J Exp Psychol Appl 6:130–147
2. Bellavista P, Küpper A, Helal S (2008) Location-based services: back to the future. IEEE Pervasive Comput 7(2):85–89
3. Beresford AR (2005) Location privacy in ubiquitous computing. PhD thesis, University of Cambridge
4. Bettini C, Mascetti S, Wang XS, Freni D, Jajodia S (2009) Anonymity and historical *k*-anonymity in location-based services. In: Privacy in location-based applications. Lecture Notes in Computer Science, vol 5599. Springer, pp 1–30

5. Bolger F, Wright G (1993) Coherence and calibration in expert probability judgement. Omega 21(6):629–644
6. Brinkhoff T (2002) A framework for generating network-based moving objects. Geoinformatica 6(2):153–180
7. Chen X, Pang J (2012) Measuring query privacy in location-based services. In: Proc. 2nd ACM conference on data and application security and privacy (CODASPY). ACM Press, pp 49–60
8. Chen X, Pang J (2013) Exploring dependency for query privacy protection in location-based services. In: Proc. 3rd ACM conference on data and application security and privacy (CODASPY). ACM Press, pp 37–47
9. Chen X, Pang J, Xue R (2013) Constructing and comparing user mobility profiles for location-based services. In: Proc. 28th ACM Symposium on Applied Computing (SAC). ACM Press, pp 261–266
10. Cheng R, Zhang Y, Bertino E, Prabhakar S (2006) Preserving user location privacy in mobile data management infrastructures. In: Proc. 6th international workshop on Privacy Enhancing Technologies (PET). Lecture Notes in Computer Science, vol 4258. Springer, pp 393–412
11. Chow CY, Mokbel MF, Aref WG (2009) Casper*: query processing for location services without compromising privacy. ACM Trans Database Syst 34(4):1–48
12. Devroye, L., Lugosi, G.: Combinatorial methods in density estimation. Springer (2001)
13. Dewri R, Ray I, Ray I, Whitley D (2010) On the formation of historically k-anonymous anonymity sets in a continuous LBS. In: Proc. 6th international conference on security and privacy in communication networks (SecureComm). Lecture Notes in Computer Science, vol 50. Springer, pp 71–88
14. Dewri, R., Ray, I., Ray, I., Whitley, D.: Query *m*-invariance: preventing query disclosures in continuous location-based services. In: Proc. 11th international conference on Mobile Data Management (MDM). IEEE Computer Society, pp 95–104 (2010)
15. Díaz C, Seys S, Claessens J, Preneel B (2003) Towards measuring anonymity. In: Proc. 2nd international workshop on Privacy Enhancing Technologies (PET). Lecture Notes in Computer Science, vol 2482. Springer, pp 54–68
16. Gedik B, Liu L (2008) Protecting location privacy with personalized *k*-anonymity: architecture and algorithms. IEEE Trans Mob Comput 7(1):1–18
17. Ghinita G, Kalnis P, Khoshgozaran A, Shahabi C, Tan KL (2008) Private queries in location based services: anonymizers are not necessary. In: Proc. the ACM SIGMOD international conference on management of data. ACM Press, pp 121–132
18. Ghinita G, Kalnis P, Skiadopoulos S (2007) PRIVE: anonymous location-based queries in distributed mobile systems. In: Proc. 16th international conference on World Wide Web (WWW). ACM Press, pp 371–380
19. Giannotti F, Nanni M, Pedreschi D, Pinelli F (2006) Mining sequences with temporal annotations. In: Proc. 21st ACM Symposium on Applied Computing (SAC). ACM Press, pp 593–597
20. Giannotti F, Nanni M, Pedreschi D, Pinelli F, Axiak M (2007) Trajectory pattern mining. In: Proc. 13th ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD). ACM Press, pp 330–339
21. González MC, Hidalgo CA, Barabási AL (2008)Understanding individual human mobility patterns. Nature 453:779–782
22. Gruteser M, Grunwald D (2003) Anonymous usage of location-based services through spatial and temporal cloaking. In: Proc. 1st international conference on Mobile Systems, applications, and services (MobiSys). USENIX Association
23. Han J, Kamber M (2000) Data mining: concepts and techniques. Morgan Kaufmann
24. Hoh B, Gruteser M, Xiong H, Alrabady A (2007) Preserving privacy in GPS traces via uncertainty-aware path cloaking. In: Proc. 14th ACM conference on Computer and Communications Security (CCS). ACM Press, pp 161–171
25. Jaynes ET (1957) Information theory and statistical mechanics. Phys Rev 106(4):620–630
26. Jaynes ET (1957) Information theory and statistical mechanics II. Phys Rev 108(2):171–190
27. Kalnis P, Ghinita G, Mouratidis K, Papadias D (2007) Preventing location-based identity inference in anonymous spatial queries. IEEE Trans Knowl Data Eng 19(12):1719–1733

28. Kido H, Yanagisawa Y, Satoh T (2005) Protection of location privacy using dummies for location-based services. In: Proc. 21st International Conference on Data Engineering (ICDE). IEEE Computer Society, pp 12–48
29. Li N, Li T, Venkatasubramanian S (2007) *t*-closeness: privacy beyond *k*-anonymity and *l*-diversity. In: Proc. 23rd International Conference on Data Engineering (ICDE). IEEE Computer Society, pp 106–115
30. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M (2007) ℓ-diversity: Privacy beyond *k*-anonymity. ACM Trans Knowl Discov Data 1(1), Article 3
31. MacQueen JB (1967) Some methods for classification and analysis of multivariate observations. In: Proc. 5th Berkeley symposium on mathematical statistics and probability, vol 1. University of California Press, pp 281–297
32. Manning C, Schuütze H (1999) Foundations of statistical natural language processing. Cambridge
33. Mascetti S, Bettini C, Freni D, Wang XS (2007) Spatial generalization algorithms for LBS privacy preservation. J Location Based Services 1(3):179–207
34. Mokbel MF, Chow CY, Aref WG (2007) The new casper: a privacy-aware location-based database server. In: Proc. 23rd International Conference on Data Engineering (ICDE). IEEE Computer Society, pp 1499–1500
35. Rebollo-Monedero D, Parra-Arnau J, Díaz C, Forné J (2013) On the measurement of privacy as an attacker's estimation error. Int J Inf Secur 12(2):129–149
36. Reiter MK, Rubin AD (1998) Crowds: anonymity for web transactions. ACM Trans Inf Syst Secur 1(1):66–92
37. Riboni D, Pareschi L, Bettini C (2008) Privacy in georeferenced context-aware services: a survey. In: Proc. 1st international workshop on Privacy in Location-Based Applications (PiLBA). CEUR Workshop Proceedings, vol 397. CEUR
38. Riboni D, Pareschi L, Bettini C, Jajodia S (2009) Preserving anonymity of recurrent location-based queries. In: Proc. 16th international symposium on temporal representation and reasoning (TIME). IEEE Computer Society, pp 62–69
39. Samarati P (2001) Protecting respondents' identities in microdata release. IEEE Trans Knowl Data Eng 13(6):1010–1027
40. Santos F, Humbert M, Shokri R, Hubaux JP (2011) Collaborative location privacy with rational users. In: Proc. 2nd international conference on decision and game theory for security (GameSec). Lecture Notes in Computer Science, vol 7037. Springer, pp 163–181
41. Serjantov A, Danezis G (2003) Towards an information theoretic metric for anonymity. In: Proc. 2nd international workshop on Privacy Enhancing Technologies (PET). Lecture Notes in Computer Science, vol 2482. Springer, pp 41–53
42. Shin H, Atluri V, Vaidya J (2008) A profile anonymization model for privacy in a personalized location based service environment. In: Proc. 9th international conference on Mobile Data Management (MDM). IEEE Computer Society, pp 73–80
43. Shin H, Atluri V, Vaidya J (2011) A profile anonymization model for location-based services. J Comput Secur 19(5):795–833
44. Shokri R, Theodorakopoulos G, Boudec JYL, Hubaux JP (2011) Quantifying location privacy. In: Proc. 32nd IEEE symposium on Security and Privacy (S&P). IEEE Computer Society
45. Shokri R, Troncoso C, Díaz C, Freudiger J, Hubaux JP (2010) Unraveling an old cloak: *k*-anonymity for location privacy. In: Proc. 2010 ACM Workshop on Privacy in the Electronic Society (WPES). ACM Press, pp 115–118
46. Tan KW, Lin Y, Mouratidis K (2009) Spatial cloaking revisited: distinguishing information leakage from anonymity. In: Proc. 11th international Symposium on Spatial and Temporal Databases (SSTD). Lecture Notes in Computer Science, vol 5644. Springer, pp 117–134
47. Xu T, Cai Y (2009) Feeling-based location privacy protection for location-based services. In: Proc. 16th ACM conference on Computer and Communications Security (CCS). ACM Press, pp 348–357
48. Xue M, Kalnis P, Pung HK (2009) Location diversity: enhanced privacy protection in location based services. In: Proc. 4th international symposium on Location and Context Awareness (LoCA). Lecture Notes in Computer Science, vol 5561. Springer, pp 70–87
49. Yiu ML, Jensen CS, Huang X, Lu H (2008) Spacetwist: managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In: Proc. 24th International Conference on Data Engineering (ICDE). IEEE Computer Society, pp 366–375

Xihui Chen a PhD student at University of Luxembourg within the Security and Trust of Software Systems (http://satoss.uni.lu) research group under the supervision of Prof. Dr. Sjouke Mauw. He was funded by the Luxembourgish national funding agency (FNR), in cooperation with a Luxembourgish company (itrust) and the European Space Agency (ESA) since August 2010. His research project is "Secure and Private Location Proofs: Architecture and Design for Location-based Services".



Jun Pang is a senior researcher in the "Security and Trust of Software Systems" research group at University of Luxembourg within the Computer Science and Communications Research Unit.

He received a PhD in Computer Science from Free University Amsterdam, based on his research performed at CWI in Amsterdam as a junior researcher. Before joining the University of Luxembourg in 2008, he was research assistant at the Computer Science Department of University of Oldenburg in Germany, where he contributed to the DFG transregional collaborative research center AVACS (Automatic Verification and Analysis of Complex Systems) and developed theory, techniques and tools for verification of hybrid systems. Before Oldenburg, he was postdoctoral researcher at the Comète project (Concurrency, Mobility and Transactions) of INRIA Futurs and Ecole Polytechnique in France, where he studied quantitative aspects of computation including semantics for timed and probabilistic process algebras and applied probabilistic concurrency theory for the analysis of anonymity.

His research interests are mainly in formal methods and computer security, e.g., process algebra, model checking, probabilistic systems, and formal verification with a strong focus on application of formal techniques to security and privacy in large and distributed systems. He has been working on formalising security and privacy properties in different application domains such as electronic voting, online auction and healthcare, and developing algorithms to support automatic verification. More recently, he is working on an ESA project on "Location Assurance Service Provider" and supervising an FNR-AFR PhD project on "Secure and Private Location Proofs". Within these two projects, he starts to work on quantitative and statistical aspects of privacy in location-based services.

Dr Pang was and is member of many scientific committees (e.g., ICTAC'13, FHIES'13, TASE'13, ACCESS'13, ICECCS'13, ICFEM'13, ATVA'12), and organized a number of conferences and workshops (e.g., Luxembourg Day on Security and Reliability'09, SecCo'11, SAC-SVT'12, SAC-SVT'13, ESSS'13). He has co-organized the international summer school "Verification Technology, Systems & Applications" since 2009. At the University of Luxembourg, he has supervised 11 Master students and 3 PhD students. He has published more than 80 refereed publications in international competitive conferences/workshops and prestige journals. His h-index is 17 according to citation statistics by Google Scholar.