

A Predictive Differentially-Private Mechanism for Mobility Traces.

Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Marco Stronati

CNRS, INRIA, LIX Ecole Polytechnique

Abstract. With the increasing popularity of GPS-enabled handheld devices, location based applications and services have access to accurate and real-time location information, raising serious privacy concerns for their millions of users. Trying to address these issues, the notion of *geo-indistinguishability* was recently introduced, adapting the well-known concept of Differential Privacy to the area of location-based systems. A Laplace-based obfuscation mechanism satisfying this privacy notion works well in the case of a *sporadic* use; Under repeated use, however, *independently* applying noise leads to a quick loss of privacy due to the correlation between the location in the trace.

In this paper we show that correlations in the trace can be in fact exploited in terms of a *prediction function* that tries to guess the new location based on the previously reported locations. The proposed mechanism tests the quality of the predicted location using a private test; in case of success the prediction is reported otherwise the location is sanitized with new noise. If there is considerable correlation in the input trace, the extra cost of the test is small compared to the savings in budget, leading to a more efficient mechanism.

We evaluate the mechanism in the case of a user accessing a location-based service while moving around in a city. Using a simple prediction function and two budget spending strategies, optimizing either the utility or the budget consumption rate, we show that the predictive mechanism can offer substantial improvements over the independently applied noise.

1 Introduction

In recent years, the popularity of devices capable of providing an individual's position with a range of accuracies (e.g. wifi-hotspots, GPS, etc) has led to a growing use of "location-based systems" that record and process location data. A typical example of such systems are Location Based Services (LBSs) – such as mapping applications, Points of Interest retrieval, coupon providers, GPS navigation, and location-aware social networks – providing a service related to the user's location. Although users are often willing to disclose their location in order to obtain a service, there are serious concerns about the privacy implications of the constant disclosure of location information.

In this paper we consider the problem of a user accessing a LBS while wishing to hide his location from the service provider. We should emphasize that, in contrast to several works in the literature [1,2], we are interested not in hiding the user's *identity*, but instead his *location*. In fact, the user might be actually authenticated to the provider, in order to obtain a personalized service (personalized recommendations, friend information from a social network, etc); still he wishes to keep his location hidden.

Several techniques to address this problem have been proposed in the literature, satisfying a variety of location privacy definitions. A widely-used such notion is k -anonymity (often called l -diversity in this context), requiring that the user's location is indistinguishable among a set of k points. This could be achieved either by adding *dummy locations* to the query [3,4], or by creating a *cloaking region* including k locations with some semantic property, and querying the service provider for that cloaking region [5,6,7]. A different approach is to report an *obfuscated* location z to the service provider, typically obtained by adding random noise to the real one. Shokri et al. [8] propose a method to construct an obfuscation mechanism of optimal privacy for a given quality loss constraint, where privacy is measured as the expected error of a Bayesian adversary trying to guess the user's location [9].

The main drawback of the aforementioned location privacy definitions is that they depend on the adversary's background knowledge, typically modeled as a prior distribution on the set of possible locations. If the adversary can rule out some locations based on his prior knowledge, then k -anonymity will be trivially violated. Similarly, the adversary's expected error directly depends on his prior. As a consequence, these definitions give no precise guarantees in the case when the adversary's prior is different.

Differential privacy [10] was introduced for statistical databases exactly to cope with the issue of prior knowledge. The goal in this context is to answer aggregate queries about a group of individuals without disclosing any individual's value. This is achieved by adding random noise to the query, and requiring that, when executed on two databases x, x' differing on a single individual, a mechanism should produce the same answer z with similar probabilities. Differential privacy has been successfully used in the context of location-based systems [11,12,13] when *aggregate* location information about a large number of individuals is published. However, in the case of a single individual accessing an LBS, this property is too strong, as it would require the information sent to the provider to be independent from the user's location.

Our work is based on "*geo-indistinguishability*", a variant of differential privacy adapted to location-based systems, introduced recently in [14]. Based on the idea that the user should enjoy strong privacy within a small radius, and weaker as we move away from his real location, *geo-indistinguishability* requires that the closer (geographically) two locations are, the more indistinguishable they should be. This means that when locations x, x' are close they should produce the same reported location z with similar probabilities; however the probabilities can become substantially different as the distance between x and x' increases. This property can be achieved by adding noise to the user's location drawn from a 2-dimensional Laplace distribution.

In practice, however, a user rarely performs a *single* location-based query. As a motivating example, we consider a user in a city performing different activities throughout the day: for instance he might have lunch, do some shopping, visit friends, etc. During these activities, the user performs several queries: searching for restaurants, getting driving directions, finding friends nearby, and so on. For each query, a new obfuscated location needs to be reported to the service provider, which can be easily obtained by independently adding noise at the moment when each query is executed. We refer to independently applying noise to each location as the *independent mechanism*.

However, it is easy to see that privacy is degraded as the number of queries increases, due to the *correlation* between the locations. Intuitively, in the extreme case when the user never moves (i.e. there is perfect correlation), the reported locations are centered around the real one, completely revealing it as the number of queries increases. Technically, the independent mechanism applying ϵ -geo-indistinguishable noise (where ϵ is a privacy parameter) to n location can be shown to satisfy $n\epsilon$ -geo-indistinguishability [14]. This is typical in the area of differential privacy, in which ϵ is thought as a privacy *budget*, consumed by each query; this linear increase makes the mechanism applicable only when the number of queries remains small. Note that any obfuscation mechanism is bound to cause privacy loss when used repeatedly; geo-indistinguishability has the advantage of directly quantifying this loss terms of the consumed budget.

The goal of this paper is to develop a *trace obfuscation* mechanism with a smaller *budget consumption rate* than applying independent noise. The main idea is to actually use the correlation between locations in the trace to our advantage. Due to this correlation, we can often *predict* a point close to the user's actual location from information previously revealed. For instance, when the user performs multiple different queries from the same location - e.g. first asking for shops and later for restaurants - we could intuitively use the same reported location in all of them, instead of generating a new one each time. However, this implicitly reveals that the user is not moving, which violates geo-indistinguishability (nearby locations produce completely different observations); hence the decision to report the same location needs to be done in a private way.

Our main contribution is a *predictive mechanism* with three components: a *prediction function* Ω , a *noise mechanism* N and a *test mechanism* Θ . The mechanism behaves as follows: first, the list of previously reported locations (i.e. information which is already public) are given to the prediction function, which outputs a predicted location \tilde{z} . Then, it tests whether \tilde{z} is within some threshold l from the user's current location using the test mechanism. The test itself should be private: nearby locations should pass the test with similar probabilities. If the test succeeds then \tilde{z} is reported, otherwise a new reported location is generated using the noise mechanism.

The advantage of the predictive mechanism is that the budget is consumed only when the test or noise mechanisms are used. Hence, if the prediction rate is high, then we will only need to pay for the test, which can be substantially cheaper in terms of budget. The configuration of N and Θ is done via a *budget manager* which decides at each step how much budget to spend on each mechanism. The budget manager is also allowed to completely skip the test and blindly accept or reject the prediction, thus saving the corresponding budget. The flexibility of the budget manager allows for a dynamic behavior, constantly adapted to the mechanism's previous performance. We examine in detail two possible budget manager strategies, one maximizing utility under a fixed budget consumption rate and one doing the exact opposite, and explain in detail how they can be configured.

Note that, although we exploit correlation for efficiency, the predictive mechanism is shown to be private independently from the prior distribution on the set of traces. If the prior presents correlation, and the prediction function takes advantage of it, the mechanism can achieve a good budget consumption rate, which translates either to better utility or to a greater number of reported points than the independent mechanism. If

there is no correlation, or the prediction does not take advantage of it, then the budget consumption can be worse than the independent mechanism. Still, thanks to the arbitrary choice of the prediction function and the budget manager, the predictive mechanism is a powerful tool that can be adapted to a variety of practical scenarios.

We experimentally verify the effectiveness of the mechanism on our motivating example of a user performing various activities in a city, using two large data sets of GPS trajectories in the Beijing urban area ([15,16]). The results for both budget managers, with and without the skip strategy, show considerable improvements with respect to independently applied noise. More specifically, we are able to decrease average error up to 40% and budget consumption rate up to 64%. The improvements are significant enough to broaden the applicability of geo-indistinguishability to cases impossible before: in our experiments we cover 30 queries with reasonable error which is enough for a full day of usage; alternatively we can drive the error down from 5 km to 3 km, which make it acceptable for a variety of application.

Note that our mechanism can be efficiently implemented on the user’s phone, and does not require any modification on the side of the provider, hence it can be seamlessly integrated with existing LBSs.

Contributions The paper’s contributions are the following:

- We propose a predictive mechanism that exploits correlations on the input by means of a prediction function.
 - We show that the proposed mechanism is private and provide a bound on its utility.
 - We instantiate the predictive mechanism for location privacy, defining a prediction function and two budget managers, optimizing utility and budget consumption rate.
 - We evaluate the mechanism on two large sets of GPS trajectories and confirm our design goals, showing substantial improvements compared to independent noise.
- All proofs can be found in the report version of this paper [17].

2 Preliminaries

Differential Privacy and Geo-indistinguishability. The privacy definitions used in this paper are based on a generalized variant of differential privacy that can be defined on an arbitrary set of secrets \mathcal{X} (not necessarily on databases), equipped with a metric $d_{\mathcal{X}}$ [18,19]. The distance $d_{\mathcal{X}}(x, x')$ expresses the *distinguishability level* between the secrets x and x' , modeling the privacy notion that we want to achieve. A small value denotes that the secrets should remain indistinguishable, while a large value means that we allow the adversary to distinguish them.

Let \mathcal{Z} be a set of *reported values* and let $\mathcal{P}(\mathcal{Z})$ denote the set of probability measures over \mathcal{Z} . The multiplicative distance $d_{\mathcal{P}}$ on $\mathcal{P}(\mathcal{Z})$ is defined as $d_{\mathcal{P}}(\mu_1, \mu_2) = \sup_{Z \subseteq \mathcal{Z}} \left| \ln \frac{\mu_1(Z)}{\mu_2(Z)} \right|$ with $\left| \ln \frac{\mu_1(Z)}{\mu_2(Z)} \right| = 0$ if both $\mu_1(Z), \mu_2(Z)$ are zero and ∞ if only one of them is zero. Intuitively $d_{\mathcal{P}}(\mu_1, \mu_2)$ is small if μ_1, μ_2 assign similar probabilities to each reported value.

A mechanism is a (probabilistic) function $K : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{Z})$, assigning to each secret x a probability distribution $K(x)$ over the reported values. The generalized variant of differential privacy, called $d_{\mathcal{X}}$ -privacy, is defined as follows:

Definition 1 ($d_{\mathcal{X}}$ -privacy). A mechanism $K : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{Z})$ satisfies $d_{\mathcal{X}}$ -privacy iff:

$$d_{\mathcal{P}}(K(x), K(x')) \leq d_{\mathcal{X}}(x, x') \quad \forall x, x' \in \mathcal{X}$$

or equivalently $K(x)(Z) \leq e^{d_{\mathcal{X}}(x, x')} K(x')(Z) \quad \forall x, x' \in \mathcal{X}, Z \subseteq \mathcal{Z}$.

Different choices of $d_{\mathcal{X}}$ give rise to different privacy notions; it is also common to scale our metric of interest by a privacy parameter ϵ (note that $\epsilon d_{\mathcal{X}}$ is itself a metric).

The most well-known case is when \mathcal{X} is a set of databases with the hamming metric $d_h(x, x')$, defined as the number of rows in which x, x' differ. In this case ϵd_h -privacy is the same as ϵ -differential privacy, requiring that for adjacent x, x' (i.e. differing on a single row) $d_{\mathcal{P}}(K(x), K(x')) \leq \epsilon$. Moreover, various other privacy notions of interest can be captured by different metrics [19].

Geo-indistinguishability In the case of location privacy, which is the main motivation of this paper, the secrets \mathcal{X} as well as the reported values \mathcal{Z} are sets of locations (i.e. subsets of \mathbb{R}^2), while K is an obfuscation mechanism. Using the Euclidean metric d_2 , we obtain ϵd_2 -privacy, a natural notion of location privacy called geo-indistinguishability in [14]. This privacy definition requires that the closer (geographically) two locations are, the more similar the probability of producing the same reported location z should be. As a consequence, the service provider is not allowed to infer the user's location with accuracy, but he can get approximate information required to provide the service.

Seeing it from a slightly different viewpoint, this notion offers privacy *within any radius* r from the user, with a level of distinguishability ϵr , proportional to r . Hence, within a small radius the user enjoys strong privacy, while his privacy decreases as r gets larger. This gives us the flexibility to adjust the definition to a particular application: typically we start with a radius r^* for which we want strong privacy, which can range from a few meters to several kilometers (of course a larger radius will lead to more noise). For this radius we pick a relatively small ϵ^* (for instance in the range from $\ln 2$ to $\ln 10$), and set $\epsilon = \epsilon^*/r^*$. Moreover, we are also flexible in selecting a different metric between locations, for instance the Manhattan or a map-based distance.

Two characterizations results are also given in [14], providing intuitive interpretations of geo-indistinguishability. Finally, it is shown that this notion can be achieved by adding noise from a 2-dimensional Laplace distribution.

Protecting location traces. Having established a privacy notion for single locations, it is natural to extend it to location *traces* (sometimes called *trajectories* in the literature). Although location privacy is our main interest, this can be done for traces having any secrets with a corresponding metric as elements. We denote by $\mathbf{x} = [x_1, \dots, x_n]$ a trace, by $\mathbf{x}[i]$ the i -th element of \mathbf{x} , by $[\]$ the empty trace and by $x :: \mathbf{x}$ the trace obtained by adding x to the head of \mathbf{x} . We also define $\text{tail}(x :: \mathbf{x}) = \mathbf{x}$. To obtain a privacy notion, we need to define an appropriate metric between traces. A natural choice is the maximum metric $d_{\infty}(\mathbf{x}, \mathbf{x}') = \max_i d_{\mathcal{X}}(\mathbf{x}[i], \mathbf{x}'[i])$. This captures the idea that two traces are as distinguishable as their most distinguishable points. In terms of protection within a radius, if \mathbf{x} is within a radius r from \mathbf{x}' it means that $\mathbf{x}[i]$ is within a radius r from $\mathbf{x}'[i]$. Hence, ϵd_{∞} -privacy ensures that all secrets are protected within a radius r with the same distinguishability level ϵr .

mechanism $\text{IM}(\mathbf{x})$

```

 $\mathbf{z} := []$ 
for  $i := 1$  to  $|\mathbf{x}|$ 
   $z := N(\epsilon_N)(\mathbf{x}[i])$ 
 $\mathbf{z} := z :: \mathbf{z}$ 
return  $\mathbf{z}$ 

```

Fig. 1: Independent Mechanism

In order to sanitize \mathbf{x} we can simply apply a *noise mechanism* independently to each secret x_i . We assume that a family of noise mechanisms $N(\epsilon_N) : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{Z})$ are available, parametrized by $\epsilon_N > 0$, where each mechanism $N(\epsilon_N)$ satisfies ϵ_N -privacy. The resulting mechanism, called the *independent mechanism* $\text{IM} : \mathcal{X}^n \rightarrow \mathcal{P}(\mathcal{Z}^n)$, is shown in Figure 1. As explained in the introduction, the main issue with this approach is that IM is $n\epsilon d_\infty$ -private, that is, the budget consumed increases linearly with n .

Utility. The goal of a privacy mechanism is not to hide completely the secret but to disclose enough information to be useful for some service while hiding the rest to protect the user’s privacy. Typically these two requirements go in opposite directions: a stronger privacy level requires more noise which results in a lower utility.

Utility is a notion very dependent on the application we target; to measure utility we start by defining a notion of *error*, that is a distance d_{err} between a trace \mathbf{x} and a sanitized trace \mathbf{z} . In the case of location-based systems we want to report locations as close as possible to the original ones, so a natural choice is to define the error as the average geographical distance between the locations in the trace:

$$d_{\text{err}}(\mathbf{x}, \mathbf{z}) = \frac{1}{|\mathbf{x}|} \sum_i d_2(\mathbf{x}[i], \mathbf{z}[i]) \quad (1)$$

We can then measure the utility of a trace obfuscation mechanism $K : \mathcal{X}^n \rightarrow \mathcal{P}(\mathcal{Z}^n)$ by the *average-case* error, defined as the expected value of d_{err} :

$$E[d_{\text{err}}] = \sum_{\mathbf{x}} \pi(\mathbf{x}) \sum_{\mathbf{z}} K(\mathbf{x})(\mathbf{z}) d_{\text{err}}(\mathbf{x}, \mathbf{z})$$

where $\pi \in \mathcal{P}(\mathcal{X}^n)$ is a prior distribution on traces.

On the other hand, the worst-case error is usually unbounded, since typical noise mechanisms (for instance the Laplace one) can return values at arbitrary distance from the original one. Hence, we are usually interested in the p -th percentile of the error, commonly expressed in the form of $\alpha(\delta)$ -*accuracy* [20]. A mechanism K is $\alpha(\delta)$ -accurate iff for all δ : $Pr[d_{\text{err}}(\mathbf{x}, \mathbf{z}) \leq \alpha(\delta)] \geq \delta$. In the rest of the paper we will refer to $\alpha(0.9)$ (or simply α) as the “worst-case” error.

Note that in general, both $E[d_{\text{err}}]$ and $\alpha(\delta)$ depend on the prior distribution π on traces. However, due to the mechanism’s symmetry, the utility of the Laplace mechanism is independent from the prior, and as a result, the utility of the independent mechanism (using the Laplace as the underlying noise mechanism) is also prior-independent. On the other hand, the utility of the predictive mechanism, described in the next section, will be highly dependent on the prior. As explained in the introduction, the mechanism takes advantage of the correlation between the points in the trace (a property of the prior), the higher the correlation the better utility it will provide.

3 A predictive $d_{\mathcal{X}}$ -private mechanism

We are now ready to introduce our prediction-based mechanism. Although our main motivation is location privacy, the mechanism can work for traces of any secrets \mathcal{X} ,

equipped with a metric d_x . The fundamental intuition of our work is that the presence of correlation on the secret can be exploited to the advantage of the mechanism. A simple way of doing this is to try to predict new secrets from past information; if the secret can be predicted with enough accuracy it is called *easy*; in this case the prediction can be reported without adding new noise. On the other hand, *hard* secrets, that is those that cannot be predicted, are sanitized with new noise. Note the difference with the independent mechanism where each secret is treated independently from the others.

Let $\mathcal{B} = \{0, 1\}$. A boolean $b \in \mathcal{B}$ denotes whether a point is easy (0) or hard (1). A sequence $\mathbf{r} = [z_1, b_1, \dots, z_n, b_n]$ of reported values and booleans is called a *run*; the set of all runs is denoted by $\mathcal{R} = (\mathcal{Z} \times \mathcal{B})^*$. A run will be the output of our predictive mechanism; note that the booleans b_i are considered public and will be reported by the mechanism.

Main components The predictive mechanism has three main components: first, the *prediction* is a deterministic function $\Omega : \mathcal{R} \rightarrow \mathcal{Z}$, taking as input the run reported up to this moment and trying to predict the next *reported value*. The output of the prediction function is denoted by $\tilde{z} = \Omega(\mathbf{r})$. Note that, although it is natural to think of Ω as trying to predict the secret, in fact what we are trying to predict is the reported value. In the case of location privacy, for instance, we want to predict a reported location at acceptable distance from the actual one. Thus, the possibility of a successful prediction should not be viewed as a privacy violation.

Second, a *test* is a family of mechanisms $\Theta(\epsilon_\theta, l, \tilde{z}) : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{B})$, parametrized by $\epsilon_\theta, l, \tilde{z}$. The test takes as input the secret x and reports whether the prediction \tilde{z} is acceptable or not for this secret. If the test is successful then the prediction will be used instead of generating new noise. The purpose of the test is to guarantee a certain level of utility: predictions that are farther than the threshold l should be rejected. Since the test is accessing the secret, it should be private itself, where ϵ_θ is the budget that is allowed to be spent for testing.

The test mechanism that will be used throughout the paper is the one below, which is based on adding Laplace noise to the threshold l :

$$\Theta(\epsilon_\theta, l, \tilde{z})(x) = \begin{cases} 0 & \text{if } d_x(x, \tilde{z}) \leq l + \text{Lap}(\epsilon_\theta) \\ 1 & \text{ow.} \end{cases} \quad (2)$$

The test is defined for all $\epsilon_\theta > 0, l \in [0, +\infty), \tilde{z} \in \mathcal{Z}$, and can be used for any metric d_x , as long as the domain of reported values is the same as the one of the secrets (which is the case for location obfuscation) so that $d_x(x, \tilde{z})$ is well defined.

Finally, a *noise mechanism* is a family of mechanisms $N(\epsilon_N) : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{Z})$, parametrized by the available budget ϵ_N . The noise mechanism is used for hard secrets that cannot be predicted.

Budget management The parameters of the mechanism's components need to be configured at each step. This can be done in a dynamic way using the concept of a *budget manager*. A budget manager β is a function that takes as input the run produced so far and returns the budget and the threshold to be used for the test at this step as well as the budget for the noise mechanism: $\beta(\mathbf{r}) = (\epsilon_\theta, \epsilon_N, l)$. We will also use β_θ and β_N as shorthands to get just the first or the second element of the result.

| | |
|--|---|
| <pre> mechanism PM(\mathbf{x}) $\mathbf{r} := []$ for $i := 1$ to \mathbf{x} $(z, b) := \text{Step}(\mathbf{r})(\mathbf{x}[i])$ $\mathbf{r} := (z, b) :: \mathbf{r}$ return \mathbf{r} </pre> | <pre> mechanism Step(\mathbf{r})(x) $(\epsilon_\theta, \epsilon_N, l) := \beta(\mathbf{r})$ $\tilde{z} := \Omega(\mathbf{r})$ $b := \Theta(\epsilon_\theta, l, \tilde{z})(x)$ if $b == 0$ then $z := \tilde{z}$ else $z := N(\epsilon_N)(x)$ return (z, b) </pre> |
|--|---|

(a) Predictive Mechanism

(b) Single step of the Predictive Mechanism

Of course the amount of budget used for the test should always be less than the amount devoted to the noise, otherwise it would be more convenient to just use the independent noise mechanism. Still, there is great flexibility in configuring the various parameters and several strategies can be implemented in terms of a budget manager.

The mechanism We are now ready to fully describe our mechanism. A single step of the predictive mechanism, displayed in Figure 2b, is a family of mechanisms $\text{Step}(\mathbf{r}) : \mathcal{X} \rightarrow \mathcal{P}(\mathcal{Z} \times \mathcal{B})$, parametrized by the run \mathbf{r} reported up to this point. The mechanism takes a secret x and returns a reported value z , as well as a boolean b denoting whether the secret was easy or hard. First, the mechanism obtains the various configuration parameters from the budget manager as well as a prediction \tilde{z} . Then the prediction is tested using the test mechanism. If the test is successful the prediction is returned, otherwise a new reported value is generated using the noise mechanism.

Finally, the predictive mechanism, displayed in Figure 2a, is a mechanism $\text{PM} : \mathcal{X}^n \rightarrow \mathcal{P}(\mathcal{R})$. It takes as input a trace \mathbf{x} , and applies $\text{Step}(\mathbf{r})$ to each secret, while extending at each step the run \mathbf{r} with the new reported values (z, b) .

Note that an important advantage of the mechanism is that it is *online*, that is the sanitization of each secret does not depend on future secrets. This means that the user can query at any time during the life of the system, as opposed to *offline* mechanisms where all the queries need to be asked before the sanitization. Furthermore the mechanism is *dynamic*, in the sense that the secret can change over time (e.g. the position of the user) contrary to static mechanism where the secret is fixed (e.g. a static database).

It should be also noted that, when the user runs out of budget, he should in principle stop using the system. This is typical in the area of differential privacy where a database should not be queried after the budget is exhausted. In practice, of course, this is not realistic, and new queries can be allowed by resetting the budget, essentially assuming either that there is no correlation between the old and new data, or that the correlation is weak and cannot be exploited by the adversary. In the case of location privacy we could, for instance, reset the budget at the end of each day. We are currently investigating proper assumptions under which the budget can be reset while satisfying a formal privacy guarantee. The question of resetting the budget is open in the field of differential privacy and is orthogonal to our goal of making an efficient use of it.

The main innovation of this mechanism is the use of the prediction function, which allows to decouple the privacy mechanism from the correlation analysis, creating a family of modular mechanisms where by *plugging* in different predictions (or updating the existing) we are able to work in new domains. Moreover proving desirable security

properties about the mechanism independently of the complex engineering aspects of the prediction is both easier and more reliable, as shown in the next sections.

3.1 Privacy

We now proceed to show that the predictive mechanism described in the previous section is d_x -private. The privacy of the predictive mechanism depends on that of its components. In the following, we assume that each member of the families of test and noise mechanisms is d_x -private for the corresponding privacy parameter:

$$\forall \epsilon_\theta, l, \tilde{z}. \Theta(\epsilon_\theta, l, \tilde{z}) \text{ is } \epsilon_\theta d_x\text{-private} \quad (3)$$

$$\forall \epsilon_N. N(\epsilon_N) \text{ is } \epsilon_N d_x\text{-private} \quad (4)$$

In the case of the test $\Theta(\epsilon_\theta, l, \tilde{z})$ defined in (2), we can show that it is indeed d_x -private, independently of the metric or threshold used.

Fact 1 (Privacy of Test function). *The family of test mechanisms $\Theta(\epsilon_\theta, l, \tilde{z})$ defined by (2) satisfies assumption 3.*

The global budget for a certain run \mathbf{r} using a budget manager β is defined as:

$$\epsilon_\beta(\mathbf{r}) = \begin{cases} 0 & \text{if } |\mathbf{r}| = 0 \\ \beta_\theta(\mathbf{r}) + b(\mathbf{r}) \times \beta_N(\mathbf{r}) + \epsilon_\beta(\mathbf{tail}(\mathbf{r})) & \text{o.w.} \end{cases} \quad (5)$$

As already discussed, a hard step is more expensive than an easy step because of the cost of the noise mechanism.

Building on the privacy properties of its components, we first show that the predictive mechanism satisfies a property similar to d_x -privacy, with a parameter ϵ that depends on the run.

Lemma 1. *Under the assumptions (3),(4), for the test and noise mechanisms, the predictive mechanism PM, using the budget manager β , satisfies*

$$\text{PM}(\mathbf{x})(\mathbf{r}) \leq e^{\epsilon_\beta(\mathbf{r}) d_\infty(\mathbf{x}, \mathbf{x}')} \text{PM}(\mathbf{x}')(\mathbf{r}) \quad \forall \mathbf{r}, \mathbf{x}, \mathbf{x}' \quad (6)$$

This results shows that there is a difference between the budget spent on a “good” run, where the input has a considerable correlation, the prediction performs well and the majority of steps are easy, and a run with uncorrelated secrets, where any prediction is useless and all the steps are hard. In the latter case it is clear that our mechanism wastes part of its budget on tests that always fail, performing worse than an independent mechanism.

Finally, the overall privacy of the mechanism will depend on the budget spent on the worst possible run.

Theorem 1 (d_x -privacy). *Under the assumptions (3),(4), for the test and noise mechanisms, the predictive mechanism PM, using the budget manager β , satisfies ϵd_∞ -privacy, with $\epsilon = \sup_{\mathbf{r}} \epsilon_\beta(\mathbf{r})$.*

Based on the above result, we will use ϵ -bounded budget managers, imposing an overall budget limit ϵ independently from the run. Such a budget manager provides a fixed privacy guarantee by sacrificing utility: in the case of a bad run it either needs to lower the budget spend per secret, leading to more noise, or to stop early, handling a smaller number of queries. In practice, however, using a prediction function tailored to a specific type of correlation we can achieve good efficiency.

3.2 Utility

We now turn our attention to the utility provided by the predictive mechanism. The property we want to prove is $\alpha(\delta)$ -accuracy, introduced in Section 2. Similarly to the case of privacy, the accuracy of the predictive mechanism depends on that of its components, that is, on the accuracy of the noise mechanism, as well as the one of the Laplace mechanism employed by the test $\Theta(\epsilon_\theta, l, \tilde{z})$ (2). We can now state a result about the utility of a *single step* of the predictive mechanism.

Proposition 1 (accuracy). *Let \mathbf{r} be a run, β a budget manager, let $(\epsilon_\theta, \epsilon_N, l) = \beta(\mathbf{r})$ and let $\alpha_N(\delta)$, $\alpha_\theta(\delta)$ be the accuracy of $N(\epsilon_N)$, $Lap(\epsilon_\theta)$ respectively. Then the accuracy of Step(\mathbf{r}) is $\alpha(\delta) = \max(\alpha_N(\delta), l + \alpha_\theta(\delta))$*

This result provides a bound for the accuracy of the predictive mechanism at each step. The bound depends on the triplet used $(\epsilon_\theta, \epsilon_N, l)$ to configure the test and noise mechanisms which may vary at each step depending on the budget manager used, thus the bound is step-wise and may change during the use of the system.

It should be noted that the bound is independent from the prediction function used, and assumes that the prediction gives the worst possible accuracy allowed by the test. Hence, under a prediction that always fails the bound is tight; however, under an accurate prediction function, the mechanism can achieve much better utility, as shown in the evaluation of Section 5.

3.3 Skipping the test

The amount of budget devoted to the test is still linear in the number of steps and can amount to a considerable fraction; for this reason, given some particular conditions, we may want to skip it altogether using directly the prediction or the noise mechanism. The test mechanism we use (2) is defined for all $\epsilon_\theta > 0, l \in [0, +\infty)$. We can extend it to the case $\epsilon_\theta = 0, l \in \{-\infty, +\infty\}$ with the convention that $\Theta(0, +\infty, \tilde{z})$ always returns 1 and $\Theta(0, -\infty, \tilde{z})$ always returns 0. The new test mechanisms are independent of the input x so they can be trivially shown to be private, with no budget being consumed.

Fact 2 (Privacy of Test function). *The test functions $\Theta(0, +\infty, \tilde{z})$ and $\Theta(0, -\infty, \tilde{z})$ satisfy assumption 3.*

Now if β returns $(0, \epsilon_N, -\infty)$ we always fallback to the noise mechanism $N(\epsilon_N)$; this is especially useful when we know the prediction is not in conditions to perform well and testing would be a waste of budget. For instance, consider a prediction function that needs at least a certain number n of previous observables to be able to predict with enough accuracy; in this case we can save some budget if we directly use the noise mechanism for those n steps without testing. Note that the bound on utility is preserved in this case, as we can rely on the $\alpha_N(\delta)$ -accuracy of $N(\epsilon_N)$.

On the other hand, the budget manager can return $(0, 0, +\infty)$ which causes the prediction to be reported without spending any budget. This decision could be based on any public information that gives high confidence to the prediction. A good use of this case can be found in Section 5 where *timing information* is used to skip the test.

Note that the prediction is computed from public knowledge, so releasing it has no privacy cost. However in this case we loose any guarantee on the utility of the reported

answer, at least in the general case; based on the criteria for skipping the test (as in the case of the user walking in the city), we could make assumptions about the quality of the prediction which would allow to restore the bound.

4 Predictive mechanism for location privacy

The applicability of d_x -privacy to location-based systems, called geo-indistinguishability in this context, was already discussed in Section 2. Having studied the general properties of our predictive mechanism, we are ready to apply it for location privacy.

As already described in the preliminaries the sets of secret and observables are sets of geographical coordinates, the metric used is the euclidean distance and we will use $\Theta(\epsilon_\theta, l, \tilde{z})$ (2) as test function. We start with the description of a simple prediction function, followed by the design of two budget managers and finally some heuristics used to skip the test.

Prediction Function. For the prediction function we use a simple strategy, the parrot prediction, that just returns the value of the last observable, which ultimately will be the last hard observable.

$$\text{parrot}((z, b) :: \mathbf{r}) = z \quad (7)$$

Despite its simplicity, this prediction gives excellent results in the case when the secrets are close to each other with respect to the utility required - e.g. suppose the user queries for restaurants and he is willing to accept reported points as far as 1 km from the secret point, if the next positions are tens of meters apart, then the same reported point will be a good prediction for several positions. Similarly, the prediction is quite effective when the user stays still for several queries, which is a typical case of a smartphone user accessing an LBS.

More concretely, we define the *step* of a trace as the average distance between its adjacent points $\sigma(\mathbf{x}) = \text{avg}_{0 \leq i < |\mathbf{x}|} d(x_i, x_{i+1})$ and we compare it with the $\alpha_N(0.9)$ -accuracy of the noise mechanism. The intuition is that the parrot prediction works well on a trace \mathbf{x} if $\sigma(\mathbf{x})$ is smaller than $\alpha_N(0.9)$ or in the presence of clusters because once we release a hard point we can use it as a good enough prediction for several other secret points close to it.

Furthermore the parrot prediction can be trivially implemented on any system and it has the desirable property of being independent from the user; taking into account past traces of the user, for instance, would give a more effective prediction, but it would be restricted to that particular user.

Budget Managers When configuring a mechanism we need to take into account 3 global parameters: the global privacy, the utility and the number of interactions, written (ϵ, α, n) for brevity. All three are interdependent and fixing one we obtain a relation between the other two. In our case we choose to be independent of the length of the traces; to do so we introduce the *privacy consumption rate* (or just rate) which is the amount of budget spent at each step on average: $\rho(\mathbf{r}) = \frac{\epsilon(\mathbf{r})}{|\mathbf{r}|}$. This measure represent the privacy usage of the mechanism or how *fast* we run out of budget and given this value we can easily retrieve how many points we can cover given a certain initial budget. As

already done for d_{err} , we also introduce the average-case rate for the mechanism as the expected value of ρ , given a prior distribution $\pi \in \mathcal{P}(\mathcal{X}^n)$ on traces:

$$E[\rho] = \sum_{\mathbf{x}} \pi(\mathbf{x}) \sum_{\mathbf{r}} \text{PM}(\mathbf{x})(\mathbf{r}) \rho(\mathbf{r})$$

Given that our main concern is privacy we restrict ourselves to ϵ -*bounded* budget managers, that guarantee that the total budget consumed by the mechanism will never exceed ϵ , and divide them in two categories:

Fixed Utility: In the independent mechanism if we want to guarantee a certain level of utility, we know that we need to use a certain amount of budget at each step, a fixed rate, thus being able to cover a certain number n of steps. However in our case, if the test is successful, we may save the cost of the noise and meet the fixed utility with a smaller rate per point; smaller rates translates in additional interactions possible after n . We fix the utility and minimize the rate.

Fixed Rate: Alternatively, if in the independent mechanism we want to cover just n steps, thus fixing the rate, we would obtain a certain fixed utility. On the contrary the predictive mechanism, in the steps where the test succeeds, spends less than the chosen rate, allowing the next steps to spend *more* than the rate. This alternance creates a positive behavior where hard points can use the saved budget to increase their accuracy that in turn makes predicting more accurate and likely to succeed, leading to more saving. Of course the average cost for all steps meets the expected rate. In this case we fix the rate and maximize the utility.

Configuration of the mechanism We now give an overview of the constraints that are present on the parameters of the predictive mechanism and a guideline to configure them to obtain the desired levels of privacy and utility. The only settings that the user needs to provide are ϵ and either α or ρ . The budget manager will define at each step the amount of budget devoted to the test ϵ_θ , the noise mechanism ϵ_N and the test threshold l , starting from the global settings.

Budget usage First we define the *prediction rate* PR as the percentage points predicted successfully; this property will be used to configure and to verify how effective is the predictive mechanism. We can then introduce a first equation which relates ϵ_θ and ϵ_N to the budget consumption rate: $\rho = \epsilon_\theta + (1 - PR)\epsilon_N$. This formula is derived from the budget usage of the mechanism (Lemma 1), with the two following approximations. First, ϵ_θ and ϵ_N in future steps are assumed constant. In practice they will be variable because this computation is re-done at each step with the actual remaining budget. Second, we assume the hard steps are evenly distributed along the run. This allows us to use PR , which is a global property of the trace, in a local computation.

Note that ρ is constant in the fixed rate case and is computed over the current run for the fixed utility case. We already knew that the budget available at each step had to be split between Θ and N , this result confirms the intuition that the more we manage to predict (higher PR) the less we'll need to spend for the noise generation (on average over the run).

Utility From the utility result given by Proposition 1 we obtain an equation that relates all the parameters of the mechanism, ϵ_θ , ϵ_N and l . Given that the global utility will be

budget manager $\beta(\mathbf{r})$
if $\epsilon(\mathbf{r}) \geq \epsilon$ **then STOP**
else
 $\epsilon_\theta := \eta \frac{c_\theta}{\alpha} (1 + \frac{1}{\gamma})$
 $\epsilon_N := \frac{c_N}{\alpha}$
 $l := \frac{c_\theta}{\gamma \epsilon_\theta}$
return $(\epsilon_\theta, \epsilon_N, l)$

budget manager $\beta(\mathbf{r})$
if $\epsilon(\mathbf{r}) \geq \epsilon$ **then STOP**
else
 $\epsilon_N := \frac{\rho}{(1-PR) + \frac{c_\theta}{c_N} \eta (1 + \frac{1}{\gamma})}$
 $\epsilon_\theta := \epsilon_N \eta \frac{c_\theta}{c_N} (1 + \frac{1}{\gamma})$
 $l := \frac{c_\theta}{\gamma \epsilon_\theta}$
return $(\epsilon_\theta, \epsilon_N, l)$

(a) Fixed Utility configured with ϵ and α (b) Fixed Rate configured with ϵ , ρ and PR

the worst of the two, we decide to give both the noise and predictive components the same utility: $\alpha_N = l + \alpha_\theta$. Moreover, as discussed in the utility section, this result is a bound valid for every possible prediction function, even one that always fails, for this reason the bound may be too pessimistic for the practical cases where the prediction does work. In order to reduce the influence of the accuracy of the predictive component we introduce a parameter $0 \leq \eta \leq 1$ that can be set to 1 to retrieve the strict case or can safely go as low as 0.5 as shown in our experiments. Finally we obtain the following relation between the parameters: $\alpha = \alpha_N = \eta(l + \alpha_\theta)$.

Noise-threshold ratio Now we have two equations for three parameters and to completely configure the mechanism we introduce an additional parameter $0 \leq \gamma \leq 1$ that is used to tune, in the predictive component, the ratio between the threshold l and the Laplacian noise added to it so that $\gamma = \frac{\alpha_\theta}{l}$. The intuition is that γ should not be bigger than 1, otherwise the noise could be more important than the threshold and we might as well use a random test. For our experiments we found good values of γ around 0.8.

Note that both η and γ are values that should be determined using a representative sample of the expected input, in a sort of tuning phase, and then fixed in the mechanism. The same goes for the expected prediction rate that is used to configure the budget managers, at least in the beginning this value is necessary to allocate some resource for Θ , after some iterations it is computed from the actual run.

Relation between accuracy and epsilon The final simplification that we apply is when we compute the accuracy of the noisy components, for both the linear Laplacian and the polar Laplacian we can compute their maximum value up to a certain probability δ using their inverse cumulative probability distributions, that we denote icll and icpl respectively. Fixing δ to 0.9, both these functions can be expressed as the ratio of a constant and the epsilon used to scale the noise $\alpha_N(\delta) = \text{icpl}(\epsilon_N, \delta) = \frac{c_N(\delta)}{\epsilon_N}$ and $\alpha_\theta(\delta) = \text{icll}(\epsilon_\theta, \delta) = \frac{c_\theta(\delta)}{\epsilon_\theta}$.

Now that we have the equations that relate the various parameters, from the settings given by the user we can realize the two budget managers, shown in Figure 3a and 3b.

Furthermore we can compare the expected rate or accuracy of our mechanism with those of an independent mechanism and find the prediction rate that we need to meet to provide an improvement. We obtain in both cases a lower bound on the prediction rate: $PR \geq \eta \frac{c_\theta}{c_N} (1 + \frac{1}{\gamma})$. This gives an idea of the feasibility of a configuration before

actually running it, for example using the parameters of our experiments we find that it is necessary to predict at least 46% of points to make up for the cost of the test.

5 Case study

To evaluate our mechanism, we follow our motivating example stated in the introduction of a user performing several activities while moving around the city throughout a day, possibly using different means of transport. During these activities, the user performs queries to an LBS using his mobile device, while wishing to keep his location private.

We assume that the user queries the LBS only when being still or moving at a slow speed (less than 15 km/h); this reflects the semantic of a geo localized query: there is usually little value in asking information relative to one's current position if the position is changing quickly. We perform a comparison between the independent mechanism IM and our predictive mechanism PM, both using polar Laplace noise as the underlying noise mechanism. The mechanisms are evaluated on two data sets of real GPS trajectories, using both a fixed-utility and fixed-rate budget managers and a skip strategy.

Data sets The first data set we tested our mechanism against, is the well known GeoLife [15] which collects 18.670 GPS trajectories from 182 users in Beijing during a period of over five years. In this set the users take a variety of means of transport, from walking and biking to car, train, metro, taxi and even airplane. Regarding the trajectories length we can roughly divide them on three equal groups, less than 5 km, between 5 and 20 km and more than 20 km. As for duration 58% are less than 1 hour, 26% between 1 and 6 hours and 16% more than 6 hours.

The second data set is Tdrive [16], a collection of about 9000 taxi trajectories, always in the city of Beijing. As opposed to the variety of Geolife in this set we have only cars movements and the trajectories tends to be longer in both time and distance. The interest of using this set, which does not exactly correspond to our target use case of a user walking in a city, is to test the flexibility of the mechanism.

In order to use this sets some preprocessing is needed in order to model our use case. GPS trajectories present the problem of having *all the movements* of the user, instead of just the points where the user actually *queried* the LBS, which is a small subset of the trajectory. For this reason we perform a probabilistic "sampling" of the trajectories that, based on the speed and type of user, produces a trace of query points. First, we select the part of the trace where the speed is less than 15 km/h, and in these segments we sample points depending on the type of user, as explained below.

Users are classified based on the frequency of their use of the LBS, from occasional to frequent users. This is achieved by defining two intervals in time, one brief and the other long (a *jump*), that could occur between two subsequent queries. Then each class of users is generated by sampling with a different *probability of jumping* p , that is the probability that the next query will be after a long interval in time. Each value of p gives rise to a different prior distribution π on the produced traces, hence affecting the performance of our mechanism.

The interval that we used in our experiments are 1 and 60 minutes, both with addition of a small Gaussian noise; frequent users will query almost every minute while occasional users around every hour. In our experiments we generated 11 such priors,

with probability of jumping ranging from 0 to 1 at steps of 0.1, where each trace was sampled 10 times.

Configuration In order to configure the geo-indistinguishable application, first the user defines a radius r^* where she wishes to be protected, that we assume is 100 meters, and then the application sets ϵ^* , the global level of privacy, to be $\ln 10$. This means that taken two points on the radius of 100 meters their probability of being the observables of the same secret differ at most by 10, and even less the more we take them closer to the secret. We think this is a reasonable level of privacy in a dense urban environment. For what concerns the two budget managers, the fixed-rate was tested with a 3.3% rate, which corresponds to about 30 queries, which in a day seems a reasonable number even for an avid user. For the fixed-utility we set an accuracy limit 3 km, again reasonable if we consider a walking distance and that these are worst cases.

Skip-the-test strategy While the aim of the mechanism is to hide the user’s position, the timestamp of a point is observable, hence we can use the elapsed time from the last reported point to estimate the distance that the user may have traveled. If this distance is less than the accuracy required, we can report the predicted value without testing it, we know that the user can’t be too far from his last reported position. The risk of this approach lies in the speed that we use to link elapsed time and traveled distance, if the user is faster than expected (maybe he took a metro) we would report an inaccurate point. To be on the safe side it should be set to the maximum speed we expect our users to travel at, however with lower values we’ll be able to skip more, it is a matter of how much we care about accuracy or how much we know about our users. In our experiments we assumed this speed to be 0.5 km/h.

We would expect this approach to be more convenient in a context where accuracy is not the primary goal; indeed skipping the test will provide the greatest advantage for the fixed-utility case, where we just don’t want to exceed a worst case limit.

Additionally we use another skip-the-test strategy to use directly with the noise mechanism when we are in the first step and thus there is no previous hard point for the parrot prediction to report. This is a trivial example of skip strategy, yet it can lead to some budget savings.

Results It should be noted that both the preprocessing and the sanitization were performed with same configuration on both data sets. The results of running the mechanism on the samples traces from the Geolife data set, are reported in figures 4, 5, the graphs of Tdrive are omitted for reason of space as they show a very similar behavior to Geolife (they can be found in [17]). In the horizontal axis we have the probability p that was used during the sampling, to determine how often the user performs a *jump* in time: the smaller the value the more frequent the queries. For each budget manager we plot: In the first graph, some general statistics about the mechanism, such as the prediction rate achieved, the amount of budget devoted to Θ and the amount of skipped points; In the second column the average ($E[d_{\text{err}}]$) and 90-th percentile ($\alpha(0.9)$) of the error; In the third the average budget consumption rate $E[\rho]$. Furthermore we run the experiments with and without the skip the test strategy, for the sake of comparison.

The graphs present a smooth behaviour, despite the use of real data, because of the sampling on each trace and the averaging over all traces of all users. As general remarks,

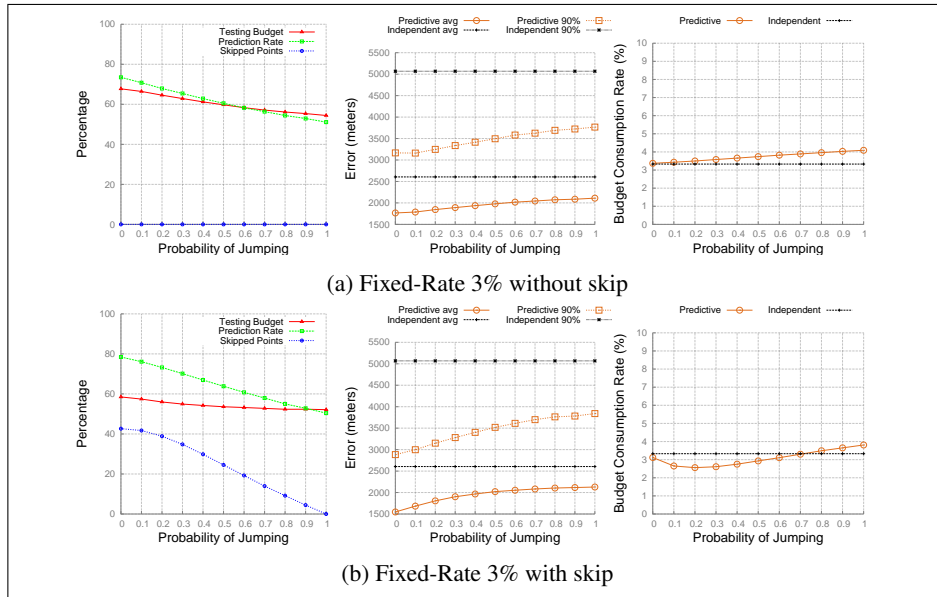


Fig. 4: General statistics, Average Error and Rate for Fixed-Rate budget manager.

we can see that the prediction rate degrades as the users become more occasional, thus less predictable, and the same goes for the number of skipped points. Notice that the testing budget adapts with the prediction rate which is a sign that the budget managers reconfigure dynamically.

Fixed-rate (Fig. 4): fixing the rate to 3.3% to cover 30 points, we can devote the budget saved to improve the accuracy. In the right most graph we see that indeed the rate is very stable even in the unpredictable cases, and very close to the rate of the independent mechanism. The graph in the center shows great improvements in the average error, 500 m in the worst case and 700 m in the best, and even more remarkable is the improvement for the maximum error, 1.3km up to 1.9km. With the skip strategy we see a small improvement for $p \leq 0.5$, again both in average and maximum error, which correspond to a decrease in the testing budget in the left most graph: the budget saved skipping the test is invested in more accurate noise.

Fixed-utility (Fig. 5): fixing the maximum utility (or in-accuracy) to 3 km, our mechanism manages to save up to 1.5% of budget rate. If we want to compare the number of points covered, the independent mechanism can do around 17 points while the predictive 24. As expected the average and max errors are below the independent mechanism corresponding values which confirms that the budget manager is working correctly keeping the utility above a certain level. Despite this they don't show a stable behavior like the rate in the fixed-rate case, this is due to the fact that while we can finely control the amount of budget that we spend, the error is less controllable, especially the one produced by the predictive component. With the skip strategy in this case we obtain a very noticeable improvement in this case, with rates as low as 2% in the best case which translates to 50 points covered. As already pointed out, in this case the skip strategy is more fruitful because we care less about accuracy.

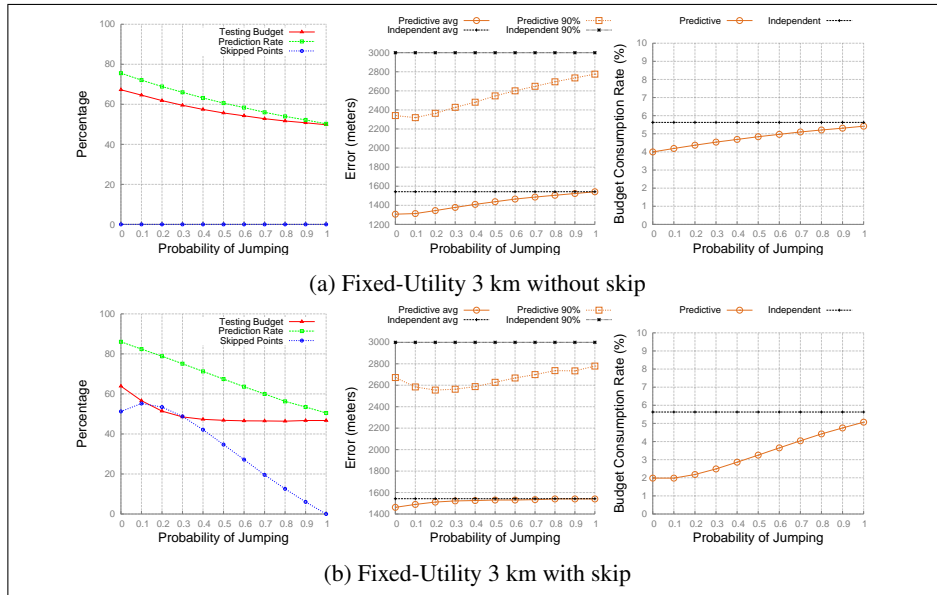


Fig. 5: General statistics, Average Error and Rate for Fixed-Utility budget manager.

Tdrive This data set reports remarkably similar performance to Geolife when the probability of jumping p is less than 0.7. In this cases the predictive mechanism is consistently a better choice than the independent mechanism on both budget managers. On the contrary for higher values of p the independent mechanism performs better, it is interesting to notice that the prediction rate at $p = 0.7$ starts to be lower than 46%, as expected from Section 4. This difference between the best and worst case is more accentuated in *Tdrive* precisely because the prediction function was not designed for this scenario. The more sporadic users are even less predictable as they are moving at higher speeds and roaming larger areas. Also the skip strategy, again designed for walking users, shows some spikes in the average error, due to wrongly skipped points where probably the taxi speeded up suddenly.

Figure 6 displays one of Geolife trajectories sanitized with fixed utility. The original trace, in red, starts south with low speed, moves north on a high speed road and then turns around Tsinghua University for some time, again at low speed, for a total of 18 km traveled in 10 hours. The sampled trace was obtained with a probability 0.5 of jumping and is plotted in light blue: as expected, 9 of the points are north, one south and the middle part was skipped. Finally in yellow we have the reported trace with 3 locations, which were used once for the point at the bottom, 7 times for the one in the middle and twice for point in the top.

6 Conclusions and Future Work

Future Work. As the experiments show the more efficient use of budget allows us to cover a day of usage, which was the goal we were aiming for in order to attack realistic applications. The intuition is that even if there is correlation between the traces of the same user on several days (for example we go to work and home every day) still it

is not enough to accurately locate the user at a precise moment in time (we might go to work later, or follow a different road). It is not clear though if one day is enough time to break the correlation and possibly reset the budget, we leave to future work to investigate in which cases it is indeed possible to reset the system and when on the contrary the epsilon keeps increasing.

One other possibility to prolong even further the use of the system is to improve the prediction. An extension we plan to develop consist in using the mobility traces of a user, or of a group of users, to designate locations where the next position is likely to be. In [21] the authors already developed inference attacks on the reported locations of users to discover points of interests and future locations, among other things; the idea is to use these attacks as a prediction.

We are also developing a *linearizing* prediction, that using past points tries to establish the next location using a linear regression method.

Alternatively we are considering the use of public geographic information to improve the prediction, which could simply translate to using already developed map-matching algorithms: typically in navigation systems an approximate location needs to be matched to an existing map, for example to place the user on a road. Map matching would make trivial predicting the direction of the user moving on a road for example, while in crossroads could be dealt with with the help of the mobility traces already discussed before: if on the left the is just countryside and on the right a mall, the user is more likely to turn right. Ultimately if more than one prediction function prove effective, we are interested in the possibility to merge them, for instance using multiplicative weights or related technique (e.g. Kalman filters).

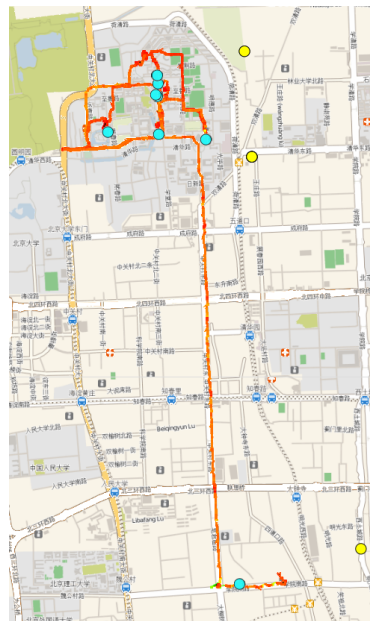


Fig. 6: Original trace (red), sampled trace (light blue) and reported trace (yellow).

Related work. On the predictive mechanism side, our mechanism was mainly inspired by the median mechanism [20], a work on differential privacy for databases based on the idea of exploiting the correlation on the queries to improve the budget usage. The mechanism uses a concept similar to our *prediction* to determine the answer to the next query using only past answers. An analogous work is the multiplicative weights mechanism [22], again in the context of statistical databases. The mechanism keeps a parallel version of the database which is used to predict the next answer and in case of failure it is updated with a multiplicative weights technique.

A key difference from our context is that in the above works, several queries are performed against the *same database*. In our setting, however, the secret (the position of the user) is always changing, which requires to exploit correlations in the data. This

scenario is explored also in [23] where the authors consider the case of an evolving secret and develop a differentially private counter.

Concerning location privacy, there are excellent works and surveys [24,25,26] that present the threats, methods, and guarantees. Like already discussed in the introduction the main trends in the field are those based on the expectation of distance error [9,8,27,28] and on the notion of k -anonymity [3,4,5,6,7], both dependents on the adversary's side information, as are some other works [29] and [30].

Notions that abstract from the attacker's knowledge based on differential privacy can be found in [11] and [12] although only for *aggregate* information.

The notion we based our work on, geo-indistinguishability [14], other than abstracting from the attacker's prior knowledge, and therefore being suitable for scenarios where the prior is unknown, or the same mechanism must be used for multiple users, can be used for single users. In addition, being the definition an instantiation of the more general notion of d_x -privacy [19] we were able to generalize our mechanism as well, being the prediction the only domain specific component.

Conclusions. We designed a general framework for private predictive d_x -private mechanisms able to manage the privacy budget more efficiently than the standard approach, in the cases where there is a considerable correlation on the data. The mechanism is modular and clearly separates the privacy protecting components from the predictive components, allowing ease of analysis and flexibility. We provide general configuration guidelines usable for any notion of d_x -privacy and a detailed instantiation for geo-indistinguishability. We tested the geo private mechanism obtained with two large sets of GPS trajectories and confirmed the goals set in the design phase. Experimental results show that the correlation naturally present in a user data is enough for our mechanism to outperform the independent mechanism in the majority of prior tested.

References

1. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: Proc. of MobiSys, USENIX (2003)
2. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: Query processing for location services without compromising privacy. In: Proc. of VLDB, ACM (2006) 763–774
3. Kido, H., Yanagisawa, Y., Satoh, T.: Protection of location privacy using dummies for location-based services. In: Proc. of ICDE Workshops. (2005) 1248
4. Shankar, P., Ganapathy, V., Iftode, L.: Privately querying location-based services with Sybil-Query. In: Proc. of UbiComp, ACM (2009) 31–40
5. Bamba, B., Liu, L., Pesti, P., Wang, T.: Supporting anonymous location queries in mobile environments with privacygrid. In: Proc. of WWW, ACM (2008) 237–246
6. Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: Proc. of PERVASIVE. Volume 3468 of LNCS., Springer (2005) 152–170
7. Xue, M., Kalnis, P., Pung, H.: Location diversity: Enhanced privacy protection in location based services. In: Proc. of LoCA. Volume 5561 of LNCS., Springer (2009) 70–87
8. Shokri, R., Theodorakopoulos, G., Troncoso, C., Hubaux, J.P., Boudec, J.Y.L.: Protecting location privacy: optimal strategy against localization attacks. In: Proc. of CCS, ACM (2012) 617–627
9. Shokri, R., Theodorakopoulos, G., Boudec, J.Y.L., Hubaux, J.P.: Quantifying location privacy. In: Proc. of S&P, IEEE (2011) 247–262

10. Dwork, C.: Differential privacy. In: Proc. of ICALP. Volume 4052 of LNCS., Springer (2006) 1–12
11. Machanavajjhala, A., Kifer, D., Abowd, J.M., Gehrke, J., Vilhuber, L.: Privacy: Theory meets practice on the map. In: Proc. of ICDE, IEEE (2008) 277–286
12. Ho, S.S., Ruan, S.: Differential privacy for location pattern mining. In: Proc. of SPRINGL, ACM (2011) 17–24
13. Chen, R., Ács, G., Castelluccia, C.: Differentially private sequential data publication via variable-length n-grams. In: Proc. of CCS, ACM (2012) 638–649
14. Andrés, M.E., Bordenabe, N.E., Chatzikokolakis, K., Palamidessi, C.: Geoidistinguishability: differential privacy for location-based systems. In: Proc. of CCS, ACM (2013) 901–914
15. Zheng, Y., Xie, X., Ma, W.Y.: Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* **33**(2) (2010) 32–39
16. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., Huang, Y.: T-drive: driving directions based on taxi trajectories. In: GIS. (2010) 99–108
17. Chatzikokolakis, K., Palamidessi, C., Stronati, M.: A predictive differentially-private mechanism for mobility traces. *CoRR* **abs/1311.4008** (2013)
18. Reed, J., Pierce, B.C.: Distance makes the types grow stronger: a calculus for differential privacy. In: Proc. of ICFP, ACM (2010) 157–168
19. Chatzikokolakis, K., Andrés, M.E., Bordenabe, N.E., Palamidessi, C.: Broadening the scope of Differential Privacy using metrics. In: Proc. of PETS. Volume 7981 of LNCS., Springer (2013) 82–102
20. Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: Proc. of STOC. (2010) 765–774
21. Gams, S., Killijian, M.O., del Prado Cortez, M.N.: Show me how you move and i will tell you who you are. *Trans. on Data Privacy* **4**(2) (2011) 103–126
22. Hardt, M., Rothblum, G.N.: A multiplicative weights mechanism for privacy-preserving data analysis. In: FOCS, IEEE (2010) 61–70
23. Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N.: Differential privacy under continual observation. In: STOC, ACM (2010) 715–724
24. Terrovitis, M.: Privacy preservation in the dissemination of location data. *SIGKDD Explorations* **13**(1) (2011) 6–18
25. Krumm, J.: A survey of computational location privacy. *Personal and Ubiquitous Computing* **13**(6) (2009) 391–399
26. Shin, K.G., Ju, X., Chen, Z., Hu, X.: Privacy protection for users of location-based services. *IEEE Wireless Commun* **19**(2) (2012) 30–39
27. Hoh, B., Gruteser, M.: Protecting location privacy through path confusion. In: Proc. of SecureComm, IEEE (2005) 194–205
28. Dewri, R.: Local differential perturbations: Location privacy under approximate knowledge attackers. *IEEE Trans. on Mobile Computing* **99**(PrePrints) (2012) 1
29. Cheng, R., Zhang, Y., Bertino, E., Prabhakar, S.: Preserving user location privacy in mobile data management infrastructures. In: Proc. of PET. Volume 4258 of LNCS., Springer (2006) 393–412
30. Ardagna, C.A., Cremonini, M., Damiani, E., di Vimercati, S.D.C., Samarati, P.: Location privacy protection through obfuscation-based techniques. In: Proc. of DAS. Volume 4602 of LNCS., Springer (2007) 47–60