Pervasive and Mobile Computing 90 (2023) 101755

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Privacy preservation for spatio-temporal data in Mobile Crowdsensing scenarios

Federico Montori [a,*], Luca Bedogni [b]

[a] *University of Bologna, Italy*
[b] *University of Modena and Reggio Emilia, Italy*

A R T I C L E   I N F O

Article history:
Received 28 October 2022
Received in revised form 18 January 2023
Accepted 31 January 2023
Available online 2 February 2023

Keywords:
Mobile Crowdsensing
Privacy
Location based service

A B S T R A C T

Mobile Crowdsensing has become an important paradigm in the last decade for on-demand monitoring scenarios in Smart Cities and vehicular networks, when the deployment of a dedicated sensor network is no longer affordable. To foster the participation of a large user base, it is common to reward them on top of the amount and the quality of data provided. Regardless of the MCS policy adopted, this requires the crowdsourcer to keep track of the participants. Since the contributed data inherently carries sensitive spatio-temporal information, privacy problems arise if a malicious entity gains access to it; still, in some cases, the spatio-temporal precision is crucial for the benefit of the application and cannot be distorted. In this paper we propose a privacy preserving framework for opportunistic MCS scenarios that includes data collection and rewarding phases. The framework both retains the precision of spatio-temporal information and limits the sensitivity of information disclosed through an algorithm that clusters the data points into low correlated sets. The framework is agnostic about how correlation is calculated, and we propose three exemplary correlation functions. We evaluate our framework against six real world datasets, assessing its efficacy and envisioning its implementation in practical deployments.

1. Introduction

Smart Cities are built upon the data that describes phenomena occurring in the urban territory, in order to monitor events of interest as well as being able to react timely and precisely for the common benefit. Smart Cities are then unavoidably tied to the concept of the Internet of Things (IoT), as raw data is usually provided by sensors installed at certain points in the monitored area. The aggregation of such data in the fog or in the cloud provides then enhanced services for the community [1]. This is an extremely challenging scenario, as manual installation and maintenance of sensors involve high costs, which become unbearable when it comes to achieving a wide coverage of large and complex environments. Moreover, novel Smart Cities-enabled applications are becoming more and more on-demand, hence demanding a flexible architecture.

For these reasons, the paradigm of Mobile CrowdSensing (MCS) gained worldwide popularity in the last decade [2]. Founded on the principles of crowdsourcing, MCS leverages the usage of mobile devices owned by the citizens, who participate in the sensing campaign and referred to as *participants*. The campaign is then issued by a *crowdsourcer*, who typically distributes to the participants a client application that is able to provide the requested data by leveraging the

* Corresponding author.
E-mail address: federico.montori2@unibo.it (F. Montori).

https://doi.org/10.1016/j.pmcj.2023.101755

sensors installed on the device. Data is then sent back to a cloud entity for later processing and aggregation, representing the only cost physically sustained by the crowdsourcer. Of course, this brings in additional challenges, such as how to assess the quality of data [3], how to recruit participants in push-based scenarios (*i.e.* where it is possible to directly issue them with tasks) [4], how to distribute the data collection in order to avoid wastage of resources and the occurrence of sparse and dense scenarios [5]. Furthermore, it is essential to ensure a sufficient coverage of the monitored phenomenon, which translates in a high user participation. This challenge is addressed by incentivization techniques, which require to reward the participants with either a monetary compensation or an adequate set of services [6]. Data collected through MCS is often geolocalized and reporting a time, in order for the crowdsourcer to characterize the monitored phenomenon. This raises privacy concerns, as a potential attacker could extract sensitive information from the series of data points reported by a participant which include sensitive information. These include points of interest, daily routines and the home location of the user, which eventually results in possibly understanding political orientation, religious beliefs and other private information [7]. For these reasons, privacy in MCS is a major concern in any scenario, often resulting in adopting concepts based on k-anonymity [8] and differential privacy [9]. While being a efficient expedients, they hinders the precision of the reported location and time, which cannot be acceptable to some applications that have precision requirements, *e.g.* [10].

In this work we address the above mentioned challenge by proposing a privacy-aware framework that preserves location and time precision without any perturbation on the single data point. We consider opportunistic scenarios, which differ from participatory ones in that measurements are not triggered actively by the user. Instead, the data collection process takes place in the background. We consider pull-based scenarios, where the crowdsourcer does not directly recruit participants, instead they contribute by uploading their measurements online spontaneously. In fact, for the dual scenario, known as push-based, the topic of privacy has to be discussed differently, as, in such case, the central entity is aware of the location of users at all times in order to issue them with tasks. Although the data collection process in pull-based scenarios is less affected by privacy issues, the rewarding phase, in which participants receive a prize for the contributed data, causes the central entity to keep track of the contributors. On top of these premises, our proposed privacy preserving framework allows participants to send their measurements atomically without being identified. In exchange, they get back as many tokens, which are then used by the participant in sets of a defined size to claim rewards by sending them over to the server. Token sets are selected in a way in which two or more sets cannot be directly associated to the same participant. Still, tokens within each set can be associated with the respective data points, however, we propose an algorithm that minimizes the data correlation within each single setThis ensures a minimal amount of significant information to be released, without perturbing the single measurement at any stage. Differently from our preliminary work [11], here we consider the interdependence of different observations within a defined set, instead of considering each measurement singularly. This causes us to consider, for each set of measurements, a matrix that outputs the correlation of each pair of measurement, implying a much higher mathematical complexity. Although for the purpose of this paper we designed three specific correlation functions to distribute the data point in sets, we also note that it is possible to define other correlation functions according to definite mathematical properties defined by our framework. We then quantify the privacy leak by using a scenario-specific privacy metric (*i.e.* the distance from a point of interest), although not limiting the applicability of different privacy metrics to our framework [12]. We perform extensive experiments over six real world datasets, showing how the system adapts to different conditions and data granularities. This is a major improvement over our previous work [11], in which we considered only one correlation function and two datasets.

The rest of this paper is structured as follows: Section 2 presents related works from literature; Section 3 outlines our proposed framework and discusses the threat model; Section 4 details the mathematical basis and the algorithms for selecting measurements; Section 5 explains the correlation functions used to quantify the privacy issues; Section 6 presents the datasets and details the experiments; Section 7 discusses the results of the performance evaluation; Section 8 concludes this study.

2. Related works

MCS is an actively studied topic that attracted researchers from all over the world over the last years. The challenges presented by this paradigm are various and all of them are investigated thoroughly in literature [2]. In particular, studies on incentivization techniques show that users tend to be more willing to share their data with a third party organization if that leads to a reward of some kind [6]. In general, rewards can be broadly categorized into two different classes: monetary or non-monetary. Monetary rewards grant the user a concrete prize, being it money or valuable goods, therefore causing the crowdsourcer to actually invest on the campaign. Non-monetary rewards, on the other hand, exploit methods like gamification or user competitiveness, awarding the winners with services. Rewarding users, especially according to their data quality and accuracy, unavoidably leads to privacy-related issues, in fact the accuracy/privacy trade-off is a long known issue in MCS [13]. Many existing works in literature leverage private information enclosure in order to cope with common de-anonymization oriented attacks (*e.g.* collusion and eavesdropping). In particular, recent works in MCS tackle the privacy issue by applying extensively experimented and solid methods such as k-anonymity [8], t-closeness [14], l-diversity [15] and differential privacy [9]. For instance, the work in [16] operates in sparse MCS scenarios by obfuscating the location information of participants, designing both an optimal trade-off between the data alteration and the information outcome as well as an inference algorithm to minimize the information loss. In [17], the authors design a privacy-aware task allocation paradigm by counterbalancing the amount of sensitive information shared by the participants (such as location and points of interests) with the ultimate efficiency. In [18], the authors design an auction-based recruitment framework in which participants inject noise in their bid, obscuring part of their personal information. The work in [19] makes use of location cloaking and collaborative caching in continuous reporting scenarios to confuse potential attackers able to infer the trace of users based on the spatial and temporal correlations of their observations. A different approach is taken in [20], where privacy is used as a trading currency, the more privacy the participant requires for herself or himself, the lower is the probability to be recruited. We observe that the vast majority of the works in the related literature, in order to ensure a degree of privacy, alter in some way parts of the data shared with the crowdsourcer, most of the times obscuring the precision of the location of the measurements. While this has been proven to be effective in many scenarios, there are still situations in which imprecise location can jeopardize the utility of the collective application. For such reason, differently from the works cited here, in this paper we focus on preserving the privacy of users while still guaranteeing precise location information from the crowd. Furthermore, we recall that this work focuses mainly on opportunistic MCS scenarios with continuous reporting, therefore, differently from works like [17,21], explicit recruitment is not performed.

For push-based scenarios there are many works which address privacy from an architectural point of view. Clearly in push-based scenarios the central server must know the identity of the participants, in order to assign tasks and receive data back. Therefore, privacy of the users is typically enforced on the remote server itself, by using a series of techniques such as the anonymization of users by aggregating data remove single measurements from the database or by using pseudonyms, by obfuscating data adding noise to it, or by using tessellation or micro-aggregation [22]. There are also specific attacks which can be performed on the task selection or the location of it, which are typically tackled by limiting the amount of data shared. Nevertheless, users typically have the ability to customize their privacy preferences in order to set tailored constraints on the application behavior [23].

3. Framework

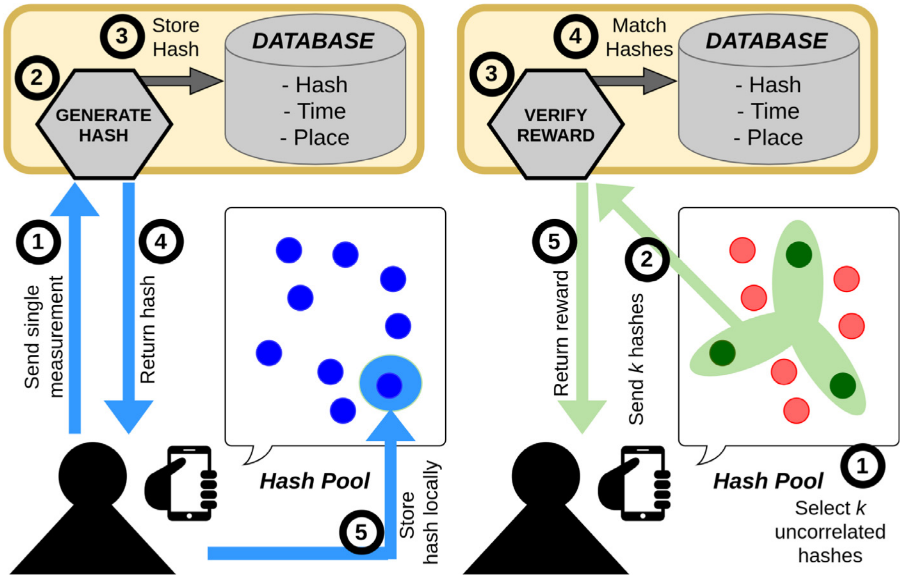
In this section we detail our proposed privacy-aware framework. We recall that in our framework the central server does not store any direct association that can lead to identify the participant, as the information itself is sent anonymously and atomically from the participant to the server. In other words, the framework must be supported by a data collection policy in which the server is unaware of the identity and the location of each user. As highlighted in Section 1, the main challenge in these architectures is the design of a mechanism that rewards users for their contribution, without the need for data alteration. To claim their compensations for their contributed data, participants would intuitively need to expose their identity and, consequently, the history of their location. This would hinder the privacy by design guaranteed by the data collection architecture; our proposal aims to fill this gap.

The architectural model of our proposal is shown in Fig. 1. Let P be a generic client participating in the considered MCS campaign (from now on we will use the term *client*, *user* and *participant* interchangeably). As the participant moves in the area of interest for the campaign, it sends **sensor measurements** or **observations** to the central server, with a rate that depends on the considered data collection framework. Let then I_i be the ith observation sent by P to the central server and let us define I_i as follows:

$$I_i = \{l_i, t_i, D_i, d_i\},$$

where l_i is the location at which the observation has been performed, t_i is the timestamp, D_i is the type of data reported and d_i is the numerical value of the measurement. The whole set of chronologically ordered observations $\{I_1, \dots, I_n\}$ performed by P is defined as the *trace* of P. As introduced earlier, even though the observation I_i is reported by user P, the server has no evidence about the identity of the user who reported the measurement as the tuple only contains data referring to the observation itself. Once the observation tuple I_i is received, the server generates an associated token hash h_i, that represents uniquely I_i. Such token is both stored by the server and sent back to P in response, which stores it in its local memory. Therefore, after generating the hash based on I_i and sending it back to the client, the server does not own any data that binds I_i to a user. The process described is depicted in Fig. 1(a).

All the hashes stored locally on the client device can then be used to collect the relative rewards from the entity running the MCS campaign, by sending a set of them to the central server to collect the prize. The rewarding phase is depicted in detail in Fig. 1(b). We assume that a reward can be released in exchange for a set of k hashes with $k \geq 1$, which is proportional to the amount of data contributed to the campaign. Once the hashes are received, the server marks the correspondent local copies and returns a single code in exchange, which can be used by the participant to collect the final prize (*e.g.* a QR code). The way in which such code is physically used for collecting the reward is out of the scope of this paper, however, we note that the exchange of information taking place in the rewarding phase, again, does not contain neither the identity of the user, nor a pseudonym. This means that any subsequent reward that is redeemed by the same user cannot be associated to the same person by the server directly. Additionally, this means that the server is never aware of the total number of participants. Because the server marks the local copies, users cannot redeem a hash more than once to obtain rewards. More formally, we define each MCS campaign with a set of rewards $\mathcal{R} = \{R_w\}$. A reward R_w needs k_w distinct hashes in exchange to be obtained. For the sake of simplicity and without loss of generality,



(a) Data collection phase (b) Rewarding phase

Fig. 1. A model showing how participants send measurements (a) and how they get eventually rewarded (b). The succession of actions are identified with sequential numbers.

in this work we assume that all the prizes need the same number k of hashes to be obtained. The goal is then to define a function $f : I^n \rightarrow I^k$ that locally selects k hashes related to the measurements out of the set of available ones.

Whenever a participant wants to reclaim the hashes, it has to send them back to the central server. The way in which a participant chooses such k hashes has an impact on the amount of sensitive information disclosed, since then the respective k observations can then be associated to the same person: note that the association between a measurement I_i and its hash h_i is always known to the central entity. Moreover, if a malicious entity has access to the server, it may also have its locations exposed. In our framework, we tackle this core challenge by proposing a strategy in Section 4 for selecting k observations in order to release as little sensitive information as possible.

3.1. Privacy risks and threat model

Based on the proposed model, we discuss here the privacy threats. Data sent by the user is anonymous, hence, the threat occurs if multiple data points from the same user are available to a malicious entity. In this case, it is then possible to correlate such data points, and assess key information about the user routine, most visited places, and points of interest [24]. Attacks can then occur in three different places. First of all, on the mobile device itself, if the malicious user has access to it. However, in this case the attacker may monitor on its own the actions and data produced by the user, hence also monitoring the data sent to the MCS remote server does not add any more information. Secondly, they can occur on the communication medium, however, if we assume an external attacker, a malicious user can act as a man-in-the-middle, to eavesdrop communications between P and the server when the data is collected. In case this happens, the man-in-the-middle only obtains a single data point with no identification information, as the framework does not require authentication. In the rewarding phase however the man-in-the-middle may eavesdrop a set of k hashes which belong to the same user, through which it can try to reconstruct part of the trace of P. However, hashes themselves do not carry any sensitive information about the data points they refer to, therefore the sole access to the hashes does not disclose any private information about the participant. To be truly effective, the attacker must then eavesdrop multiple communications from the same user. While this is possible, given that we are targeting a mobile scenario, it translates into the attacker continuously acting as man-in-the-middle, which is very unlikely. Even if this happens, then it means that the attacker has access to a much greater set of information about the user than only knowing the sporadic data points sent to the MCS remote server, making the privacy mechanisms discussed here a much lesser concern.

Finally, the attack can take place on the server: an external attacker may leak all the tuples I_i stored in the server, alongside with their hashes. In case this occurs, the attacker has a large amount of information, however none of the data points is bound to any user nor pseudo-id. Hence, it cannot directly correlate any couple $\{I_i, I_j\}$ of measurements, as it cannot discern which one belongs to the same participant. Also this attack alone does not pose a threat on users' private information. In this scenario, the risk mitigation has to be performed on the selection of the k hashes to be sent to the server, by identifying those that, in case they are stolen, would reveal as little information as possible about the locations of the user. There are many different privacy metrics that can be used for this selection; our proposed framework is designed to accommodate different ones, as better explained in Section 4.1

Privacy impact assessment. The above discussion assumes a trustworthy server, which does not deviate from the protocol. However, an attacker hacking the server completely may have access to the whole database of information about user collected data, and can also monitor subsequent queries made by the users in order to understand which hashes are correlated. This would basically be equivalent to the server itself being malicious and can happen in a number of ways, such as SQL injection attacks [25]. This is one of the major concerns related to protecting the user privacy in MCS [26] and it is also why data integrity and security is an issue in these systems, as many Common Vulnerabilities and Exposures (CVE) are possible [27]. It is important to remark that, in such a scenario, a number of additional considerations have to be made. Let us assume that a malicious adversary completely hacks the server, having thus at hand the full set of measurements, which we define as M, the selection function f and the set of k hashes sent over by a certain user. The attacker may attempt to run the function f on every subset of M until it gives exactly those k hashes as an output, finding thus the n data points that such user has sent to the server so far. The attack can in theory be conducted, constituting then a risk for the presented scenario. However, let us consider the following two aspects:

1. The attacker does not know n. A brute-force attack implies that the attacker must run the algorithm f over all members of the powerset of M that contain the k elements, resulting in $2^{|M|-k}$ iterations, only to find one of the users. This number may be mitigated by iteratively adding measurements from M to the initial subset containing the k hashes and stopping to a realistic value of n, however, it still results in a significant computational effort in its worst-case complexity.
2. f is not injective, which means there may easily exist two subsets of M, namely M_1 and M_2 such that $f(M_1) = f(M_2)$. An attacker may then find more than one subset that would yield the k hashes as a result, and many of them contain measurements not belonging to the same participant.

Upon these premises, we acknowledge that the above mentioned attack can in theory be conducted, however, if applied to a sufficiently consistent set of observation, its practical impact is likely to be negligible. We in fact expect common and practical MCS scenarios to easily feature thousands of measurements and thousands of users, hence to be reasonably safe against this type of attack. To give a practical idea, let us take into account the datasets used in this paper (detailed information on them can be found on Section 6.1) which have a size that approximately spans from a minimum of 220,000 observations to a maximum of 6.5 million. If we assume realistic values such as $k = 10$ and $n = 100$, let us define M_γ as any n-sized subset of M that may belong to the attacked user, thus such that contains the k elements that are known. An attacker would need to try to run f over all possible subsets M_γ, which can be calculated as $\binom{M-k}{n-k}$. If we consider $M = 220,000$ (our smallest dataset), the number of eligible subsets becomes intractable (over 10^{300}), and this is only for subsets of size n. Given that the attacker does not even know n, the above calculation should be repeated similarly for every single value of n that the attacker is willing to consider as plausible. Furthermore, let us estimate how many of these subsets M_γ can be erroneously attributed to the attacked user. This happens when the execution of $f(M_\gamma)$ outputs exactly our k elements. The probability that this happens is 1 over $\binom{n}{k}$ (less than 10^{14}). However, as stated above, M_γ can assume $\binom{M-k}{n-k}$ different values, thus we can expect the number of M_γ possible subsets such that $f(M_\gamma)$ outputs our k values to be roughly over 10^{300-14}. The above statement, again, assumes that the attacker knows n, which is not the case, therefore we expect these numbers to be much higher. Obviously, these considerations hold in presence of uniform probabilities without contextual information, however, it still gives a general idea about the difficulty of the attack from a mathematical standpoint.

4. The selection strategy

Assuming n as the total number of hashes available to P, in this section we define a criterion for choosing k observations (and their respective hashes) out of n. We first outline formally the concept of correlation, then we show how it is used to perform the selection. A key remark here is that a user must have performed already n observations in order to apply this algorithm offline. This implies that, depending on the use case, for continuous MCS campaigns there will be a certain threshold for n to be considered "big enough" to trigger the selection of k observations. We do not focus here on how to choose n due to the heterogeneity of use cases, however, the general indication is that $n \gg k$. If, instead, the MCS campaign is limited in time, n will be the number of observations collected by a certain participant at the end of the campaign.

4.1. Definition of correlation functions

Ideally, for each reward, the user wants to select a subset of data that yields the least possible sensitive information to a potential attacker. We choose to quantify the "sensitivity" of each piece of information by assigning it a *correlation value*, which represents how much the observation is related to the pool of measurements collected. We define a **correlation function** $\Gamma : I^n \rightarrow Corr$ that associates each observation (or group of observations) to a general concept of correlation $Corr$. This function is computed locally by the participant against all the elements of $\{I_1, \ldots, I_n\}$, then the least k correlated elements are selected to be converted into a reward. Within the framework it is possible to define multiple correlation functions $\Gamma_1, \ldots, \Gamma_m$ that establish the criteria for choosing k observations (and their respective hashes) out of n, in short, each of them should represent an aspect of correlation that could be potentially used by an attacker. We intentionally define the *Corr* value as generic data meta-type; it can be a single numeric value as well as a composite data structure that takes into account every other observation in the pool. Within the scope of this paper, we provide as a specific case a definition of correlation with strict mathematical properties. In particular, differently from our previous work [11], we do not define correlation functions on top of a single input, as the relation with all the other observations in the pool should be considered. Instead, a correlation function is formally defined here as a function $\Gamma : I^n \rightarrow \mathbb{R}^{n^2}$ that takes in input the trace of the user, consisting of n elements, and returns in output a $n \times n$ **pairwise correlation matrix**. This allows to better select the hashes to send not only in a greedy way, but also considering future reward claims. Moreover, this way we are ensuring that the selection of k elements strongly depends on the features of the n total elements on which the selection algorithm is executed. Finally, we also significantly extended the performance evaluation on more dataset, which thanks to their heterogeneity provide a more complete overview of the performance figures of our proposal. The pairwise correlation matrix $C^{\{p\}}$ is the result of the application of $\Gamma_p(\{I_1, \ldots, I_n\})$, in which each element of the matrix $C^{\{p\}}_{i,j}$ is the pairwise correlation of the elements I_i and I_j of the trace. This choice is driven by the consistency of the mathematical definition of a metric, which is often built on top of the relationship between couples of elements [12]. Each correlation function has to satisfy the following three properties:

1. An element of the matrix $C^{\{p\}}_{i,j}$ is a correlation in the form of a numerical value normalized by [0, 1] (1 means that the elements are maximally correlated, 0 means that the elements are not correlated at all):
$$\forall i, j : C^{\{p\}}_{i,j} \in [0, 1].$$
2. Each element is maximally correlated with itself. Therefore, by construction, the main diagonal of the correlation matrix is made by ones:
$$\forall i : C^{\{p\}}_{i,i} = 1.$$
3. The correlation value between two hashes is commutative, therefore the correlation matrix is symmetrical with respect to the main diagonal:
$$\forall i, j : C^{\{p\}}_{i,j} = C^{\{p\}}_{j,i}.$$

Optionally, a correlation function may also be **mutual independent**, defined as follows: a correlation function is mutually independent if the calculation of each element of the matrix depends only on the two observations taken into account and does not change upon adding or removing observations. More formally if

$$\forall i, j : C^{\{p\}}_{i,j} = f(I_i, I_j).$$

The definition of unique and universal correlation functions is out of the scope of this paper as the sensitivity of a piece of information can depend on a high number of factors with respect to different use cases. We recall that our framework is totally independent on the specific correlation functions used, hence it is flexible to accommodate other methodologies to quantify privacy risks on top of the MCS environment taken into account. We formally propose in Section 5 a number of specific correlation functions that address different aspect of privacy concerns, which we will use throughout the experiments.

Before getting into the selection algorithm, it is necessary to define the concept of **average correlation** of a pairwise correlation matrix as the arithmetic mean of all the values of the matrix *except* the ones on the diagonal:

$$avg(C^{\{p\}}) = \frac{\sum_{i=1,j=1}^{n,i\neq j} C^{\{p\}}_{i,j}}{n^2 - n}.$$

We then define two strategies: the greedy (Section 4.2) and clustering selection (Section 4.3).

4.2. Greedy selection of observations

The goal of this strategy is to select k observations out of n such that their average pairwise correlation is minimum (*i.e.* the average pairwise correlation of the submatrix formed by the selected k observations). It is easy to recognize that such problem is NP-hard as it requires to scan all the possible combinations of k elements over n (it can be conducted to the maximum volume problem in [28]), however it can be easily approximated by the use of a heuristic. First of all, the correlation matrix can be seen as an adjacency matrix and, therefore, it is easily translatable into a fully connected undirected graph in which each edge is weighted with the correlation value between the nodes. We will refer to it as the **correlation graph**: for each matrix $C^{\{p\}}$, we will refer to its respective correlation graph with $G^{\{p\}}$. The heuristic works as follows: the $n - k$ nodes with the highest degree – the degree of a node in a weighted graph is obtained by summing up the weights of all the adjacent edges – are removed from the graph iteratively, which means that, upon the removal of one node, the degrees are being recalculated. The remaining nodes denote the selected observations, representing a

Algorithm 1: Clustering Selection Heuristic.

```
Require: Empty cluster matrices $\Phi = \{\phi^{\{1\}}, \dots, \phi^{\{q\}}\}$,
    Observations $I_1, \dots, I_n$
    Cluster size $k$
Ensure: $(avg(\phi^{\{1\}}) + \dots + avg(\phi^{\{q\}}))/q$ is minimum
 1: Perform initial assignment of observations to clusters
 2: $corr = (avg(\phi^{\{1\}}) + \dots + avg(\phi^{\{q\}}))/q$
 3: $index := 0$
 4: for $iter := 1$ to $MAXITER$ do
 5:   $L$ = Sort the $I_i$ observations by $h(I_i)$, descending order
 6:   $I_{swap} := L[index]$
 7:   for all $I_j \in L$ such that $I_j$ and $I_{swap}$ are in a different cluster do
 8:     $swap(I_{swap}, I_j)$
 9:     $corr_{tmp} = (avg(\phi^{\{1\}}) + \dots + avg(\phi^{\{q\}}))/q$
10:     if $corr_{tmp} < corr$ then
11:       $corr := corr_{tmp}$
12:       $index := 0$
13:       break
14:     else
15:       $swap(I_j, I_{swap})$   # invalidate the swap
16:     end if
17:   end for
18:   if no swap has been done then
19:     $index++$    # convergence if index out of bound
20:   end if
21: end for
```

local optimum, for which this heuristic has a quadratic time complexity. The average correlation of their resulting pairwise correlation submatrix is denoted as $min(C^{\{p\}}, k)$. The process can be used to find also the submatrix yielding the maximum average pairwise correlation $max(C^{\{p\}}, k)$, *i.e.* selecting iteratively the $n-k$ nodes with the lowest degree. While the former is the goal of the selection, the latter is useful as a term of comparison. The greedy solution is an efficient baseline, however it fails to be practical as we will see in Section 4.3, where a more robust version is proposed.

4.3. Clustering selection of observations

Although the greedy selection of k best observations in Section 4.2 is conceptually optimum in the transient in which it is invoked, it is likely to be inefficient in the long run. In fact, as this selection is performed repeatedly over time, in order to "spend" all the hashes in the pool, the participant will, sooner or later, end up with a bunch of hashes that will never be chosen as part of the k to be sent, as they might keep having a high correlation value and, therefore, the participant will hardly find a way to spend them. Our proposal to cope with this is to divide the set of observations in k-sized clusters, with the goal of minimizing the intra-cluster correlation, *i.e.* the mean value over the average correlations of each cluster (a cluster results in a $k \times k$ correlation matrix). This way, the participant is able to spend, one by one, the k-sized clusters of observations, for which the correlation value is not optimum, but it is considered acceptably low. The problem itself is an instance of the min–max clustering problem [29], which is again NP-complete, as it would imply to try out all the possible combinations of cluster assignments. We therefore propose a clustering heuristic, based on the Expectation–Maximization (EM) paradigm.

We initially calculate the number of k-sized clusters as $q = \lfloor \frac{n}{k} \rfloor$. This unavoidably causes $n \bmod k$ observations to be left out; for simplicity in this section we consider n to be a multiple of k, if this is not the case, the algorithm works in the same way although having an additional smaller cluster. Each cluster is represented by the $k \times k$ correlation submatrix of the observations assigned to it. The set of submatrices is defined as $\Phi = \{\phi^{\{1\}}, \dots, \phi^{\{q\}}\}$. We also define the distance of an observation I_i from its cluster $\phi^{\{j\}}$ as the sum of the correlation between the observation and all the other observations in the same cluster: $h(I_i) = \sum^{v} \phi^{\{j\}}_{u,v}$, assuming that row u in $\phi^{\{j\}}$ denotes information I_i. The clustering procedure is detailed in Algorithm 1.

Specifically, we first perform an initial assignment of observations to clusters. In our case, we shuffle the observations and then assign them, one by one, to a non-full cluster for which their distance is minimum. We then sort the I_i observations by their $h(I_i)$ (line 5) in descending order, assuming that the first ones, having their $h(I_i)$ high, would be the most likely to be swapped with another observation from another cluster. Next, we pick the first observation from the list as a swap candidate (line 6) and iterate over all the other observations that do not belong to the same cluster (line 7). For each of them we try to perform the swap (line 8) and recalculate the correlations (line 9). If the swap does not lower the average correlation, then we cancel it and revert to the previous state (line 15) and, if no swap has been done after scanning all the possible ones, then we go to the next swap candidate (line 19). If, instead, the swap lowers the average correlation, we commit it, break this cycle, re-sort the observations and start over from the first observation in the list as a swap candidate (lines 10–13). This happens because performing a swap may change the conditions and other swap candidates that we already investigated may need to be reconsidered. If we took into account all the observations without performing any swap, we found a local optimum, otherwise we stop when the maximum number of iterations (identified as *MAXITER*) has been reached. Note that the algorithm still works even if n mod $k > 0$, since swaps can be performed against observations outside the clusters (or in a smaller cluster) and the minimization metric is calculated globally (line 10). Once the clusters are defined, they remain fixed until the participant redeems all of them singularly. This ensures that observations in the same cluster are correlated minimally with each other, with the constraint of redeeming all the observations in the pool. This happens efficiently if the clustering selection is performed when n mod $k = 0$. We then denote the mean over the average correlations of the clusters as

$$min\_clus(C^{\{p\}}, k) = \frac{avg(\phi^{\{1\}}) + \cdots + avg(\phi^{\{q\}})}{q},$$

where $C^{\{p\}}$ is the correlation matrix of all the observations of the user trace. We can easily obtain its dual, $max\_clus(C^{\{p\}}, k)$, by maximizing the correlation, instead of minimizing it, in line 10 of Algorithm 1.

For better clarity, let us consider a simple example of a user who submitted $n = 4$ observation and wants to claim the rewards. In this particular case, we assume $k = 2$ and a unique correlation function, that holds the property of mutual independence and outputs the correlation matrix C', reported in the two examples below.

	1	**2**	**3**	**4**
1	1.0	0.4	0.3	0.1
2	0.4	1.0	0.9	0.7
3	0.3	0.9	1.0	0.2
4	0.1	0.7	0.2	1.0

$\Rightarrow$ (Greedy) $\Rightarrow$

	1	**2**	**3**	**4**
1	🟩 1.0	0.4	0.3	🟩 0.1
2	0.4	🟪 1.0	🟪 0.9	0.7
3	0.3	🟪 0.9	🟪 1.0	0.2
4	🟩 0.1	0.7	0.2	🟩 1.0

	1	**2**	**3**	**4**
1	1.0	0.4	0.3	0.1
2	0.4	1.0	0.9	0.7
3	0.3	0.9	1.0	0.2
4	0.1	0.7	0.2	1.0

$\Rightarrow$ (Clustering) $\Rightarrow$

	1	**2**	**3**	**4**
1	🟪 1.0	🟪 0.4	0.3	0.1
2	🟪 0.4	🟪 1.0	0.9	0.7
3	0.3	0.9	🟩 1.0	🟩 0.2
4	0.1	0.7	🟩 0.2	🟩 1.0

By applying the greedy selection algorithm, we end up with the submatrices highlighted on the right end side of the first line, where the green cells identify the first set of k observations chosen, while the violet cells highlight the second set — the leftovers. Evidently, the greedy selection first makes a very convenient choice, as the green submatrix has an average correlation of 0.1, however, the leftover observations are very correlated with each other, as the remaining submatrix has a high average correlation of 0.9. On the right end side of the second line, instead, are highlighted the two submatrices obtained by performing a clustering selection. Here, it is clearly visible how, because the clusters are all drawn at the same time, the best possible combination of sets is chosen and the mean of the average correlations, defined above, is minimized (0.3 for the clustering selection vs. 0.5 for the greedy selection). This aspect is also analyzed quantitatively over large datasets in Section 7.1.

5. Proposed correlation functions

In order to validate our model, we describe in this section the correlation functions we use to address the main aspects of location-aware privacy:

- Time correlation between measurements, addressed in Section 5.1, called **Absolute Distance in Time**.
- Habits in the user trajectory, addressed in Section 5.2, called **Location and Time-Aware Daily Transitions**.
- Sensitive locations, addressed in Section 5.3, called **Distance from a Point of Interest**.

We note again that, although we study these correlation functions, the main purpose of this work is to present the architecture and mechanism to obtain a privacy preserving, precise location data in MCS. Different correlation functions can be used in our framework by either using the pairwise correlation, as long as they cope with the requirements we described in Section 4.1. The correlation functions we present are in line with other privacy metrics leveraging data similarity discussed in [12]. We note that it is even possible to jointly use multiple correlation functions by combining them as in [30].

5.1. Absolute Distance in Time

This correlation function is meant to cover the time correlation between observations: the more two observations are close in time, the more privacy concerns may arise. As a matter of fact, exposing hashes relative to measurements

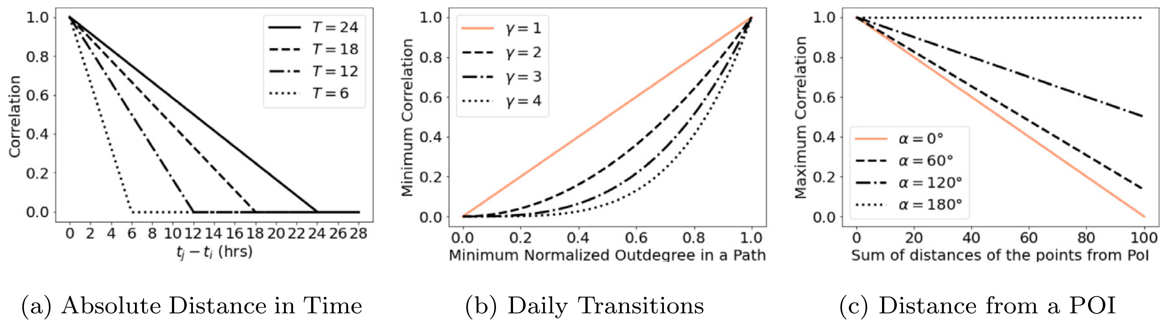


Fig. 2. Mathematical analysis of the proposed correlation functions.

that are all distant in time is much more likely to protect sensitive information regarding fine-grained movements of the participant. We define the Absolute Distance in Time correlation function (or simply Distance in Time) $\mathcal{T}^{\{\Delta t\}}$ starting from the actual time difference between the timestamps of two measurements. Δt defines a time slot, which is our atomic unit of time. Then, if two measurements are taken within the same Δt slot they have distance 0, otherwise their distance is calculated as the integer part of their actual time difference divided by Δt; *e.g.* if they are 15min apart and Δt =10min, then their distance is 1. We also normalize it by setting a maximum amount T of time slots beyond which all the distances are ideally the same (*e.g.* a day). Thus, our definition becomes:

$$\mathcal{T}^{\{\Delta t\}}(I_i, I_j) = 1 - \frac{\lfloor |t_j - t_i| / \Delta t \rfloor}{T}$$

of course, the result of the fraction is constrained to a maximum of 1 when $|t_j - t_i|$ is 0, while it is 0 when $|t_j - t_i| \geq T$, therefore the first property is satisfied. We can also easily see that $\mathcal{T}^{\{\Delta t\}}(I_i, I_i) = 1$ and $\mathcal{T}^{\{\Delta t\}}(I_i, I_j) = \mathcal{T}^{\{\Delta t\}}(I_j, I_i)$, therefore the other properties are satisfied as well. It is also easy to recognize that this correlation function fulfills the property of mutual independence, as the pairwise correlation between two observations does not take into account any other observation that may occur. A mathematical analysis of the behavior of this correlation function is presented in Fig. 2(a), where we show the correlation calculated on top of $\lfloor |t_j - t_i| / \Delta t \rfloor$ for different values of T. It is easy to see that points are less correlated at heir time difference increases.

5.2. Location and Time-Aware Daily Transitions

This correlation function addresses the sensitivity of releasing paths that are followed daily with a regularity by the participant. Obviously, their disclosure affects the user privacy as it might unveil part of his or her habits. The idea is therefore to attribute a high correlation to any couple of observations that are within a frequently traveled route. We define here the Location and Time-Aware Daily Transitions correlation function (or simply Daily Transitions) as $\mathcal{P}^{\{\Delta t\}}$, with Δt being the atomic unit of time for this correlation function.[1] It is also necessary to introduce, for each observation I_i, an additional attribute $\tau_i \in [0, 86400/\Delta t]$, defined as the time of the day extracted from t_i – recall that 86,400 is the number of seconds in 24hrs – divided by Δt. We then construct a fully connected graph such that each node is identified by a tuple $\langle l, \tau \rangle$, which represents the presence of the user in a location l at a defined Δt-sized time slot of the day τ, and each edge is initialized with weight equal to 0. Next, for each couple of *consecutive* measurements within the trace I_i, I_j, we increase the weight of the edge $(\langle l_i, \tau_i \rangle, \langle l_j, \tau_j \rangle)$ by one. Then, every weighted edge is normalized by the outdegree of its source node, therefore representing the probability of the user in doing such transition. In order to output a correlation matrix, we pick all the couples of measurements I_i, I_j and we extract from the graph the value of the edge $(\langle l_i, \tau_i \rangle, \langle l_j, \tau_j \rangle)$, or vice versa if I_i is more recent than I_j. That is the value of the correlation between the two. In order to cover a more complete definition of correlation, we extend the above definition with a more generic correlation function $\mathcal{P}_{\gamma}^{\{\Delta t\}}$. For each couple of measurements I_i, I_j, we define their correlation by summing up the probability of *all* the paths from $\langle l_i, \tau_i \rangle$ to $\langle l_j, \tau_j \rangle$ in γ steps or less. The probability of a path of $\{E_1, \dots, E_X\}$ ordered edges is defined as:

$$\prod_{x=1}^{X} weight(E_x).$$

The sum of such paths cannot exceed 1 in absence of cycles. If there are cycles we constrain the maximum value to 1, so the first property is preserved. We also define the correlation of two observations that are in the same node to be 1, thus the second property is satisfied. The definition is also commutative, as $\mathcal{P}_{\gamma}^{\{\Delta t\}}(I_i, I_j)$ evaluates the paths from $\langle l_i, \tau_i \rangle$ to

[1] It is not necessarily the same for each correlation function. Since correlation functions are calculated independently from each other, this definition does not clash with others as the scope of Δt is limited to this subsection.

$\langle l_j, \tau_j \rangle$ only if I_i happened before I_j, otherwise it does the opposite, therefore the third property is respected. This is an example of a correlation function that does not respect mutual independence, as the pairwise correlation between two observations is influenced by other observations in the same path.

The objective of this correlation function is to hide possible routines of the user. For instance, even if distant in time and space, a user may leave his home in the morning and continue to work, arriving several minutes later in a point far from his home. However, if repeated every day, this can easily uncover private information. This function is mathematically analyzed in Fig. 2(b), where we quantify the correlation on top of the minimum normalized outdegree found in a path that connects two selected points in the graph, for different values of the maximum path length γ. In particular, since this does not depend only on the two points, we give an estimate of the correlation by showing the minimum correlation (i.e. when all the other intermediate nodes have also the minimum outdegree) and the maximum correlation shown in lighter color (i.e. when all the other intermediate nodes have outdegree equal to 1). Clearly, if $\gamma = 1$ then the minimum and maximum correlation are the same.

5.3. Distance from a Point of Interest

With this correlation function we want to address the problem of releasing data about sensitive locations. For this correlation function, we leverage the concept of Point of Interest (POI), being it a location that is generally of interest for the user, and its disclosure could be potentially harmful. In [31] the authors try to guess the home of participants by using the location history and the centrality of their Twitter posts, showing the effectiveness of the proposed method. We consequently propose a correlation function $\mathcal{V}$ which profiles the closeness of measurements to a POI of the user. Such POI is, for simplicity, assumed to be at the center of the n observations taken by the user. For an observation I_i we define its latitude as $l_i^{\{y\}}$ and its longitude as $l_i^{\{x\}}$; therefore we define the Distance from a Point of Interest correlation function (or simply Distance from POI) as the inverse distance from their middle point to the POI:

$$\mathcal{V}(I_i, I_j) = 1 - vin\left(\left(\frac{l_i^{\{y\}} + l_j^{\{y\}}}{2}, \frac{l_i^{\{x\}} + l_j^{\{x\}}}{2}\right), POI\right),$$

where vin is the ellipsoidal Vincenty distance (in kilometers) between two points identified by latitude and longitude. This means that two points are correlated as much as their middle point is close to the center of the observations. As a corner case, we set $\mathcal{V}(I_i, I_j) = 1$ if the Vincenty distance is more than 1km, consequently ensuring the values on the diagonal of the matrix to be 1 and all the values in the matrix to be between 0 and 1, therefore the first two properties are satisfied. The calculation of the middle point is commutative with respect to the two input points, ensuring the third property to be respected. Finally, if we assume the POI to be fixed, then the correlation function possesses the mutual independence. The objective of this correlation function is to select points far from a POI. For instance a user may collect several points close to his home, in different days and at different times. However, if such points are disclosed altogether, they can easily reveal that indeed they are within a location of interest for the user [31]. This also applies for points that are far apart from each other but symmetrical with respect to the POI. Fig. 2(c) shows that the mutual correlation between two points depends on their distance from the POI and on the angle that the segments that connect them to the POI form. In the figure we represent the correlation as a function of the sum of the distances from the POI for different angles. We outline the maximum correlation (i.e. when their distance from the POI of the two points is the same) and the minimum correlation as a lighter colored line (when one of the points is exactly on the POI). The rationale here is straightforward, the closer the angle is to 180°, the higher the chance for the two points to have a high correlation, as the POI is more likely to fall close to their middle point.

5.4. Example

We now provide a simplified example to show how the above correlation functions work. Assume that a user reports measurements as in Table 1.

The three correlation functions will then output the following results:

- Correlation function 1, which is the Absolute Distance in Time, considers the differences between the timings of two subsequent measurements. In this example we consider $\Delta t = 3600$ (1 h) and $T = 24$ (1 day). Specifically for the measurements reported in Table 1 we will have the following correlations:

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	1	0.958	0	0	0	0	0
2	0.958	1	0	0	0	0	0
3	0	0	1	1	0.625	0	0
4	0	0	1	1	0.666	0	0
5	0	0	0.625	0.666	1	0	0
6	0	0	0	0	0	1	0.916
7	0	0	0	0	0	0.916	1

Note how correlation is not zero only when pair of measurements take place on the same day.

Table 1
Example measurements.

ID	Date	Time	Latitude	Longitude
1	1/1	8:30	44.42	11.31
2	1/1	9:59	44.39	11.37
3	12/1	8:40	44.42	11.31
4	12/1	9:30	44.39	11.37
5	12/1	17:50	44.40	11.35
6	24/1	9:35	44.39	11.37
7	24/1	12:30	44.41	11.35

Table 2
Example measurements.

Node ID	Measurements	Hour	Latitude	Longitude
1	{1, 3}	8:00	44.42	11.31
2	{2, 4, 6}	9:00	44.39	11.37
3	{5}	17:00	44.40	11.35
4	{7}	12:00	44.41	11.35

- Correlation function 2 instead considers the frequency of the paths, by constructing a graph and considering the frequency of travels from one point to another.
  Considering $\Delta t = 1h$ and $\gamma = 1$, in our simplified example, the unique time and locations we have in our dataset are then computed as those reported in Table 2, therefore their correlations become:

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	1	1	1	1	0	1	0
2	1	1	0	1	0.5	1	0.5
3	1	0	1	1	0	1	0
4	1	1	1	1	0.5	1	0.5
5	0	0.5	0	0.5	1	0	0
6	1	1	1	1	0	1	0.5
7	0	0.5	0	0.5	0	0.5	1

  We note that the correlation of measurements belonging to the edge that connects Node 1 with Node 2 is 1, because it is the only edge coming out from Node 1. Instead, from Node 2 there are two outgoing edges to Node 3 and Node 4 with equal probability.
- Correlation function 3 considers instead the distance of the middle point between two measurements and a specific POI for the user. For our example, the calculated POI has coordinates $POI = (44.402, 11.347)$. The correlations between each pair of points then become:

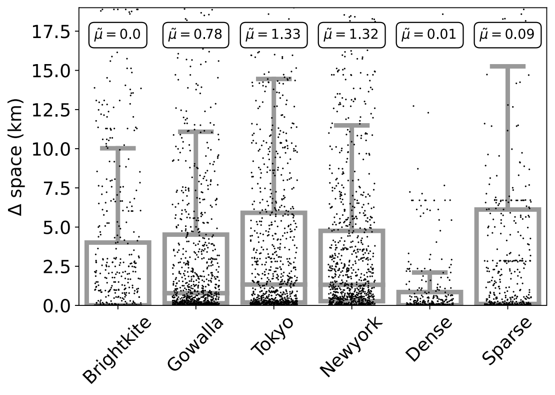
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	1	0.38	0	0.38	0	0.38	0
2	0.38	1	0.38	0	0	0	0
3	0	0.38	1	0.38	0	0.38	0
4	0.38	0	0.38	1	0	0	0
5	0	0	0	0	1	0	0.67
6	0.38	0	0.38	0	0	1	0
7	0	0	0	0	0.67	0	1

6. Performance evaluation

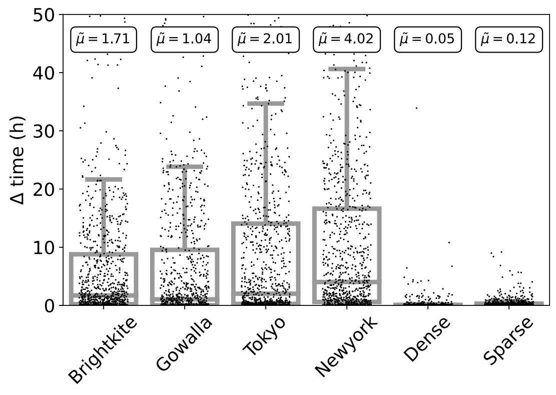
In this section we outline the performance evaluation study for our framework with our proposed correlation functions. First, in Section 6.1 we outline the real-world datasets against which our framework has been tested, then, in Section 6.2, we detail the parameters and the environment used for our ad-hoc simulation scenario.

6.1. Datasets

In order to evaluate our work, we used six different datasets, four of which are publicly available on the Internet. The datasets are check-in based, meaning that they record, for each user, a list of locations visited in GPS coordinates alongside with the timestamp. We chose such datasets because they have been used extensively in the literature for a multitude of related problems, also they have different spatial and temporal granularity, which yields different features [7]. In order to complement their granularity, relatively coarse, we introduced two more datasets for which granularity is much finer,



(a) Spatial distribution.



(b) Temporal distribution.

Fig. 3. Distribution of spatial and temporal intervals between each pair of consecutive measurements in the examined datasets. The strip plot shows only about 10% of the points for displaying purposes.

collected for the purpose of this study. Details about the structure, amount of data and granularity of each single dataset is reported in the description below:

- **Brightkite**: Brightkite [32] is a location-based social network and its dataset has been used in various studies on privacy. Observations are very distant in space even for the same user (order of kilometers), as well as in time. The dataset can be found online.[2]
- **Gowalla**: Gowalla [32] just like Brightkite, is a location-based social network and its dataset has been used in various studies on the privacy. Similarly to Brightkite, the granularity of the observations is coarse. The dataset can be found online.[3] Brightkite and Gowalla are often used in comparison studies as they have a similar structure.
- **Foursquare-Tokyo**: Foursquare [33] is yet another location-based social network, similar to the previous two, however, data gathering has been performed in a city-wide scenario, for which granularity of the observations is finer than in Gowalla and Brightkite. The page about the dataset can be found online.[4] It features several datasets, we chose to use the dataset about the city of Tokyo.
- **Foursquare-NewYork**: another dataset extracted from Foursquare [33], this time about the city of New York. The dataset can be found online on the same page as Foursquare-Tokyo.
- **Dense**: The Dense dataset has been populated by seven volunteers who had a mobile application running on their smartphone and reporting location, time and value of several sensors with a time granularity of few minutes. We broke the seven traces into many parts simulating the presence of many more users, as we do not need traces longer than few hundreds of samples. This strategy does not contaminate the nature of the dataset for our experiments, as each user is independent. The dataset was obtained from [7].
- **Sparse**: The Sparse dataset is exactly the same as the Dense, however, we sampled the measurements at a coarser granularity in a way in which they are spaced in time by an hour in average, in order to test a different event frequency.

Each dataset has been adapted in order to contain data about 100 users, with each trace being exactly 90 observations long, for the sake of the comparison. Due to the extreme heterogeneity of spatial and temporal differences within each datasets – certain users may turn off their sensing for many days and/or travel significant distances – we extracted only traces containing 90 subsequent observations such that the time difference between two consecutive observations is less than a week and the spatial distance is less than 100 km. Subsequently, location data has been enhanced in order to identify not only GPS locations, but also areas.

Fig. 3 shows the distribution of space and time differences in the six datasets by means of a boxed strip plot. Each dot in the plot shows the distance in space (Fig. 3(a)) and in time (Fig. 3(b)) between two subsequent observations in the same trace and the mean value is shown through the horizontal line inside each box, as well as written on the top. Regarding the spatial distribution, we observe that, first of all, there are consecutive observations that are tens of kilometers apart from each other – the plot is limited on the *y*-axis for improved readability – however, these are mostly outliers, as the vast majority of the points are clustered in the lowermost zone of the plot, below about 5 km. The only exception to this is the Dense dataset, which clusters the observation in way smaller distances. We observe that for Brightkite, even though

2 https://snap.stanford.edu/data/loc-Brightkite.html

3 https://snap.stanford.edu/data/loc-gowalla.html

4 https://sites.google.com/site/yangdingqi/home/foursquare-dataset

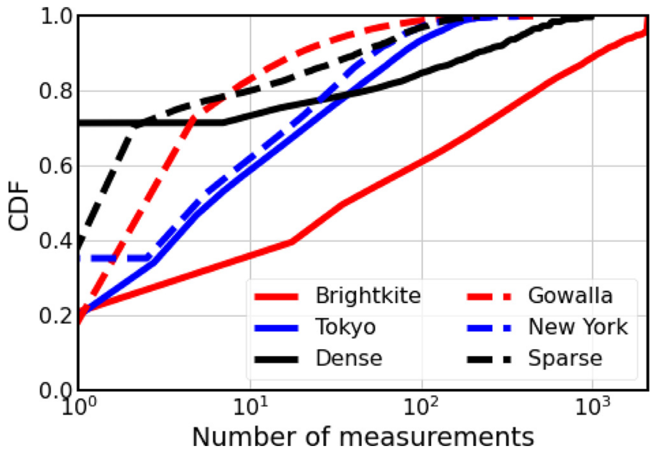


Fig. 4. CDF of the number of observations taken in a square area with the number of measurements on the x-axis.

data points can be very distant, there are many observations taken consecutively in the exact same place, as the mean value is 0. Conversely, in the Foursquare datasets, users show much more mobility as their spatial differences are more accentuated, even though the areas considered are smaller. Regarding the time distribution, we observe how almost all the data points for the Dense and the Sparse datasets are within minutes in time even though the Sparse dataset is sampled at higher time intervals. The other four datasets display common behaviors, with the Foursquare datasets showing longer time intervals between measurements compared to Brightkite and Gowalla. These 6 datasets provide heterogeneity in the data and can better highlight differences among them.

In Fig. 4 we show another aspect that occurs differently over the datasets. The figure displays a CDF in which we represent the number of observations taken in an area in which there is a number of total observations equal to the number represented on the x-axis. Here, we denoted the area by approximating the GPS coordinate of each observation to the third decimal.[5] As the curves are mostly placed against the upper left corner, we chose to use a logarithmic scale for the x-axis to emphasize the differences. From this plot we can notice how certain datasets have recurrent places in which observations happen compared to others. Brightkite shows clearly how many of its data points are in the same location – 10% of them are in a zone that hosts more than 1,000 other data points –, which does not happen for Gowalla. The distributions and behaviors shown in Fig. 3 and 4 are of paramount importance in order to characterize and describe better the results of our experiments.

6.2. Experimental setup

To evaluate our framework we performed simulations, for each dataset, with 100 users, each of them performing 90 measurements. We then extract, at certain points in time depending on the experiment, the k observations selected to claim the reward. As we consider attackers to have a local knowledge of the information – *i.e.* k measurements belonging to one user – it is not necessary to consider the scalability of the approach to a larger user base, since each attack is performed separately on each cluster of k measurements.

The value of k in each simulation is set to a unique value for all users. In real deployments, the value of k might differ across users and even across prizes for the same user over time, however, fixing it to one value facilitates the display of the results and does not prejudice our experiments from being representative of the reality. In particular, we use three different values of k: 10, 20 and 30. The purpose of the experiments is showing what is the correlation value of the selected k measurements as the number n of total measurements performed increases. More in detail, we make use of the following evaluation metrics:

- **Minimum Correlation**: as defined in Section 4.2, for a correlation function Γ_p, it corresponds to $min(C^{\{p\}}, k)$, so the average pairwise correlation of the submatrix of $C^{\{p\}}$ relative to the k selected observations. Observations here are selected using the greedy heuristic presented in Section 4.2 in order to minimize $min(C^{\{p\}}, k)$.
- **Maximum Correlation**: the opposite of the minimum correlation, defined as $max(C^{\{p\}}, k)$ in Section 4.2 and used in the experiments as a term of comparison.
- **Minimum Clustering Correlation**: as defined in Section 4.3, for a correlation function Γ_p, it corresponds to $min\_clus(C^{\{p\}}, k)$, so the mean over the average pairwise correlations of the submatrices of $C^{\{p\}}$ relative to the clusters. Observations here are clustered using the algorithm presented in Section 4.3 in order to minimize $min\_clus(C^{\{p\}}, k)$.

[5] This is done because we are focusing here on the objective comparison of the datasets rather than tuning parameters for our simulation scenario.

- **Maximum Clustering Correlation**: the opposite of the minimum clustering correlation, defined as $max\_clus(C^{\{p\}}, k)$ in Section 4.3 and used in the experiments as a term of comparison.

It is also important what a potential attacker could understand when taking hold of the k selected observations. In such cases we would need to recompute the correlation matrix of such k elements according to the selected correlation function without using the other $n-k$ observations, instead of extracting a submatrix from $C^{\{p\}}$. This produces potentially a second version of the above mentioned evaluation criteria, which we will address as "biased" and represents the correlation of the k observations from an *attacker point of view*. Conversely, by extracting a submatrix without recomputing, we are using "oracle" evaluation criteria, which instead represent the correlation of the k observations from the point of view of the owner. We can easily demonstrate that these two coincide when the correlation function has the property of mutual independence, as correlation values only depend on the considered observations, not on the whole set. If not specified otherwise, the oracle criteria are used by default. On top of the described criteria, we set up two macro experiments set: one for the performance of the clustering strategy (Section 4.3) over the greedy selection (Section 4.2), the other for studying the behavior of both such strategies in detail when n increases over time. All results are averaged over the users.

6.2.1. *Clustering performance over greedy*

In this experiment, we demonstrate the claim of Section 4.3, in which we state that using the greedy selection over time might yield a near-optimal correlation, however, in the long run, many highly-correlated observations end up never being spent or, worse, being spent all together. In order to validate this, we initialize every participant with 90 measurements and we iteratively apply the greedy approach until the remaining number is less than k, every time recording the correlation of the k-sized subset spent. We then perform the same test by applying the clustering algorithm at the beginning and iteratively compute the average correlation of each cluster. We then calculate the highest of the recorded values for both methods (which corresponds to the "worst") and report it as a comparison, averaged over the participants. Results are shown in Section 7.1.

6.2.2. *Correlation over time*

In this experiment, we focus on discussing the evaluation criteria introduced above over time. More in detail, for each participant of each dataset, we systematically add one observation, starting from $n = k$ until $n = 90$. At each of such steps we perform both a greedy selection and a clustering selection, both minimizing and maximizing the correlation, in order to show their trend. The k selected observations are not removed from the pool of n as in the other experiment. Both the oracle and the biased criteria are used here for a fair comparison. Furthermore, we perform an analysis on the amount of information that can be reconstructed by the attacker using the correlation function Distance from a Point Of Interest as an example.

7. Results

7.1. *Clustering performance over greedy*

In order to demonstrate the superiority, in realistic scenarios, of a clustered selection of observations over a greedy one, we ran simulations (according to Section 6.2.1) in which participants already performed 90 measurements each (therefore $n = 90$). Participants store measurements locally and repeatedly send k measurements to the server selecting them either with the greedy strategy (Minimum Correlation) or by clustering them in advance and sending one cluster at a time (Minimum Clustering Correlation). Results are shown in Fig. 5. For each of the participants we only take into account the worst set of k measurements (*i.e.* yielding the highest correlation). The main idea is the following: by using a greedy strategy, a participant would select the set of k measurements yielding the lowest correlation, however, as we start to redeem them, the last k measurements remaining may probably be a bad choice. We, in fact, expect the correlation of the k-sized set to increase as we start to run out of measurements. Conversely, the clustering strategy lets us group the measurements in k-sized groups a priori so that they have a similar internal correlation and, therefore, the best set is not that different from the worst set. In the figure, we represent the worst greedy correlation in tones of black and the worst clustering correlation in tones of red (different alpha values correspond to different datasets); for display purposes, we set to 1.0 the worst greedy correlation and normalized the respective worst clustering correlation accordingly. All the results are averaged out over the participants and shown for all the three correlation functions described in Section 5. Looking at Fig. 5 we first observe that the worst-case Minimum Clustering Correlation is always lower than the worst-case Minimum Correlation, hence supporting our claim. Taking a closer look at the correlation function *Absolute Distance in Time*, the two FourSquare datasets display a relatively higher difference, which is in line with what displayed in Fig. 3(b), as the observations are more spread in time. Consequently, the opposite occurs for the Sparse and Dense datasets.

7.2. *Correlation over time*

Our second experiment aims to evaluate the correlation of the selected k-sized subset of observations as the total number of observations n grows over time, according to the description in Section 6.2.2. As n increases from k to 90, participants start to compute the best k ones to be spent and report their correlation at each added observation. This is computed both with the greedy and the clustering strategies, in the latter case the correlation reported is the average over the clusters. The experiment is performed separately for each correlation function to evaluate them independently.

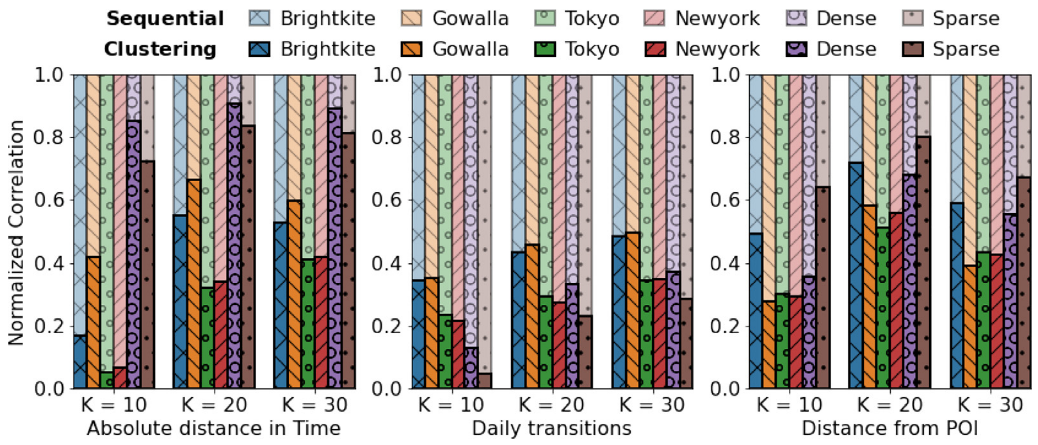


Fig. 5. Average correlation of the worst set of k values out of n measurements selected by the sequential greedy strategy (transparent colors) and the clustering strategy full colors, for each correlation function and different values of k. Results are normalized by setting the greedy value to 1.

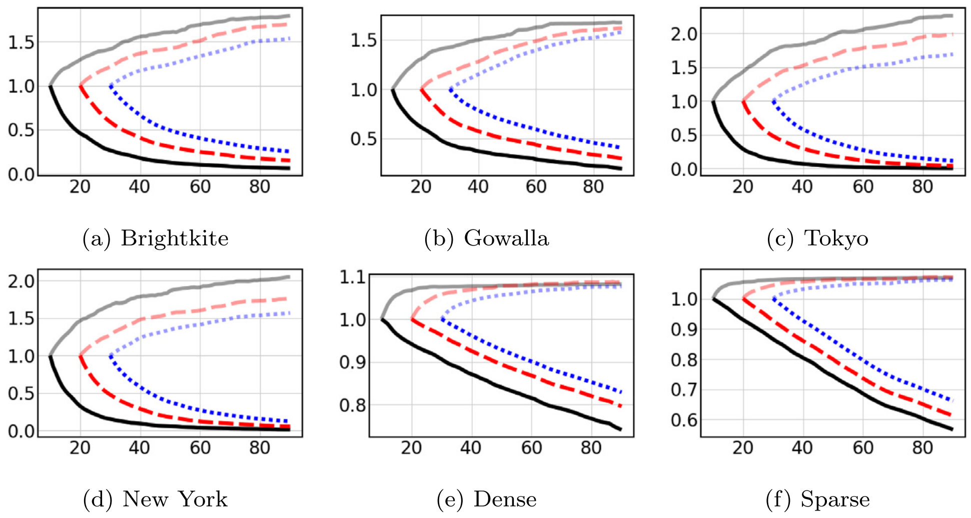


Fig. 6. Absolute Distance in time — Correlation values of the k selected measurements through the greedy strategy with N ranging from 0 to 90 for 100 users.

7.2.1. *Absolute Distance in Time*

The results for the correlation function Absolute Distance in Time are shown in Fig. 6 for the greedy version and Fig. 7 for the clustering version. Results are shown for all the six datasets by setting the value of k to 10 (black solid line), 20 (red dashed line) and 30 (blue dotted line). The value on the x-axis reports the total number of observations n, which increases over time, while on the y-axis we put the normalized correlation value. Each of the lines start when $n = k$, as it would not make sense to extract k measurements from a smaller set. Each of the lines is normalized by their starting value, as, for the purpose of this experiment, we are not so much interested in the absolute correlation value as much as their difference over time. In fact, each line is twofold: the lower one represents the minimum correlation, while the upper one represents, for a fair comparison, the maximum correlation. Both these lines are identical in color and style and their alpha value is set so that they are partially transparent, as they are easily distinguishable from each other. Furthermore, we introduced the minimum biased correlation to show potential differences. The latter is displayed with a fully opaque line for each value of k. For this correlation function we can easily see that the minimum correlation in its oracle version corresponds to the biased version as the correlation function supports mutual independence. In Fig. 6 we observe, for the greedy selection, that the minimum and the maximum correlation are mostly monotone – little perturbations are due to the greedy selection being itself an approximated method –, in particular, the more n increases, the more the maximum and the minimum correlation are dragged apart from each other showing the efficacy of the method. The clustering selection shown in Fig. 7 shows a similar trend, although the lines fluctuate. In detail we notice that a single fluctuation occurs at a constant pace, with the peak occurring when n is a multiple of k. This is easily explainable as when the line is off-peak it means that there is one cluster, smaller than all the others, in which the minimum internal correlation is

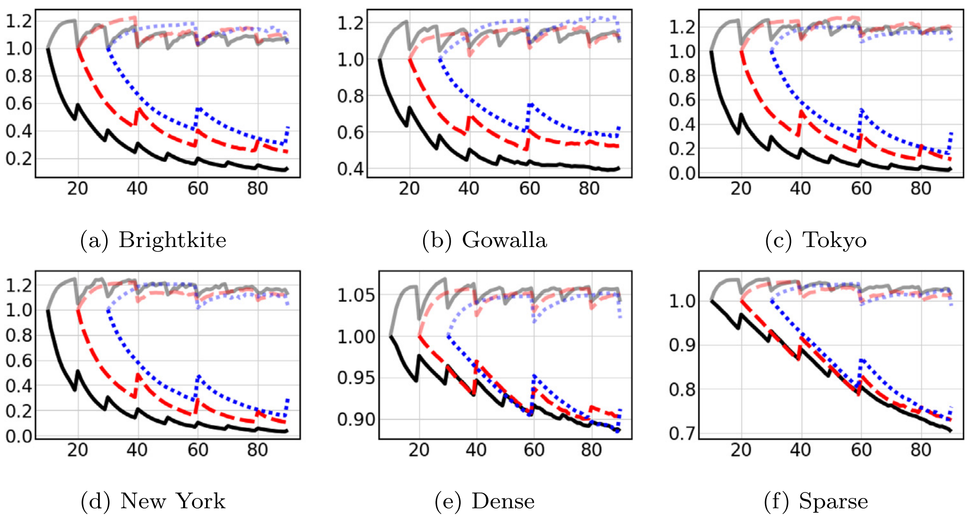


Fig. 7. Absolute Distance in Time — Cluster correlation values of clusters of size k with N ranging from 0 to 90 for 100 users.

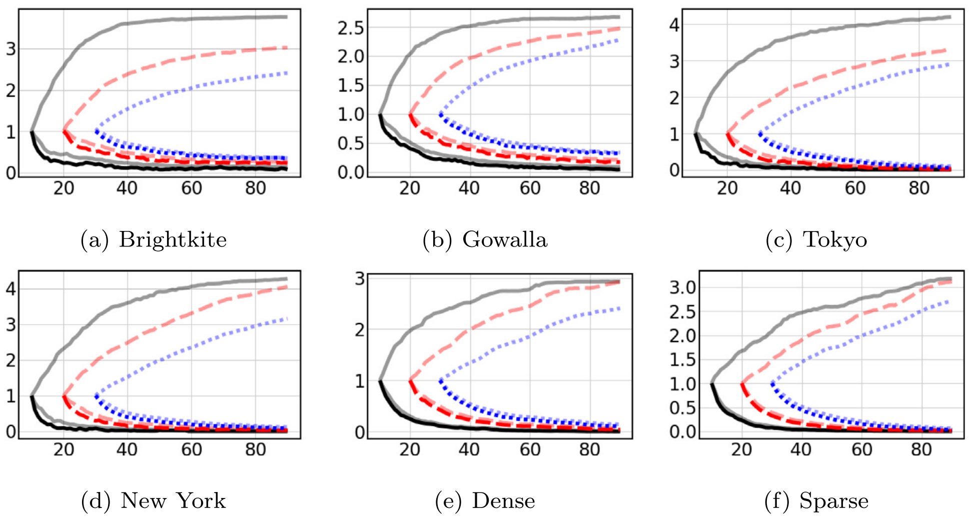


Fig. 8. Location and Time-Aware Daily Transitions — Correlation values of the k selected measurements through the greedy strategy with N ranging from 0 to 90 for 100 users.

likely to be very low, therefore causing the average correlation among the others to drop. The exact opposite potentially occurs when clustering for the maximum correlation. We also observe similar trends over the datasets, with the initial correlation being already high for Sparse and Dense. This is explained by the density of their observations in time, shown previously in Fig. 3(b).

7.2.2. Location and Time-Aware Daily Transitions

The results for the correlation function Location and Time-Aware Daily Transitions are shown in Fig. 8 for the greedy version and Fig. 9 for the clustering version, with the same methodology of Section 7.2.1. The first noticeable difference is that the oracle and the biased minimum correlations do not coincide, as the mutual independence property is not satisfied, resulting in an opaque line and a semi-transparent one very close to each other in the lowermost side of each plot. The differences are almost negligible, however we notice that the biased correlation is slightly lower than the oracle one, explained by the fact that the correlation between two observations can be potentially boosted by other unselected observations, but never reduced. In the biased version, the correlation is only calculated on the selected k-sized subset, while in the oracle version all other measurements are taken into account. Besides this difference, we assess that the correlation function efficiently fits our framework. It is also noticeable how results are similar over the different datasets.

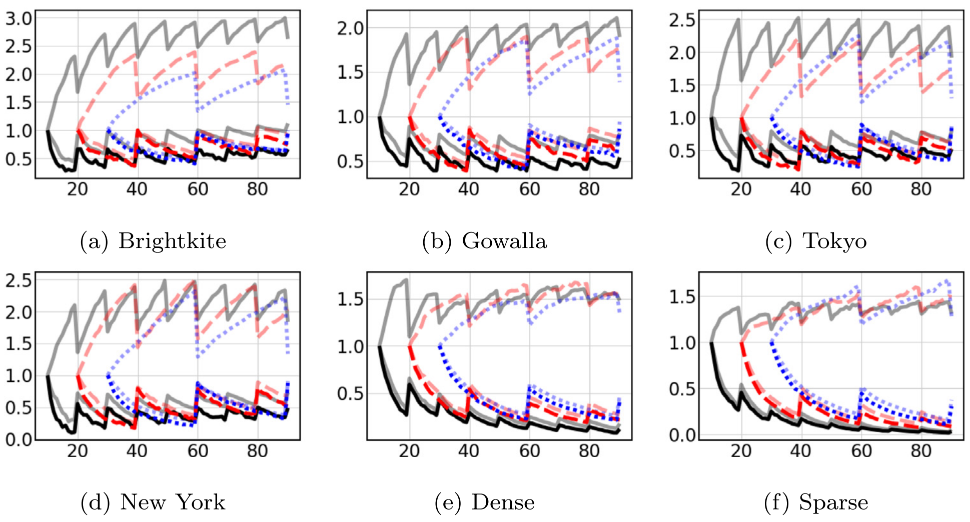


Fig. 9. Location and Time-Aware Daily Transitions — Cluster correlation values of clusters of size k with N ranging from 0 to 90 for 100 users.

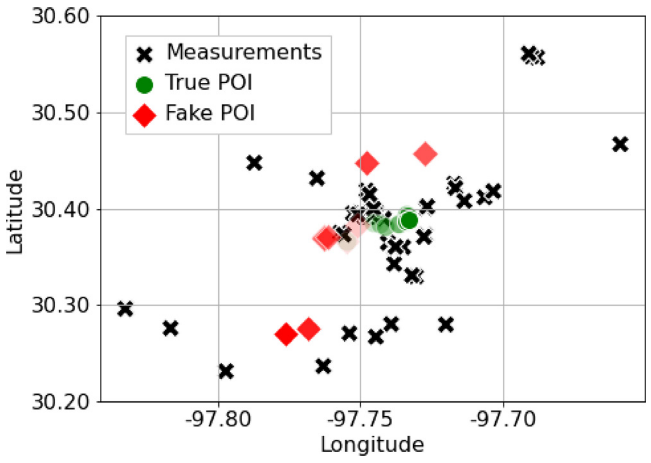


Fig. 10. Home example for Distance from a POI correlation function.

7.2.3. *Distance from a Point Of Interest*

We chose to show the results for the correlation function Distance from a Point of Interest in a different way from the other correlation functions. The rationale behind this choice is to show how a correlation function relates to a common technical privacy metric. This correlation function correlates two observation the more their middle point is close to the centroid of all the n observations stored. This means that an efficient selection of k elements would result in a subset with a centroid far from the actual one. Since we consider the centroid of the n measurements as the True POI, the farther the centroid of the k selected measurements (hereafter called the "Fake POI") is from the True POI, the less information is disclosed to a potential attacker. This is done since when the user sends different measurements with specific locations, if those are close in distance they may uncover a potential POI for such user, for instance her place, the work office or an hospital. Clearly, selecting measurements that are far from each other limits this problem, since the derived POI may be less meaningful. For a graphical feedback, we show in Fig. 10 an example of the Fake POI and the True POI for a randomly selected user in the Gowalla dataset and k set to 10. The True POI is shown in green whereas the Fake POI is shown in red. Both of them increase their opacity as n grows (*i.e.* as time passes); it is noticeable how the Fake POI results in being farther and farther from the True POI over time. As a comprehensive evaluation to this end, Fig. 11 displays, as n grows on the x-axis, the Vincenty distance between the Fake and the True POI averaged over all the traces in each of the datasets. The dotted lines identify the greedy version, whereas the solid lines identify the clustering version (here, the distance from the fake POI and the actual one is averaged out over the clusters). Also, black lines correspond to the best case,

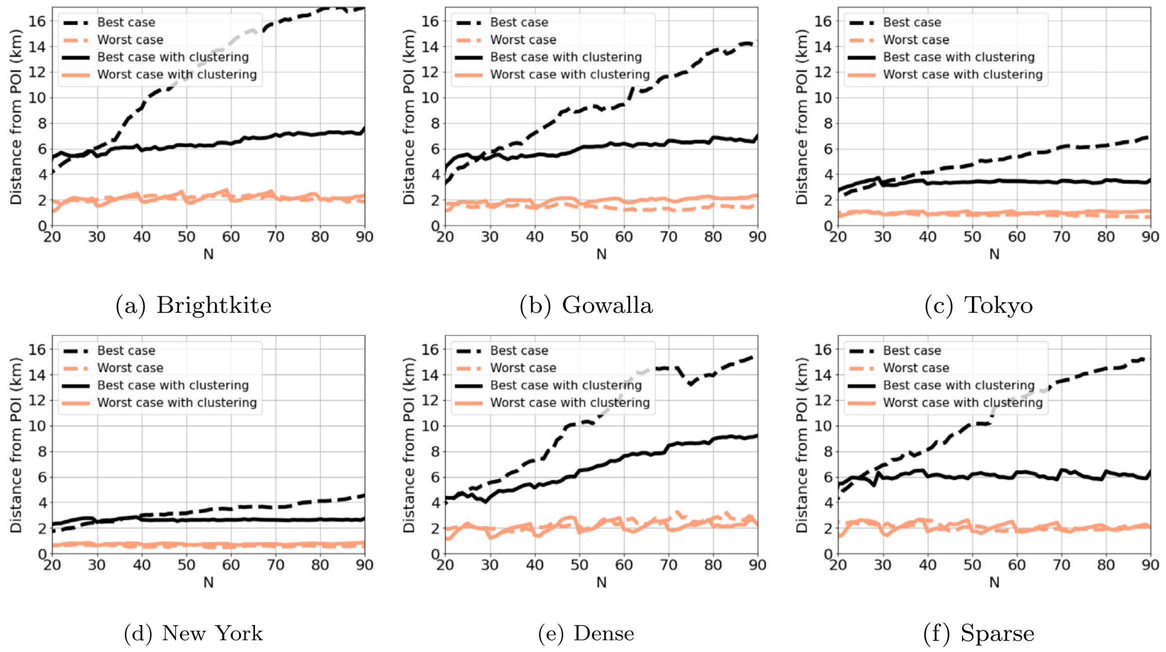


Fig. 11. Distance from a POI — Evaluation of the correlation function. Vincenty distance of the fake POI from the actual true POI with N ranging from 0 to 90 for 100 users.

i.e. using the minimum correlation, whereas light lines correspond to the worst case, *i.e.* corresponding to the maximum correlation. All distances are stated in kilometers and we constrained the value of k to 10 for displaying purposes, as other values reveal a similar behavior. We observe here that, even though the distance between the Fake POI and the True POI is not directly optimized by the selection, it still displays an advantageous behavior: as n grows, the distance in the best case grows too, both in the clustering and in the greedy version, although with a noticeable difference. This is important, as we are able to show an advantage even considering the k selected measurement in a pairwise fashion and not as a whole, which would require an exponential computational complexity that is impractical in our scenario.

7.3. Discussion

The results shown in this section reveal two key aspects of this framework. First, they validate the effectiveness and the mathematical properties of the pairwise correlation as a criterion for choosing a subset of information to be unveiled safely. Specifically Section 6.2.1 assesses the solidity of a clustering selection over time, much more practical in a real scenario, and Sections 7.2.1 and 7.2.2 display the soundness of the criterion itself as the number of total measurements increases. Secondly, the results show how it is possible to link a selection criterion based on correlation functions to a well-established technical privacy metric as a final output evaluation, specifically in Section 7.2.3. However, we do not target a specific privacy metric, as there are several of them [12] and each one is responsible for covering an aspect of the privacy that strictly depends on the use case, whereas our core contribution is in the framework and the information selection model. Specific implementations must focus on privacy metrics depending on the considered scenario.

Another aspect to discuss is the implications of the deployment of such a framework in a real scenario. Common questions that may arise are: How long are hashes stored in the personal devices? How often do users obtain rewards? What is the correlation value below which we consider the disclosure of k hashes to be safe? Real deployments adopting this framework can be different, so it can be for the correlation function chosen and, consequently, for setting the parameters. A straightforward strategy would be starting from the goal: *e.g.* for the Distance from a POI scenario we might set an acceptable distance from the fake POI to the true POI and check the respective correlation value experimentally. Furthermore, depending on the frequency of updates required by the use case (*i.e.* how many measurements a participant should provide per time unit), then other parameters can be set consequentially: for instance, if a certain number of measurements is required per hour, then we may expect a user to store a definite amount of hashes per day. By knowing approximately this number and comparing it with experimental evaluations similar to the ones in Section 7.2.3, we can estimate the conditions under which our privacy concern is fulfilled for different values of k, therefore the average number of rewards obtained by a user over a day can be estimated as well. On top of this, the actual value of a single reward should be established, however this is out of the scope of this study.

8. Conclusions

In this paper we have proposed a privacy preserving framework for opportunistic and pull-based MCS scenarios. In our proposal the central entity, or the crowdsourcer, has no information on the users at the data collection stage, while such information is necessary in the rewarding phase, when the users need to collect a prize for their contributions. Our framework is built on top of a set of correlation functions, all of them defined by a pairwise correlation formula between each couple of measurements. Subsequently, our framework is able to select, either greedily or through a clustering heuristic, the best k measurements out of a total of N – where "best" is equal to "less correlated" – in order to give as little (or as misleading) information as possible to a potential attacker able to access such k data points. This is built upon the assumption that k is the number of measurements needed in order to obtain a prize. We provided simulations that assess the effectiveness of our approach against the correlation functions we have defined, recall that other correlation functions can be added, provided they adhere to the set of properties specified in Section 4.1. We consider this as a pioneering effort towards privacy and anonymity preservation in such scenarios, in fact the majority of works in the same field tend to target task-based MCS scenarios. However, continuous sensing and pull-based scenarios are gaining more and more interest worldwide and are being increasingly used, such as in Mobile Contact Tracing, which has been recently leveraged and for which we proposed a similar mechanism [34]. We envision our approach to be adaptable to practical scenarios that can solve problems that matter.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

References

[1] F. Montori, L. Bedogni, L. Bononi, A collaborative internet of things architecture for smart cities and environmental monitoring, IEEE Internet Things J. 5 (2) (2018) 592–605.
[2] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, P. Bouvry, A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities, IEEE Commun. Surv. Tutor. 21 (3) (2019) 2419–2465.
[3] F. Restuccia, N. Ghosh, S. Bhattacharjee, S.K. Das, T. Melodia, Quality of information in mobile crowdsensing: Survey and research challenges, ACM Trans. Sensor Netw. 13 (4) (2017).
[4] W. Gong, B. Zhang, C. Li, Task assignment in mobile crowdsensing: Present and future directions, IEEE Netw. 32 (4) (2018) 100–107.
[5] F. Montori, P.P. Jayaraman, A. Yavari, A. Hassani, D. Georgakopoulos, The Curse of Sensing: Survey of techniques and challenges to cope with sparse and dense data in mobile crowd sensing for Internet of Things, Pervasive Mob. Comput. 49 (2018) 111–125.
[6] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, X. Mao, Incentives for mobile crowd sensing: A survey, IEEE Commun. Surv. Tutor. 18 (1) (2016) 54–67.
[7] L. Bedogni, M. Levorato, Rising user privacy against predictive context awareness through adversarial information injection, in: 2018 IEEE Global Communications Conference, GLOBECOM, IEEE, 2018.
[8] L. Sweeney, k-anonymity: A model for protecting privacy, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10 (05) (2002) 557–570.
[9] C. Dwork, Differential privacy: A survey of results, in: International Conference on Theory and Applications of Models of Computation, Springer, 2008, pp. 1–19.
[10] H. Chen, B. Guo, Z. Yu, Q. Han, Crowdtracking: Real-time vehicle tracking through mobile crowdsensing, IEEE Internet Things J. 6 (5) (2019) 7570–7583.
[11] F. Montori, L. Bedogni, A privacy preserving framework for rewarding users in opportunistic mobile crowdsensing, in: Proceedings of the 7th International Workshop on Crowd Assisted Sensing, Pervasive Systems and Communications, CASPer 2020, IEEE, 2020.
[12] I. Wagner, D. Eckhoff, Technical privacy metrics: a systematic survey, ACM Comput. Surv. 51 (3) (2018) 1–38.
[13] M.A. Alsheikh, Y. Jiao, D. Niyato, P. Wang, D. Leong, Z. Han, The accuracy-privacy trade-off of mobile crowdsensing, IEEE Commun. Mag. 55 (6) (2017) 132–139.
[14] N. Li, T. Li, S. Venkatasubramanian, t-closeness: Privacy beyond k-anonymity and l-diversity, in: 2007 IEEE 23rd International Conference on Data Engineering, 2007, pp. 106–115, http://dx.doi.org/10.1109/ICDE.2007.367856.
[15] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, L-diversity: Privacy beyond k-anonymity, ACM Trans. Knowl. Discov. Data 1 (1) (2007) http://dx.doi.org/10.1145/1217299.1217302, 3–es.
[16] L. Wang, D. Zhang, D. Yang, B.Y. Lim, X. Ma, Differential location privacy for sparse mobile crowdsensing, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, IEEE, 2016, pp. 1257–1262.
[17] J. Ni, K. Zhang, Q. Xia, X. Lin, X.S. Shen, Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing, IEEE Trans. Mob. Comput. 19 (6) (2019) 1317–1331.
[18] L. Yang, M. Zhang, S. He, M. Li, J. Zhang, Crowd-empowered privacy-preserving data aggregation for mobile crowdsensing, in: Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, ACM, 2018, pp. 151–160.
[19] T. Peng, Q. Liu, D. Meng, G. Wang, Collaborative trajectory privacy preserving scheme in location-based services, Inform. Sci. 387 (2017) 165–179.
[20] W. Jin, M. Xiao, M. Li, L. Guo, If you do not care about it, sell it: Trading location privacy in mobile crowd sensing, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019.
[21] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, X. Ma, Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation, in: Proceedings of the 26th International Conference on World Wide Web, ACM, 2017, pp. 627–636.
[22] I.J. Vergara-Laurens, L.G. Jaimes, M.A. Labrador, Privacy-preserving mechanisms for crowdsensing: Survey and research challenges, IEEE Internet Things J. 4 (4) (2017) 855–869, http://dx.doi.org/10.1109/JIOT.2016.2594205.

[23] L. Pournajaf, D.A. Garcia-Ulloa, L. Xiong, V. Sunderam, Participant privacy in mobile crowd sensing task management: A survey of methods and challenges, SIGMOD Rec. 44 (4) (2016) 23–34, http://dx.doi.org/10.1145/2935694.2935700.
[24] L. Bedogni, G. Cabri, Identification of Social Aspects by Means of Inertial Sensor Data, Information 11 (11) (2020) 534.
[25] H. Gu, J. Zhang, T. Liu, M. Hu, J. Zhou, T. Wei, M. Chen, DIAVA: A traffic-based framework for detection of SQL injection attacks and vulnerability analysis of leaked data, IEEE Trans. Reliab. 69 (1) (2020) 188–202, http://dx.doi.org/10.1109/TR.2019.2925415.
[26] Q. Ma, S. Zhang, T. Zhu, K. Liu, L. Zhang, W. He, Y. Liu, PLP: Protecting location privacy against correlation analyze attack in crowdsensing, IEEE Trans. Mob. Comput. 16 (9) (2017) 2588–2598, http://dx.doi.org/10.1109/TMC.2016.2624732.
[27] I. Khokhlov, L. Reznik, S. Chuprov, Framework for integral data quality and security evaluation in smartphones, IEEE Syst. J. 15 (2) (2021) 2058–2065, http://dx.doi.org/10.1109/JSYST.2020.2985343.
[28] A. Civril, M. Magdon-Ismail, Exponential inapproximability of selecting a maximum volume sub-matrix, Algorithmica 65 (1) (2013).
[29] T.F. Gonzalez, Clustering to minimize the maximum intercluster distance, Theoret. Comput. Sci. 38 (1985) 293–306.
[30] F.-J. Wu, T. Luo, CrowdPrivacy: Publish More Useful Data with Less Privacy Exposure in Crowdsourced Location-Based Services, ACM Trans. Priv. Secur. 23 (1) (2020) 1–25.
[31] J. Mahmud, J. Nichols, C. Drews, Home location identification of twitter users, ACM Trans. Intell. Syst. Technol. 5 (3) (2014) 1–21.
[32] E. Cho, S.A. Myers, J. Leskovec, Friendship and mobility: user movement in location-based social networks, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 1082–1090.
[33] D. Yang, D. Zhang, V.W. Zheng, Z. Yu, Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs, IEEE Trans. Syst. Man Cybern. Syst. 45 (1) (2014).
[34] L. Bedogni, F. Montori, F.D. Salim, Location contact tracing: Penetration, privacy, position and performance, Digit. Gov. Res. Pract. (2022).