

On the Efficiency of Privacy-Preserving Path Hiding for Mobile Sensing Applications

Delphine Christin*, Andreas Reinhardt†, Matthias Hollick*,

* Secure Mobile Networking Lab, Technische Universität Darmstadt, Mornewegstr. 32, 64293 Darmstadt, Germany

† School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia

*{delphine.christin, matthias.hollick}@seemoo.tu-darmstadt.de †andreasr@cse.unsw.edu.au

Abstract—Current mobile sensing applications typically annotate the collected sensor readings with spatiotemporal information before reporting them to a central server. Such information can however endanger the users’ privacy, as it reveals insights about their daily routines. Users must therefore trust the application administrators not to misuse the reported information. To diminish user dependence on administrators trustworthiness, we propose a privacy-preserving collaborative scheme, in which users exchange the collected sensor readings at opportunistic encounters. We model malicious administrators attempting to identify exchanged sensor readings based on spatial disparity by applying four state-of-the-art outlier detection algorithms. We thoroughly investigate the influence of different exchange patterns and the parameters of the algorithms on their performance based on a real-world dataset. The results for location traces of 20 users gathered during 14 days show that our algorithm achieves a high level of privacy protection.

Index Terms—Mobile applications, security and privacy protection.

I. INTRODUCTION

Mobile sensing applications benefit from the ubiquity of mobile phones and their integrated multimodal sensors to gather information about the users and their environment in unprecedented quality and quantity. For example, the embedded microphones can be leveraged to measure noise pollution [1], while accelerometers may serve to monitor road conditions [2]. Commonly, the gathered sensor readings are annotated together with the time and location of their collection and reported to the application server. This puts the user’s privacy at stake because application administrators have direct access to the reported data. The users’ daily motion paths can be easily tracked and information about their homes and workplaces extracted [3]. Although there exist some mechanisms to protect user’s privacy, they often rely on trusted third parties. For example, spatial cloaking builds groups of users sharing a common attribute (e.g., k users located in the same district) to render them indistinguishable from each other [4]. However, users need to report their exact locations to this third party in order to compute this common attribute. Users privacy protection hence primarily depends on the trustworthiness of application administrators as well as involved third parties.

In order to reduce this dependency, we proposed in [5] a concept called path jumbling consisting of a collaborative privacy-preserving mechanism. By using our scheme, users

can preserve their privacy in a decentralized fashion by exchanging their annotated sensor readings with those of other users at opportunistic physical encounters. These exchanges allow effectively swap users paths; the prior path of one participant becomes that of another participant and vice versa, thus breaking the link between users identities and collected sensor readings. In our previous work, we analyzed the design space and confirmed the viability of our scheme by means of a thorough performance analysis, yet we did not quantify the achievable degree of privacy protection in realistic settings.

In this paper, we therefore analyze the path jumbling concept by determining its degree of privacy protection. To do so, we assess the capability of curious administrators in inferring the exchanged sensor readings from their spatial distribution based on three so-called *exchange strategies*. These strategies define how the sensor readings to be exchanged are selected. The contributions of this paper are summarized as follows.

- 1) We apply the path jumbling concept in a participatory sensing application based on a realistic dataset. For the path jumbling scheme, we consider three different exchange strategies differing in terms of number and sequence of exchanged sensor readings.
- 2) We model the behavior of malicious application administrators, who attempt to distinguish exchanged sensor readings from original ones based on their spatial distribution. To this end, we implement four different state-of-the-art outlier detection algorithms and tailor them to the specificities of our model.
- 3) We thoroughly investigate the performance of the selected algorithms in identifying exchanged sensor readings by means of extensive simulations. We especially examine the impact of the applied exchange strategy and the parametrization of the algorithms on their overall performance.

The remainder of this paper is structured as follows. We present our assumptions and threat model in Section II. We briefly revisit our concept of path jumbling in Section III and present our problem statement in Section IV. Section V is devoted to our evaluation settings and results. After summarizing existing work in Section VI, we make concluding remarks in Section VII.

II. ASSUMPTIONS AND THREAT MODEL

A. System Model

For our system, we assume participatory sensing applications without real-time constraints for data delivery. Examples include monitoring noise pollution [1] and road conditions [2]. In these applications, the participants carry mobile phones equipped with embedded sensors, wireless interfaces and positioning systems. The mobile phones autonomously collect sensor readings (e.g., sound samples and accelerometer data). Each sensor reading is stamped with the collection time and location information to form the following triplet $T = \langle t, l, s \rangle$ where t is time, l is location, and s is the corresponding sensor reading. Sensor readings can be either vectors or scalar values. Additional processing may be locally applied on the sensor readings for feature extraction and/or prevent sensitive information from being disclosed. For example, possibilities include extracting the noise level from the collected sound samples in applications monitoring noise pollution.

The triplets are then autonomously reported to the application server. The application server is able to establish a link between the reported triplets and the participants who reported them. The establishment of this link can be based on either explicit identifiers, such as user ID and pseudonyms, or the analysis of reporting metadata to, e.g., infer the location from the used IP addresses. Ultimately, the application can analyze the reported triplets to provide statistics or build summary maps (e.g., illustrating the noise level or the road conditions across the city) accessible to the public.

B. Adversary Model

Malicious application administrators are a common threat to the privacy of the users in participatory sensing applications, as they have direct access to the triplets reported by the users. In absence of privacy-preserving mechanisms locally applied on the mobile phones, the triplets contain information about the locations visited by the users and may fully disclose the followed paths. Given that the collected sensor readings are reported and stored on a central application server, they might be exposed to different privacy threats. For example, malicious administrators may attempt to gain further information about the participants by analyzing their sensor readings and/or intentionally disclose or misuse the sensor readings and derived information to untrusted parties. Additionally, external attacks can be directed at the server to fraudulently access the stored data.

In what follows, we assume an honest-but-curious adversary model, in which administrators attempt to passively breach the privacy of the participants, but run the system normally and faithfully. This means that the application administrators focus on the data reported by the participants to the application server. They, however, do not launch active attacks to obtain further information.

Users can also become adversaries as an artifact of the collaborative nature of the path hiding mechanism. For example, malicious users can drop exchanged triplets or create

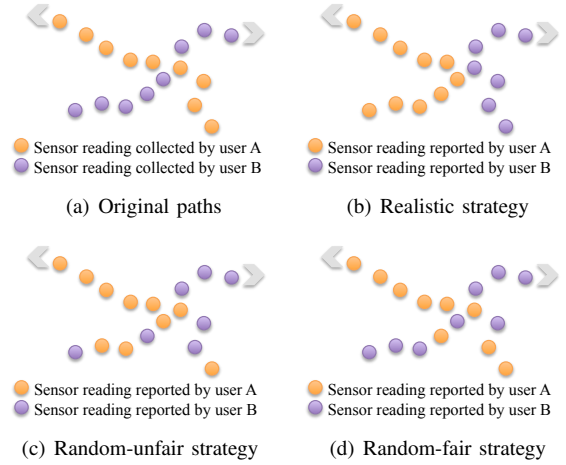


Fig. 1. Examples of paths jumbled using different exchange strategies

false triplets to be exchanged in order to alter the results consolidated at the server side and perturb the function of the collaborative path hiding mechanism. We have addressed this particular adversary scenario in [6] and do not specifically regard the impact of false contributions in this paper.

III. THE PATH JUMBLING CONCEPT

The objective of this concept we introduced in [5] is to break the link between the spatiotemporal context (i.e., time and location) at which the sensor readings were taken and the identity of the participants (i.e., mobile devices) in order to protect their privacy. The spatiotemporal context reveals the visited locations and paths followed by the participants during the sensing process, thus providing insights about them. In our decentralized approach, the participants collaborate to preserve their privacy. Since the triplets contain spatiotemporal information, those triplets collected on user devices between the participants in physical proximity are jumbled and thus unlinked from their original collectors. In the following, we investigate the influence of the following exchange strategies on participants' privacy protection. These exchange strategies have been chosen based on their diversity in terms of number of triplets involved and continuity. Figure 1 illustrates examples of paths jumbled using the following exchange strategies.

- *Realistic strategy*: Users exchange the entire set of collected and/or exchanged triplets with a certain probability at encounters. This strategy exchanges consecutive triplets, which hence form realistic path segments as illustrated in Fig. 1(b).
- *Random-unfair strategy*: Each user independently and randomly determines the number of triplets to exchange. Statistically, users exchange fewer triplets than with the realistic strategy. Consequently, this requires a lower degree of trust in other users, but may result in discontinuous paths as shown in Fig. 1(c).
- *Random-fair strategy*: Users agree on a common number of triplets to exchange at their meetings. Using this strategy, the participants fairly share the reporting overhead,

since they upload the same number of triplets as collected to the application server. The jumbling degree is however statistically lower than with the other strategies, because it is based on the minimum of both parties' offers.

As a result, each user uploads a combination of the samples received from encountered users and their own gathered samples to the application server once every hour. Readings collected since the last physical encounter are never uploaded to the server directly, but always buffered locally until the next exchange in order to maintain each user's privacy.

IV. PROBLEM STATEMENT

Our path jumbling mechanism breaks the association between user identities and the collected sensor readings. As a result, the reported sensor readings may not have been collected by the users reporting them; the users may even have visited entirely different locations at the time of the data collection. Still, curious administrators can try to identify whether the data transmitted to the application server has been exchanged or locally collected by the participant based on the spatiotemporal distribution of the sensor readings. This identification of exchanged sensor readings might enable an attacker to reconstruct the original paths and thus eventually to identify the individual user trajectories. We specifically regard:

- *Velocity disparities*: Exchanged sensor readings can be identified due to differences in velocity as illustrated in Fig. 2. Those sensor readings gathered while moving at a particular velocity are likely to belong to the same user. However, administrators cannot accurately link sensor readings to the corresponding users identity. Velocity disparities can especially be observed when users exchange large sets of consecutive triplets or apply the realistic exchange strategy, since a mean velocity can easily be computed.
- *Spatial outliers*: Exchanged sensor readings can be distinguished from collected ones, if they have been collected in different geographical areas as depicted in Fig. 3. A sensor reading collected in the vicinity of a group of sensor readings reported by another user is likely to belong to the same user. Again, user's identity cannot be inferred by curious administrators. Since users exchange all sensor readings at each encounter when using the realistic exchange strategy, this case can only happen when users apply the random-unfair or random-fair strategies.
- *Unreachable velocities*: The spatiotemporal distribution of the collected and exchanged triplets can reveal impossible velocities between triplets appearing as consecutive. This is particularly the case when users exchange sparse triplets collected in remote locations and at short time intervals. This effect is therefore more likely to happen when either the random-fair or random-unfair strategies are applied.

We thus examine to which extent curious application administrators can leverage these issues to distinguish exchanged sensor readings from the original ones and hence, measure the privacy protection provided by our scheme.

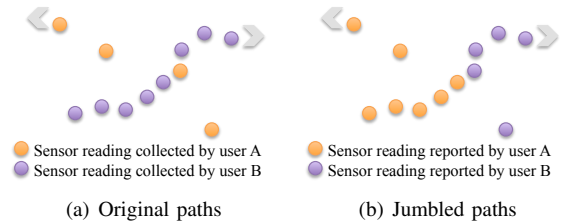


Fig. 2. Example of paths jumbled showing different velocities

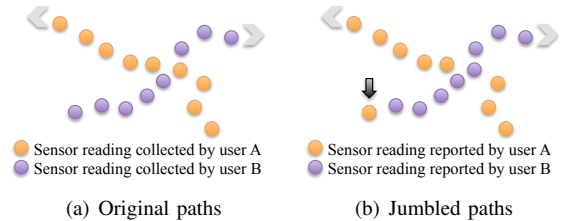


Fig. 3. Example of paths showing a spatial outlier

V. EVALUATION

In our evaluation, we model malicious application administrators who aim at distinguishing jumbled triplets from the original ones based on their spatiotemporal information. To this end, we apply four different state-of-the-art outlier detection algorithms on a realistic dataset in order to evaluate administrators performance in identifying jumbled triplets and thus quantify the privacy protection provided by our approach. We first present our simulation setup and method. In particular, we describe the main principles of the selected outlier detection algorithms and highlight their key parameters. Next, we describe our results and specially discuss how these algorithms behave depending on the evaluated parameters.

A. Simulation Setup and Method

Our evaluation is based on the GPS traces from the GeoLife project [7]. In this real-world deployment, the users carried GPS-enabled devices to monitor their location. We extend the initial scenario to a participatory sensing application by assuming that a triplet was collected at each monitored location. From this dataset, we selected a period of 14 days and 20 users having at least met one other user during this period. Figure 4 depicts the daily number of meetings during this period with around 200 triplets per paths on average.

Next, we apply one of the following exchange strategies on the triplets collected by the users during the chosen period: (1) realistic with an exchange probability of 0.5, (2) random-unfair, and (3) random-fair (see Section III). By doing so, we artificially generate jumbled paths. These paths are similar to those users would have obtained if they would have actually exchanged these triplets at physical encounters.

1) *Detection of Jumbled Triplets*: In order to analyze the efficacy of the path hiding approach, we assume that malicious application administrators apply one of the following outlier detection algorithms tailored to identify spatial outliers at the end of the considered time period.

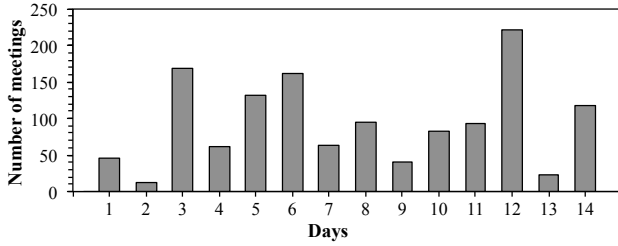


Fig. 4. Distribution of daily number of meetings

a) *Velocity-based outlier detection algorithm*: With this algorithm, we aim at identifying consecutive triplets that are inconsistent with a realistic movement model. In other words, our objective is to determine if the distance between two triplets can be covered within the period of time defined by their timestamps, i.e., if the user's velocity is realistic. To this end, we define different models based on a maximum velocity v_{max} that cover most users' transportation modalities.

The algorithm computes the velocity v_i between consecutive triplets T_i and T_{i+1} . If the computed velocity is higher than v_{max} , the triplet is considered as an outlier since its velocity mismatches with the model.

b) *Grubb-based Outlier Detection Algorithm*: Inspired from Grubb's test [8], this algorithm shown in Algorithm 1 is based on the spatial distance between neighboring triplets. In particular, it compares the Euclidian distance of each triplet T with the mean distance to its neighbors (see steps 3 to 7). Hence, it differs from the above algorithm in the metric used and the number of considered triplets. In fact, it uses the distance (instead of the velocity) and considers both direct neighbors of each triplet as well as the neighbors of its direct neighbors (instead of a unique neighbor like in the above algorithm). The computed spatial distance is then compared to the mean and the standard deviation obtained for all triplets by computing Z_s (see step 12). If Z_s is greater than the given confidence level c , the triplet is considered as outlier (see steps 13 and 14). The algorithm identifies an outlier per iteration and is hence iteratively applied when multiple outliers are assumed. As a result, the algorithm can be parametrized using the following parameters: (1) confidence level c and (2) number of algorithm iterations n .

c) *Distance-based Outlier Detection Algorithm*: This algorithm is based on the approach proposed in [9]. In this approach, a triplet is regarded to be a $DB(p, d_{min})$ -outlier if at least a fraction p of the triplets show a distance greater than the distance d_{min} from the considered triplet [9]. Let T_i be the triplet of interest. The algorithm, shown in Algorithm 2, first computes the Euclidian distance between T_i and each triplet on the same path (see step 3 in Algorithm 2). If the computed distance is lower than d_{min} , the triplet is added to the neighborhood of T_i (see step 5). This means that the neighborhood of T_i contains the set of triplets that are within distance d_{min} to T_i . According to [9], p determines the minimum fraction of triplets that must be outside the

Algorithm 1 Grubb-based outlier detection algorithm

Require: n : number of iterations, c : confidence level, N : total number of triplets

- 1: **for** each iteration i with $i \in [1, n]$ **do**
- 2: **for** all triplets T_j with $j \in [2, N]$ **do**
- 3: compute mean distance between T_{j-1} and its direct neighbors: $\overline{d_{T_{j-1}}} = \frac{|T_{j-2}T_{j-1}| + |T_{j-1}T_j|}{2}$
- 4: compute mean distance between T_j and its direct neighbors: $\overline{d_{T_j}} = \frac{|T_{j-1}T_j| + |T_jT_{j+1}|}{2}$
- 5: compute mean distance between T_{j+1} and its direct neighbors: $\overline{d_{T_{j+1}}} = \frac{|T_jT_{j+1}| + |T_{j+1}T_{j+2}|}{2}$
- 6: compute mean distance of the direct neighbors of T_j : $\overline{d_{T_{j-1}T_{j+1}}} = \frac{\overline{d_{T_{j-1}}} + \overline{d_{T_{j+1}}}}{2}$
- 7: compute distance difference between T_j and its direct neighbors: $s_{T_j} = |\overline{d_{T_{j-1}T_{j+1}}} - \overline{d_{T_j}}|$
- 8: **end for**
- 9: $\overline{s_T} :=$ mean value of all s_{T_j} with $j \in [2, N]$
- 10: $\overline{\sigma_T} :=$ standard deviation of all s_{T_j} with $j \in [2, N]$
- 11: **for** all triplets T_j with $j \in [2, N]$ **do**
- 12: compute $Z_s(T_j) = \frac{s(T_j) - \overline{s_T}}{\overline{\sigma_T}}$
- 13: **if** $Z_s(T_j) > c$ **then**
- 14: T_j is categorized as outlier
- 15: **break**
- 16: **end if**
- 17: **end for**
- 18: **end for**

Algorithm 2 Distance-based outlier detection algorithm

Require: p : fraction of triplets, d_{min} : distance

- 1: **for** all triplet T_i **do**
- 2: **for** all triplet T_j with $T_j \neq T_i$ **do**
- 3: compute $d_{T_iT_j} = |T_iT_j|$
- 4: **if** $d_{T_iT_j} \leq d_{min}$ **then**
- 5: add T_j to the neighborhood of T_i
- 6: **if** neighborhood size of $T_i > M$ **then**
- 7: T_i is categorized as non-outlier
- 8: **break**
- 9: **end if**
- 10: **end if**
- 11: **end for**
- 12: **end for**

neighborhood of an outlier. Consequently, the maximum number of triplets in the neighborhood of an outlier is equal to $M = N(1 - p)$, N being the total number of triplets of the path. This implies that if there are at least $M + 1$ triplets included in the neighborhood of T_i , T_i is identified as a non-outlier and another triplet is examined (see steps 6 to 8). As compared to the previous algorithms, the distance-based outlier detection algorithm hence does not only compare the distance of each triplet to its direct neighbors (i.e., its successor and its predecessor), but compares the distance to any triplet until the newly defined neighborhood contains more than $M + 1$ triplets. As a result, the algorithm can be parametrized according to the following values: (1) fraction of triplets p and (2) distance d_{min} .

d) *Density-based Outlier Detection Algorithm*: The last algorithm is adapted from [10]. It is based on the computation of a *Local Outlier Factor* (LOF) that quantifies the likeliness

of a triplet to have been jumbled. Again, this method relies on the distance between triplets and determines the density of triplets around the triplet of interest. The density is defined as the number of nearest neighbors within a certain distance around a triplet. It hence measures how isolated a triplet is from potential neighbors. The rationale behind this algorithm is that outliers show lower density than non-outlier triplets. Algorithm 3 summarizes the different steps necessary to identify potential outliers. In particular, it requires the computation of the following variables introduced in [10] and adopted herein.

- The k -distance of a triplet T_i ($k \in \mathbb{N}$) is defined in Algorithm 3 as the distance $d_{T_i T_j}$ between T_i and a triplet T_j of the same path such that:

- 1) for at least k triplets T_k with $T_k \neq T_i$,

$$d_{T_i T_k} \leq d_{T_i T_j} \quad (1)$$

- 2) for at most $k - 1$ triplets T_k ,

$$d_{T_i T_k} < d_{T_i T_j} \quad (2)$$

- The k -distance neighborhood of a triplet T_i denoted $N_k(T_i)$ contains every triplet whose distance from T_i is not greater than the k -distance of T_i . The triplets contained in the k -distance neighborhood of T_i are referred to as the k -nearest neighbors of T_i .
- The reachability distance of triplet T_i with respect to triplet T_j is defined as:

$$rd_k(T_i, T_j) = \max\{k - \text{distance}(T_j), d_{T_i T_j}\} \quad (3)$$

- The local reachability density of triplet T_i is defined as:

$$lrd_k(T_i) = \frac{|N_k(T_i)|}{\sum_{T_j \in N_k(T_i)} rd_k(T_i, T_j)} \quad (4)$$

- The local outlier factor of triplet T_i is defined as:

$$LOF_k(T_i) = \frac{\sum_{T_j \in N_k(T_i)} \frac{lrd_k(T_j)}{lrd_k(T_i)}}{|N_k(T_i)|} \quad (5)$$

As a result, the clustering of the triplets is based on both the minimum number of triplets k and the reachability distance rd_k [10]. Both metrics define a density threshold denoted t_d . Triplets are considered as outliers when their LOF is below t_d , while they are considered as non-outliers for higher LOF values (see step 9 in Algorithm 3). In our evaluation, we therefore investigate the impact of the following parameters on the performance of malicious administrators in identifying jumbled triplets: (1) minimum number of neighborhood triplets k and (2) density threshold t_d .

B. Evaluation Metric

In the next step, we apply each of the presented outlier detection algorithms on the generated jumbled paths and vary their respective parameters. As a measure of the algorithm performance in identifying exchanged triplets, we use the *Matthews Correlation Coefficient* (MCC) computed as follows. *True positives* (TP) represents the number of exchanged triplets identified as exchanged ones. *True negatives* (TN) represents the number of original triplets identified as such.

Algorithm 3 Density-based outlier detection algorithm







Require: k : minimum number of triplets, t_d : density threshold

- 1: **for all** triplets T_i **do**
- 2: compute k -distance for T_i based on Eq. 1 and 2
- 3: **end for**
- 4: **for all** triplets T_i **do**
- 5: determine the k -distance neighborhood of T_i
- 6: compute rd for T_i and all other triplets according to Eq. 3
- 7: compute lrd for T_i according to Eq. 4
- 8: compute LOF for T_i according to Eq. 5
- 9: **if** $LOF \leq t_d$ **then**
- 10: T_i is categorized as outlier
- 11: **end if**
- 12: **end for**

TABLE I
EXAMPLES OF MCC VALUES FOR DIFFERENT TP AND TN VALUES (WITH $TP = 1 - FP$ AND $TN = 1 - FN$)

		TP				
		0.05	0.25	0.50	0.75	0.95
TN	0.05	-0.90	-0.71	-0.50	-0.28	0.00
	0.25	-0.71	-0.50	-0.25	0.00	0.28
	0.50	-0.50	-0.25	0.00	0.26	0.50
	0.75	-0.28	0.00	0.26	0.50	0.71
	0.95	0.00	0.28	0.50	0.71	0.90

TABLE II
CHOSEN CLASSIFICATION OF THE PERFORMANCE OF THE ALGORITHMS BASED ON THEIR MCC VALUES AND ASSOCIATED COLOR SCHEME

MCC value	Protection performance	Color
$MCC \leq 0.00$	Poor	
$0.00 < MCC < 0.30$	Fair	
$0.30 \leq MCC < 0.60$	Good	
$0.60 \leq MCC < 0.90$	Very good	
$0.90 \leq MCC < 1.00$	Excellent	
$MCC = 1.00$	Perfect	

False positives (FP) refer to the number of original triplets identified as exchanged, and *False negatives* (FN) to the number of exchanged triplets identified as original.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (6)$$

Table I illustrates examples of MCC values obtained for different values of TP and TN. The MCC values range between -1 and +1. A value of +1 indicates a perfect identification of both exchanged and original triplets, while a value of -1 indicates a total failure in identifying both categories. In what follows, the performance of the algorithms is evaluated by means of the computed MCC values and is categorized according to the classification presented in Table II. We have repeated each simulation ten times and present the averaged results.

C. Results

We discuss the results presented in Table III and obtained for each of the aforementioned outlier detection algorithms. We specially analyze the influence of the applied exchange strategies on the performance of the chosen algorithms.

TABLE III
MCC VALUES OF THE SELECTED OUTLIER DETECTION ALGORITHMS FOR DIFFERENT PARAMETER VALUES AND EXCHANGE STRATEGIES

		MCC values																	
		Realistic strategy					Random-unfair strategy					Random-fair strategy							
		Min	Q_1	Q_2	Q_3	Max	Min	Q_1	Q_2	Q_3	Max	Min	Q_1	Q_2	Q_3	Max			
Velocity-based	v_{max}	1	-0.36	-0.02	0.11	0.22	0.39	-0.36	0.00	0.11	0.23	0.46	-0.79	0.16	0.37	0.59	0.99		
		5	-0.72	0.04	0.26	0.39	0.95	-0.36	0.16	0.30	0.51	0.95	-0.79	0.11	0.29	0.52	0.99		
		10	-0.79	0.02	0.28	0.44	0.99	-0.89	0.10	0.38	0.50	0.98	-0.73	0.20	0.41	0.62	0.98		
		20	-0.91	-0.06	0.27	0.45	0.99	-0.90	0.22	0.36	0.61	0.99	-0.76	0.22	0.46	0.69	0.99		
		30	-0.95	-0.11	0.27	0.46	0.99	-0.90	0.20	0.38	0.61	0.99	-0.79	0.26	0.46	0.73	0.99		
Grubb-based	c	0.70	-0.06	0.12	0.24	0.51	0.98	-0.07	0.15	0.26	0.64	0.91	0.06	0.22	0.43	0.71	0.98		
		0.95	-0.06	0.12	0.20	0.45	0.98	-0.04	0.12	0.36	0.64	0.99	0.05	0.13	0.36	0.68	0.99		
		0.99	-0.08	0.11	0.19	0.59	0.98	-0.04	0.12	0.30	0.62	0.99	-0.14	0.12	0.35	0.65	0.99		
	n	1	-0.22	-0.10	0.04	0.14	0.33	-0.22	0.01	0.05	0.13	0.29	-0.17	0.04	0.09	0.17	0.58		
		25	-0.13	0.10	0.17	0.40	0.99	-0.10	0.12	0.26	0.50	0.96	-0.04	0.12	0.29	0.56	0.99		
		50	-0.06	0.12	0.20	0.45	0.98	-0.04	0.12	0.36	0.64	0.99	0.05	0.13	0.36	0.68	0.99		
		75	-0.19	0.11	0.23	0.51	0.97	-0.01	0.12	0.36	0.64	0.98	-0.01	0.14	0.43	0.70	0.98		
		100	-0.13	0.15	0.24	0.49	0.95	-0.13	0.13	0.34	0.70	0.98	-0.13	0.15	0.41	0.72	0.98		
		Distance-based	p	0.01	-0.19	-0.06	0.05	0.14	0.75	-0.17	0.07	0.13	0.21	0.75	-0.15	0.09	0.14	0.30	0.99
				0.05	-0.22	-0.06	0.05	0.14	0.75	-0.17	0.07	0.13	0.21	0.75	-0.15	0.09	0.14	0.30	0.99
0.50	-0.19			-0.06	0.05	0.14	0.75	-0.17	0.05	0.12	0.21	0.75	-0.15	0.09	0.14	0.30	0.99		
0.75	-0.22			-0.03	0.05	0.13	0.75	-0.21	0.04	0.12	0.20	0.75	-0.15	0.09	0.14	0.30	0.99		
1.00	-0.22			-0.08	0.05	0.14	0.75	-0.17	0.04	0.13	0.21	0.75	-0.15	0.09	0.14	0.30	0.99		
d_{min}	1		-0.15	0.06	0.38	0.80	0.98	-0.15	0.06	0.38	0.80	0.98	-0.17	0.18	0.52	0.83	0.97		
	25		-0.18	0.00	0.14	0.20	0.91	-0.16	0.11	0.18	0.31	0.91	-0.12	0.14	0.22	0.40	0.99		
	50		-0.19	-0.06	0.05	0.14	0.75	-0.17	0.05	0.12	0.21	0.75	-0.15	0.09	0.14	0.30	0.99		
	75		-0.20	-0.02	0.09	0.16	0.75	-0.18	0.05	0.12	0.18	0.75	-0.15	0.06	0.14	0.26	0.99		
	100		-0.20	-0.11	0.05	0.13	0.24	-0.18	-0.02	0.08	0.11	0.14	-0.15	0.02	0.06	0.12	0.23		
Density-based	k	1	-0.16	-0.07	0.01	0.17	0.29	-0.10	0.06	0.15	0.49	0.99	-0.16	-0.11	0.00	0.14	0.29		
		25	-0.16	-0.06	0.27	0.45	0.99	-0.15	0.07	0.18	0.33	0.95	-0.08	0.14	0.30	0.35	0.99		
		50	-0.16	-0.11	0.27	0.49	0.99	-0.12	0.01	0.20	0.36	0.93	-0.07	0.14	0.26	0.47	0.99		
		75	-0.16	-0.05	0.26	0.48	0.99	-0.13	0.15	0.34	0.59	0.98	-0.06	0.20	0.35	0.87	0.99		
		100	-0.17	-0.05	0.16	0.44	0.93	-0.13	0.15	0.35	0.67	0.98	-0.07	0.19	0.42	0.83	0.99		
	t_d	1	0.01	0.10	0.18	0.18	0.18	0.01	0.07	0.16	0.20	0.25	0.01	0.10	0.13	0.18	0.25		
		5	-0.07	0.03	0.17	0.38	0.92	-0.10	0.17	0.30	0.43	0.89	-0.09	0.20	0.30	0.46	0.89		
		10	-0.16	-0.11	0.27	0.46	0.99	-0.12	0.01	0.20	0.36	0.93	-0.07	0.14	0.26	0.47	0.99		
		15	-0.07	0.00	0.13	0.36	0.93	-0.14	0.10	0.22	0.34	0.94	-0.09	0.10	0.19	0.36	0.93		
		20	-0.03	0.01	0.16	0.37	0.80	-0.13	0.02	0.22	0.35	0.94	-0.10	0.08	0.22	0.37	0.94		

1) *Velocity-based Outlier Detection*: The first row of Table III illustrates the minimum, maximum, and quartiles (Q_1 , Q_2 , and Q_3) of the MCC values computed for the 20 users of the dataset and different maximum velocity values v_{max} . We vary v_{max} between 1 and 30 m/s in order to model pedestrians, cyclists, and vehicle passengers and drivers. The first column corresponds to the application of the realistic strategy, while the second and third columns correspond to the random-unfair and random-fair strategies, respectively.

The medians (Q_2) show that the performance of the velocity-based algorithm in identifying exchanged triplets when users apply the realistic exchange strategy is considered as *fair* to *good* on average, according to our classification introduced in Table II. In other words, only few paths can be correctly identified. In comparison, the random-unfair exchange strategy identification accuracy is higher than that of the realistic strategy and can be regarded as *good*. Algorithm performance improves ever further in the case of the random-fair strategy.

The algorithm is, however, capable of identifying almost all jumbled triplets for one particular user the data set, independently of the applied exchange strategy. This result can be explained by the distribution of the velocity in his/her set of triplets. In fact, this user shows particularly low velocity

between his/her own triplets and thus, a high v_{max} setting allows the algorithm to better filter exchanged triplets.

In summary, the velocity-based algorithm overall can identify most exchanged triplets when clients apply the random-fair exchange strategy. In contrast, it performs worse when the realistic exchange strategy is applied. This allows us to conclude that the privacy of the users moving at different velocities is best protected when they apply the realistic strategy in order to exchange their data.

2) *Grubb-based Outlier Detection*: The second row of Table III depicts the MCC values for different confidence levels (c) and number of iterations (n). Recall that the confidence level determines the threshold above which a triplet is considered to be an outlier using the computed value of Z_s . We have selected high values for the confidence level in order to consider only those triplets identified as being exchanged and thus reduce the number of false positives. For different confidence levels, the performance of the Grubb-based algorithm in identifying jumbled triplets is regarded as *fair* to *good* in most cases. The performance of the Grubb-based detection algorithm even reaches a *very good* level for the random-unfair and random-fair strategies. Overall, this means that the algorithms performs at its the best when the

random-fair exchange strategy is applied, as compared to the random-unfair and realistic strategies.

We next vary the number of iterations n . The number of iterations determines how many times the algorithm runs and how many outliers are identified, as only one outlier is identified per algorithm iteration. We have varied the number of interactions between one and 100, what approximately corresponds to a jumbling degree of 50%. Overall, the algorithm performance improves globally with the number of iterations for all exchange strategies until stabilization. For the realistic strategy, the algorithm performance is considered as *poor* to *fair* for $n = 1$, while it is *fair* to *good* from $n = 25$. For the random-unfair and random-fair strategies, the performance is slightly better and is categorized as *fair* to *very good* based on the values of the quartiles. However, the minimum MCC value is lower for $n = 100$ than for $n = 75$ for both the random-unfair and random-fair strategies. This means that the algorithm is about to reach the threshold above in which the number of false positives increases. If all outliers have already been identified, additional runs of the algorithm lead to the identification of original triplets as outliers and cause the decline of the algorithm in performance. The medians comparison shows that the algorithm performs globally better when users apply the random-fair exchange strategy compared to the random-unfair strategy. This means that the algorithm is able to better distinguish original triplets from jumbled ones. In summary, the best privacy protection is again offered by the realistic strategy, while the worst protection is provided by the random-fair strategy.

3) *Distance-based Outlier Detection*: In this algorithm, a triplet is considered to be an outlier if at least a fraction p of the triplets show a distance greater than the distance d_{min} from the considered triplet. Note that the lower the value of p , the more outliers are assumed. We therefore study the impact of (1) the fraction of triplets p and (2) the distance d_{min} on the performance of the algorithm in identifying jumbled triplets and present the results in the third row of Table III. The values of the parameters are chosen based on the distribution of the triplets and the recommendations of [9].

We first vary the fraction of triplets p . For all exchange strategies, the variation of p does not influence the performance of the algorithm. Similar to the two previous algorithms, the outcome is slightly better for the random-fair strategy compared to the realistic and random-unfair strategies when scrutinizing the quartiles. However, this algorithm globally performs worse than both the aforementioned velocity-based and Grubb-based outlier detection algorithms showing only a *fair* performance.

In the second step, we vary d_{min} . The results highlight that the algorithm performs best for $d_{min} = 1$ and for all exchange strategies. In this case, the algorithm performance is regarded as *fair* to *very good* for all exchange strategies. For the remaining d_{min} values, the behavior it exhibits is mainly *fair* with the exception of $d_{min} = 25$ for both the random-unfair and random-fair strategies. For these strategies, the performance is classified as *fair* to *good*. These results, however,

highly depend on the spatial distribution of the dataset. Indeed, the best performance is reached for short distances, which represent the majority of triplets collected in nearby areas. The larger the distance, the worse the performance, as only few triplets have been collected in more remote locations. Therefore, these results cannot be easily generalized to other datasets. Again, the realistic strategy provides overall the best protection against malicious administrators, seconded by the random-unfair and random-fair strategies when comparing the quartiles.

In summary, these results show that the distance-based outlier detection algorithm performs worse in identifying jumbled triplets when users apply the realistic exchange strategy.

4) *Density-based Outlier Detection*: We finally analyze the impact of the main parameters, i.e., the minimum number of triplets k and the density threshold t_d , of this algorithm on its performance in distinguishing jumbled triplets from original ones. Recall that k is the minimum number of triplets that are required to be in the neighborhood of the triplet of interest and t_d serves as threshold to determine whether a triplet is identified as outlier based on the computed *LOF*. The fourth row of Table III presents the results for different values of k and t_d , respectively.

For the realistic strategy, the performance of the density-based outlier detection algorithm is considered as *poor* to *fair* for $k = 1$ and then improves to *good* from $k = 25$. In comparison, the outcome is *fair* to *good* from $k = 1$ to $k = 50$ and *fair* to *very good* from $k = 75$ for the random-unfair strategy. For the random-fair strategy, the algorithm shows *poor* to *fair* performance for $k = 1$, *fair* to *good* performance for $k = 25$ and $k = 50$, and *fair* to *very good* performance from $k = 75$. Again, the algorithm shows the same behavior as the previous studied algorithms concerning the impact of the applied exchange strategies: the application of the random-fair strategy enables the algorithm to correctly identify more jumbled triplets than with the other strategies.

Finally, we vary the density threshold t_d . The performance of the algorithm in identifying exchanged triplets is considered as *fair* to *good* for all exchange strategies, with the exception of $t_d = 10$ for the realistic strategy where the first quartile is below zero. As previously determined, the random-fair strategy allows a better identification of the jumbled triplets compared to the other strategies.

In summary, as for all studied outlier detection algorithms, the density-based algorithm performs worst when users apply the realistic exchange strategy.

D. Evaluation Summary

We have shown that the performance of the selected outlier detection algorithms in identifying jumbled triplets is globally not *excellent*, but remains *fair* to *good* with some *very good* exceptions. Yet, the algorithms do not succeed in identifying all jumbled triplets as illustrated by negatives MCC values. They, however, perform better when users apply the random-fair strategy. In this case, only few triplets are exchanged at each encounter. They are hence more easily identifiable by the

algorithm due to their potential difference in both time and space dimensions. By exchanging more triplets, the triplets form groups that are coherent in terms of spatiotemporal distribution and prevent the algorithms from detecting them. Additionally, single user triplets have been almost perfectly detected by all algorithms. In this case, additional mechanisms can be run on the mobile phones to remove triplet specificities by smoothing the spatial disparities before their reporting to the application server. To conclude, the realistic exchange strategy has shown to provide the best user privacy protection and should hence be adopted as baseline exchange strategy.

VI. RELATED WORK

We have proposed an approach to protect user's privacy by breaking the link between the spatiotemporal context of the sensor readings and the users who collect them. A simple alternative to our mechanism could be that the users use pseudonyms to report the triplets to the server. However, it was demonstrated in [3], that the application can infer the real identity of the users by tracking their location traces over multiple reports, since it may expose the location of their workplaces and homes. Our approach shares features with the concept of mix zones [11], where the users change their pseudonyms when they encounter other users. However, our mechanism does not involve pseudonyms, but is solely based on the triplets collected by the users, which are actually exchanged. Our concept shares also similarities with [12], [13] and [14]. In these schemes, the paths followed by users are replaced by newly generated paths intersecting with those of other users. However, these schemes are centrally organized and require users to transmit their current and actual position to a trusted entity for the generation of the new paths. Finally, our scheme shares similitudes with the data aggregation scheme proposed in [15]. This decentralized scheme is based on data slices equally distributed between neighbors before being reported to an aggregation server. Instead of only considering an equal distribution between neighbors, we examine multiple exchange strategies that are however not addressed in this prior work. Our mechanism is also tailored for common participatory sensing applications, where each user individually and directly reports triplets to the application server.

VII. CONCLUSIONS

We have proposed a collaborative and decentralized approach to preserve user's location privacy contributing to participatory sensing applications. Our approach is solely based on the exchange of the collected sensor readings between users in physical proximity in order to conceal the paths they have followed. We have used state-of-the-art outlier detection techniques that we have tailored to the application domain in order to investigate if exchanged triplets can be distinguished. The results show that the performance of the mechanisms in

distinguishing exchanged triplets from original ones is mostly between *fair* and *good* on a scale ranging from *poor* to *perfect*. This means that at most a partial identification of jumbled triplets is possible on average. This allows us to conclude that path jumbling is a viable approach to protect user privacy, and that the best privacy protection is provided when users apply the realistic exchange strategy, i.e., the probabilistic exchange of all collected readings since the last encounter.

ACKNOWLEDGMENT

This work was supported by CASED (www.cased.de). Our thanks go to A. Schaller for his contribution.

REFERENCES

- [1] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-Phone: An End-to-end Participatory Urban Noise Mapping System," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2010, pp. 105–116.
- [2] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys)*, 2008, pp. 323–336.
- [3] J. Krumm, "Inference Attacks on Location Tracks," in *Proceedings of the 5th IEEE International Conference on Pervasive Computing (Pervasive)*, 2007, pp. 127–143.
- [4] K. L. Huang, S. S. Kanhere, and W. Hu, "Preserving Privacy in Participatory Sensing Systems," *Computer Communications*, vol. 33, no. 11, pp. 1266–1280, 2010.
- [5] D. Christin, J. Guillemet, A. Reinhardt, M. Hollick, and S. S. Kanhere, "Privacy-preserving Collaborative Path Hiding for Participatory Sensing Applications," in *Proceedings of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2011, pp. 341–350.
- [6] D. Christin, D. Rodriguez Pons-Sorolla, S. S. Kanhere, and M. Hollick, "TrustMeter: A Trust Assessment Framework for Collaborative Path Hiding in Participatory Sensing Applications," Technische Universität Darmstadt, Tech. Rep. TR-SEEMOO-2012-02, October 2012.
- [7] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding Mobility based on GPS Data," in *Proceedings of the 10th ACM International Conference on Ubiquitous Computing (UbiComp)*, 2008, pp. 312–321.
- [8] S. Shekhar, C.-T. Lu, and P. Zhang, "A Unified Approach to Spatial Outlier Detection," *GeoInformatica*, vol. 7, no. 2, pp. 139–166, 2003.
- [9] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-Based Outliers: Algorithms and Applications," *Very Large Data Bases Journal (VLDB)*, vol. 8, no. 3, pp. 237–253, 2000.
- [10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2000, pp. 93–104.
- [11] A. Beresford and F. Stajano, "Location Privacy in Pervasive Computing," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 46–55, 2003.
- [12] B. Hoh and M. Gruteser, "Protecting Location Privacy Through Path Confusion," in *Proceedings of the 1st IEEE International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM)*, 2005, pp. 194–205.
- [13] J. T. Meyerowitz and R. R. Choudhury, "Hiding Stars with Fireworks: Location Privacy through Camouflage," in *Proceedings of the 15th ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2009, pp. 345–356.
- [14] —, "CacheCloak: Enabling Real-time Location Privacy for Mobile Users," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 13, no. 3, pp. 38–41, 2010.
- [15] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "PriSense: Privacy-preserving Data Aggregation in People-centric Urban Sensing Systems," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, 2010, pp. 1–9.