# TraceMixer: Privacy-Preserving Crowd-Sensing sans Trusted Third Party

Jan Henrik Ziegeldorf, Martin Henze, Jens Bavendiek, Klaus Wehrle
Communication and Distributed Systems (COMSYS), RWTH Aachen University, Germany
Email: {ziegeldorf, henze, bavendiek, wehrle}@comsys.rwth-aachen.de

***Abstract*—Crowd-sensing promises cheap and easy large scale data collection by tapping into the sensing and processing capabilities of smart phone users. However, the vast amount of fine-grained location data collected raises serious privacy concerns among potential contributors. In this paper, we argue that crowd-sensing has unique requirements w.r.t. privacy and data utility which renders existing protection mechanisms infeasible. We hence propose *TraceMixer*, a novel location privacy protection mechanism tailored to the special requirements in crowd-sensing. *TraceMixer* builds upon the well-studied concept of mix zones to provide trajectory privacy while achieving high spatial accuracy. First in this line of research, *TraceMixer* applies secure two-party computation technologies to realize a trustless architecture that does not require participants to share locations with anyone in clear. We evaluate *TraceMixer* on real-world datasets to show the feasibility of our approach in terms of privacy, utility, and performance. Finally, we demonstrate the applicability of *TraceMixer* in a real-world crowd-sensing campaign.**

## I. Introduction

The proliferation of mobile, location-aware sensing devices has enabled and inspired *crowd-sensing*, a new paradigm for large scale data collection. By tapping into the capabilities of smart phone users, crowd-sensing promises higher coverage and availability at lesser costs than traditional methods. Successful real-world crowd-sensing campaigns range from collaborative map creation [1] to monitoring cellular networks [2] or road conditions [3]. Unsurprisingly, crowd-sensing is receiving increasing attention also from commercial players: E.g., the cellular network operator Telefonica sells insights to retailers based on its customers' location data [4], while Strava, a site for tracking sport experiences, promises to support and improve city planning based on its users' activity traces [5].

Common to most crowd-sensing campaigns is the focus on the spatial context of contributed data, e.g., in all previous examples data is annotated with a location. However, numerous attacks on location data and serious real-world incidents [6]–[8], have raised serious concerns over location privacy. Indeed, privacy concerns have been identified as the main obstruction in crowd-sensing [9]–[11]. Clearly, effective location privacy protection mechanisms (LPPMs) are required to resolve participants' privacy concerns and to realize the full potential of the crowd-sensing paradigm.

The majority of existing LPPMs are tailored to traditional location-based services (LBS). In an LBS, users annotate queries with their locations and receive answers personalized to these locations. Despite a shared focus on location information, crowd-sensing has fundamentally different characteristics than these LBS and hence poses different requirements to LPPMs. First, LBS usually involve the disclosure of only single locations, while crowd-sensing campaigns typically collect whole traces. LPPMs for crowd-sensing must thus provide *trajectory privacy* instead of the simpler and weaker notion of *sporadic location privacy*. Second, LBS value the timeliness of answers over their spatial accuracy. In contrast, many crowd-sensing campaigns such as map creation [1] or city planning [5] have rather long-term goals and favor spatial over temporal accuracy. Unfortunately, this mismatch of privacy notions, and utility goals renders LPPMs developed for LBS unsuitable for many crowd-sensing scenarios.

In addition to the mismatching privacy and utility goals, we observe that existing LPPMs typically require a trusted third party (TTP) that acts as an anonymization proxy and learns all data in clear. However, such central entities are vulnerable to attacks, database leaks, or seizure by governments [12]. This is even aggravated by the long-term data collection and storage typical in crowd-sensing. We thus argue that LPPMs must renounce centralized design patterns and should ideally never learn participants' data in clear.

**Our contributions.** In this paper, we propose *TraceMixer*, a novel LPPM for crowd-sensing. *TraceMixer* departs from traditional LBS and considers the inherent special requirements of crowd-sensing that hitherto remain unaddressed by related work (Sect. II). To address these novel requirements, *TraceMixer* implements a mix-zone based anonymization approach that i) provides trajectory privacy, ii) introduces minimal spatial distortion, and iii) is realized using advanced cryptographic protocols that operate only over encrypted locations such that traces of the participants are never shared in clear with anyone (Sect. IV). The implementation and evaluation of our prototype show the feasibility of *TraceMixer* in terms of performance, location privacy, and data utility (Sect. V). Finally, we present a real-world use case that demonstrates the real-world applicability of *TraceMixer* (Sect. VI). In conclusion, with *TraceMixer* we provide a general LPPM for long-term crowd-sensing campaigns (Sect. VII).

## II. Problem Statement

In this section, we concisely frame the problem of location privacy in crowd-sensing. We formalize our scenario (Sect. II-A), distill requirements for adequate LPPMs (Sect. II-B) and, finally, we analyze related works w.r.t. these requirements and identify the need for novel approaches (Sect. II-C).

### A. Scenario

We consider an abstract model of crowd-sensing campaigns in three phases, i) data collection, ii) privatization, and iii) publication. Data collection is initiated by the *campaign administrator* who instructs the participants with a sensing task. The *participants*, from hereon referred to as users $u_i \in \mathcal{U}$, move around in an area divided into discrete *locations* $\mathcal{L}$. Each user is equipped with a location-aware device, e.g., a smart phone, and continuously collects data reports at her current location. A single *report* is defined by the location and the sensed event. *Events* can be anything from single sensor readings such as the noise level [13] to arbitrarily sophisticated (sensed) phenomena such as road conditions [3], [14]. The event can be empty, e.g., when location information is the main target of data collection [4]. Finally, the users upload reports to the LPPM continuously or in batches. In the second phase, privatization, the LPPM is responsible for applying adequate privacy protection before releasing any data. We emphasize that our focus is on privacy protection regarding the *location information* contained in the uploaded reports. Thereby, we deliberately do not address potential privacy risks concerning the reported events themselves since they are specific to each category of sensed data [9]. In the final publication phase, the LPPM releases the privatized data to interested parties. Since attackers may disguise as benign applications, the LPPM has to trade-off data utility and privacy.

### B. Requirements Analysis

We survey literature and real deployments to analyze the unique requirements of crowd-sensing. These requirements clearly distinguish crowd-sensing from traditional LBS and make evident the need for novel LPPMs.

**Privacy notion.** In most crowd-sensing deployments, the users upload successive reports which form mobility traces, e.g., in road condition monitoring [3], map creation [1], [14], or city planning [5]. Disclosing whole traces involves significantly higher privacy risks than sporadically disclosing only single locations as typical for traditional LBS: Users exhibit unique mobility patterns which can be exploited to reidentify anonymously contributed traces, predict users' future locations, or infer sensitive information [6], [7], [15]. Hence, LPPMs for crowd-sensing must provide *trajectory privacy* instead of the weaker notion of *sporadic location privacy* assumed in the context of traditional LBS.

**Utility goals.** Due to the voluntary, unreliable nature of crowd-sensing, many campaigns focus on data collection about stable phenomena such as long-term mobility patterns [4], [5], locations of roads and buildings [1], or physical road conditions [14]. At the same time, these examples make evident the need for high spatial accuracy of contributed data, e.g., mapping roads and buildings requires spatial accuracy in the order of meters or even centimeters [1]. In contrast, the query-response model of traditional LBS usually requires timely operation. This has led to the development of LPPMs that minimize temporal delays and thus resort to spatial distortion to provide privacy. Existing LPPMs thus optimize for utility goals opposite to those in the typical long-term crowd-sensing campaigns considered in this work. We conclude that adequate LPPMs should optimize spatial over temporal accuracy.

**Trust model.** Finally, we observe that most existing LPPMs are operated as centralized anonymization proxies which learn the locations of all users in clear. However, such central entities are vulnerable to attacks, database leaks, and seizure by governments [12]. Crowd-sensing deployments, which feature rather long-term collection and storage of huge data sets, become particularly attractive targets which aggravates privacy risks. In consequence, crowd-sensing campaigns often fail to motivate enough users to participate [9], [10]. We thus argue that LPPMs in general and especially for crowd-sensing should renounce centralized design patterns and ideally never learn users' locations in clear to minimize attack vectors.

### C. Related Work

We analyze related work w.r.t. the requirements and compare it qualitatively to our own approach. We concentrate on approaches that fulfill our first requirement, trajectory privacy.

**Mix zones.** The concept of mix zones [16] is to introduce quiet zones in which users do not report locations. In analogy to anonymous communication networks, users must stay in those mix zones for a certain time so that the entry and exit events (i.e., users entering and leaving the mix zone) become unlinkable to outside observers. Several works improve upon the design and placement of mix zones, e.g., [10], [17], [18]. Our approach is also based on the idea of mixing users, but introduces two key differences compared to traditional concept of mix zones: Related works place mix zones *a priori*, hence cannot guarantee that users will actually mix. In contrast, our approach checks *a posteriori* whether a certain number of users have mixed and releases data only then.

**k-anonymity.** Different approaches adapt *k-anonymity* [19] to traces by aggregating $k$ traces such that they become indistinguishable to an attacker [20]–[23]. These approaches introduce significant spatial distortion which violates our utility requirement. Further, they require either a TTP which violates our trust model or require direct interaction between users which is difficult to achieve in the crowd-sensing scenarios targeted in this work, e.g., we observe only rare encounters in the (popular) datasets used in our evaluation.

**Differential privacy.** A recent line of research on trajectory privacy [24]–[26] derives from the *differential privacy* framework [27]. Basically, traces are privatized by adding carefully calibrated noise. Approaches based on differential privacy generally provide stronger privacy than the k-anonymity-based approaches, and can also be implemented in a user-centric fashion without a TTP. Due to the inevitable spatial distortion introduced by the addition of differentially private noise our utility requirements are, however, not met. For example, in [25] the added noise grows linear in the number of disclosed locations which greatly degrades utility even of short traces.

**Uncertainty.** Different approaches use the uncertainty of a tracking adversary as privacy metric and optimization goal. Uncertainty is achieved, e.g., through path confusion [28]

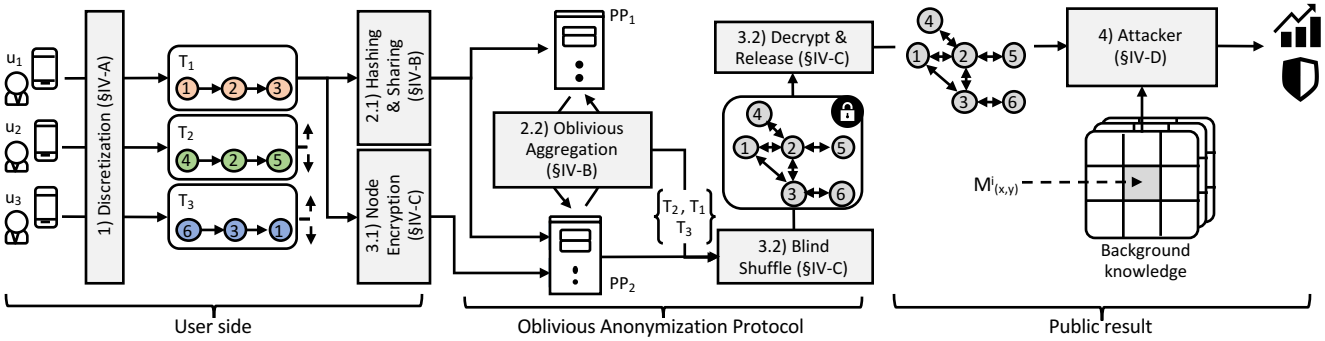


Fig. 1. Overview of *TraceMixer*: On the user side, traces are collected and preprocessed for the anonymization mechanism. The oblivious anonymization protocol is executed by two privacy peers, $PP_1$ and $PP_2$, which determine aggregates of intersecting traces that can be safely released. Finally, the achieved level of privacy is measured by the success of a tracking attacker with background knowledge about users' mobility patterns.

or path cloaking [29]. Both approaches require TTPs and significantly decrease data utility due to spatial distortion and data loss, respectively. In [30], Bayesian Stackelberg games are used to determine an optimal trade-off between utility and privacy. Being completely user-centric, the framework fits our trust model but is limited to spatial perturbation as privatization mechanism. Our approach uses different means of privatization and thus cannot be modeled in this framework.

**Dummy trajectories.** In this class of approaches, either a TTP or the users introduce a large set of dummy trajectories to hide the real trajectories within, e.g., [31]. This approach is unsuitable to crowd-sensing since the large amount of fake dummy reports cannot be distinguished from the real reports.

**Other approaches.** As summarized in [9], many proposals consider *anonymous reporting* as the main solution to protecting location privacy in crowd-sensing. However, anonymizing identities alone does not provide sufficient privacy, since location traces can be easily re-identified and sensitive information can be inferred afterwards [6], [7], [15]. Like our work, the approach in [32] is motivated by the goal to achieve high spatial accuracy. Their optimzation algorithm requires a TTP with global knowledge of all traces and considers the inference of single locations instead of a tracking attacks, i.e., assumes a different privacy notion than our work. We conclude that none of the related works proposes an LPPM that fully addresses the special requirements of the crowd-sensing scenarios targeted in this work.

## III. Background on Secure Computations

A fundamental building block of our approach is secure two-party computation (STC). It allows two mutually distrusting parties holding private inputs $x$ and $y$ to compute a functionality $\mathcal{F}(x, y)$ *obliviously*, i.e., without anyone learning the private inputs. STC achieves this by cryptographically transforming private inputs $x$ and $y$ such that they are not revealed to anyone but can still be used to compute $\mathcal{F}(x, y)$.

In the classical STC scenario, the secure computation is executed by the two peers that own $x$ and $y$. However, as explained in [33], state-of-the-art STC techniques can also be used to efficiently realize oblivious computations over inputs from $n > 2$ users: The $n$ users first provide their respective private inputs $x_1, ..., x_n$ in a secure manner (e.g., using encryption or secret sharing) to two non-colluding privacy-peers $PP_1$ and $PP_2$. The privacy peers can then compute the desired functionality $\mathcal{F}(x_1, ..., x_n)$ over the protected private inputs using state-of-the-art STC protocols, e.g., Yao's Garbled Circuits [34].

## IV. TraceMixer Design

We now explain the design of *TraceMixer* and how it addresses the special requirements of crowd-sensing in detail. We start from a high-level system overview, then explain each component and their interaction in detail (Sects. IV-A to IV-D).

**System overview.** On the highest level, *TraceMixer* follows traditional anonymization approaches: Data records are collected from a number of sources, anonymized through an adequate mechanism, and finally released to the public in privatized form. Departing from all previous approaches, *TraceMixer* anonymizes traces without spatial distortion and obliviously, i.e., without anyone learning a participant's trace in clear. The core idea that facilitates these key differences is to employ Private Set Intersection (PSI) to obliviously find sets of intersecting traces that can be safely released. Here, the intuition is that intersections of traces form natural mix zones which prevent attackers from tracking users if enough such mix zones are present in a released aggregate. PSI, a classical STC protocol, ensures that individual traces are never revealed in clear and that even the LPPM remains oblivious to them.

Figure 1 provides an overview of *TraceMixer*. At the core of *TraceMixer* is the oblivious anonymization protocol which is executed by two privacy peers and divided into two parts: i) We obliviously determine sufficiently large sets of intersecting traces. ii) Once such a set is found, we shuffle, decrypt, and release the traces in this set. The task of the user side component is to discretize traces (Step 1) and then prepare traces for the two parts of the anonymization mechanism: *Hashing and Sharing* (Step 2.1) prepares the inputs for the adapted PSI protocol of our oblivious aggregation mechanism (Step 2.2). *Trace Encryption* (Step 3.1) prepares shuffle and data release (Steps 3.2 and 3.3). We explain Step 2 in Section IV-B and Step 3 in Section IV-C. Finally, in Step 4 (Section IV-D),

**Input:** Traces $\mathcal{T} = \{T_1, ..., T_n\}$, $\mathcal{A} = \{\}$, $k \in \mathbb{N}^+$
**Output:** Aggregates $A_1, ..., A_m$ with $A_l \subseteq \mathcal{T}$ and $|A_l| \geq k$

```
1: for all T_i ∈ T do
2:     for all A_l ∈ A do
3:         for all T_j ∈ A_l do
4:             if T_i ∩ T_j ≠ ∅ then
5:                 A_l.add(T_i);
6:                 if |A_l| ≥ k then
7:                     A.pop(A_l)
8:                     ShuffleRelease(A_l)
9:                 goto Line 1
10:    A.add({T_i})
```

Fig. 2. Trace aggregation algorithm on cleartexts: The algorithm finds aggregates of at least $k$ traces that intersect and releases them. This algorithm is executed as a secure two-party computation between the two privacy peers.

we quantify how much privacy this anonymization approach actually achieves. To this end, we propose a state-of-the-art tracking adversary whose success probability serves as our privacy metric.

### A. Discretization

The goal of *Discretization* is to map traces from continuous GPS coordinates into a discrete location space. This serves two purposes: First, raw GPS data is error-prone and for many use cases overly precise. Discretization allows to adequately set the required granularity and to smooth out errors. Second, we use a PSI protocol to find intersections of traces which requires discrete locations. At first sight, the use of PSI seems to limit our approach. However, discretization introduces artificial intersections, e.g., between two traces that run close but without intersections, which is desirable as it increases privacy.

To discretize continuous GPS samples, we choose to map GPS coordinates to nearby reference points on OpenStreetMap (OSM). This mapping can be done efficiently on the client and offers worldwide coverage. To put an upper limit on the discretization error, we only map to OSM nodes that are within a maximum distance of $2\,\mathrm{m}$ and discard other locations. All further steps are now carried out over the discretized traces which are represented as lists of OSM node IDs.

### B. Oblivious Aggregation

The goal of the first part of our oblivious anonymization protocol is to find aggregates of at least $k$ intersecting traces. We first describe our algorithm and then explain how it can be realized to operate obliviously over encrypted traces.

**Insecure aggregation.** Figure 2 shows how our aggregation algorithm proceeds on cleartext traces $\mathcal{T}$. We start with an empty set of aggregates $\mathcal{A}$. For each received trace $T_i \in \mathcal{T}$, we greedily search for an aggregate $A_i \in \mathcal{A}$ intersecting $T_i$ (Lines 1 to 4). If such an aggregate $A_l$ is found, we add the trace $T$ to it (Line 5). As soon as the privacy criterion $|A_l| \geq k$ is fulfilled, we remove $A_l$ from the set of aggregates $\mathcal{A}$ and hand it to the second part of our anonymization mechanism (Lines 6 to 8), which shuffles and releases the aggregate. If no aggregate that intersects with the received trace $T_i$ is found, we create a new aggregate that only contains $T_i$ (Line 10).

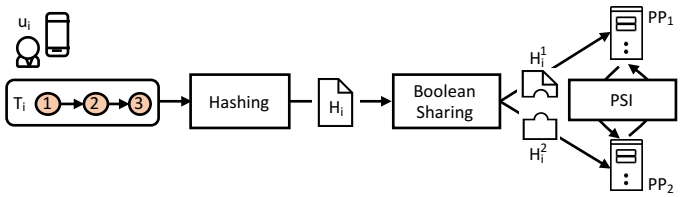


Fig. 3. Each peer computes two hash tables of its trace and shares them to the privacy peers to prepare the PSI protocol.

**Oblivious aggregation.** To achieve oblivious aggregation, we implement this algorithm as a secure two-party computation protocol that operates over encrypted traces only. The set intersection (Line 4) is the only step that requires knowledge of the traces' nodes, while all other steps just manage meta data, i.e., set memberships. Our basic idea to make aggregation oblivious is to replace the insecure set intersection with a PSI protocol executed by two privacy peers over encrypted traces. Thus, they only learn whether two traces intersect but nothing else. PSI is a classical, well-researched secure two-party computation problem and we use the most efficient PSI protocol proposed in [35].

The original protocol from [35] proceeds as follows. Each privacy peer $PP_1$ and $PP_2$ holds in clear one of the two sets to be intersected. Each peer then represents its respective set as a hash table with a configurable amount of bins. $PP_1$ and $PP_2$ then use boolean secret sharing to securely distribute hash tables between each other. Using the secret shares, they can now execute private equality testing to obliviously compare the bins element-wise which efficiently tests whether the two sets intersect. In our scenario, $PP_1$ and $PP_2$ would thus need to hold either $T_i$ or $T_j$ in clear. However, this strongly contradicts our privacy goals, as users should not share their traces with any other party in clear (cf. Section II-B).

We hence adapt the original protocol to our setting such that it respects our trust model. Our basic idea is that users carry out the hashing steps of the original protocol themselves and provide privacy peers the hash tables in secret-shared form. The secret shares do not expose any information to *a single privacy peer* but enable *both peers together* to carry out the PSI protocol. Figure 3 illustrates these steps: User $U_i$ hashes her trace $T_i$ to derive the hash table $H_i$. Still, $H_i$ carries significant information about $T_i$ and thus must not be revealed to the privacy peers in clear. Instead, $U_i$ computes a boolean secret sharing and provides each privacy peer one of the two secret shares $H_i^1, H_i^2$. The privacy peers cache shares locally and can then obliviously intersect $T_i$ with other traces using the PSI protocol from [35] to find aggregates of $k$ intersecting traces. Once an aggregate has been found and released, the shares of the traces are deleted.

### C. Shuffle and Release

In the previous step, we have determined aggregates $A_l = \{T_{i_1}, ..., T_{i_k}\}$ of intersecting traces. Now, these aggregates $A_l$ need to be safely released. So far, privacy peers only hold shares of the hash tables of each trace. If those shares

were recombined directly, the privacy peers would obtain each hash table (i.e., practically the original trace) in clear which violates our privacy requirements. Our basic idea to prevent this is to let the *users* break up traces $T_{i_1}, ..., T_{i_k}$ into small encrypted fragments (in addition to sharing them as hash tables) which are first shuffled blindly by the privacy peers with the fragments of the other traces in the aggregate before all fragments are decrypted and released together. Intuitively, the blind shuffle prevents privacy peers or outside attackers from determining which fragment belongs to which trace $T_{i_j} \in A_l$. The fragment length should generally be minimized, since this decreases the chances of an attacker to successfully recreate traces from the shuffle fragments. However, fragment length must also be decided per use case, e.g., for our use case of constructing elevation profiles (Sect. VI), we require fragments of length two, while measuring people density [4] even requires only one node per fragment.

Figure 1 includes an example of our idea. Trace $T_1$ is broken up into fragments $\{(1,2),(2,3)\}$ which are encrypted individually by $u_1$. Users $u_2$ and $u_3$ prepare $T_2$ and $T_3$ in the same way. Having determined the aggregate $A_l = \{T_1, T_2, T_3\}$, the privacy peers put the corresponding fragments into one big set and shuffle this set blindly. Finally, the privacy peers release the shuffled set, e.g., $\{(2,5),(1,2),(3,6),(2,3),(2,4),(1,3)\}$. From this set of fragments, it is possible to build many traces besides $T_1, T_2$, and $T_3$, e.g., $(4,2,3),(4,2,3,1),(1,2,5),....$ Since the original traces are hidden among these numerous other traces, an attacker can only guess which are the original traces. We evaluate his success chances in Section V-C.

Now, we explain how to realize the shuffle and release. As a preparation (Step 3.1 in Figure 1), each user divides her trace $T_i$ into fragments $f_{i_j} \subseteq T_i$ and encrypts each fragment with the public key $PK_1$ of $PP_1$ using a semantically secure crypto system $E$. The encryptions $E_{PK_1}(f_{i_j})$ are then sent to $PP_2$. When an aggregate of $k$ intersecting traces is found in Step 2, $PP_2$ shuffles the corresponding encrypted fragments and sends them to $PP_1$. $PP_1$ shuffles all fragments again, then decrypts and releases each fragment. Note that this realizes a blind shuffle, i.e., no single privacy peer knows or is able to reverse the shuffle in order to recreate traces from fragments.

### D. Attacker

In line with [17], [28], [29], [36], we consider tracking as the primary attack on users' privacy. Being able to track an anonymous user along her trace, an attacker can not only infer private information but is often able to identify the user based on her unique mobility patterns [6], [7], [15]. The attacker's success is usually measured in the distance that the attacker can correctly track a user. In our setting, the attacker aims to reconstruct users' traces from the released trace aggregates.

**Assumptions.** Before we explain in detail the attack, we make different worst-case assumptions: First, we assume that the attacker has background knowledge about each user. Concretely, we build mobility profiles for each user that are available to the attacker (cf. Figure 1, right). The mobility profiles tell the attacker the probability that user $u_i$ moves from a node $x$ to $y$. Second, we assume that the attacker knows the starting points of all users, e.g., a known home address. Finally, we defensively assume that released fragments overlap such that they can be connected to traces. Our tracking attack would need adjustments for non-overlapping fragments but, more importantly, be less effective. In conclusion, the assumptions we describe strengthen the attacker and thus yield a very defensive measure of the achieved privacy.

**Attack.** The attacker knows the user's starting point and follows her until the first mix zone, i.e., an intersection of at least two traces, by following the fragments' overlaps. A mix zone is simply a node $n$ which is traversed by multiple users (e.g., node 2 in Figure 1) Thus, to track users across mix zones the attacker needs to guess which exit node $e_1, ..., e_m$ was taken by which user. We use a Bayes estimator to derive the probability of a user taking one of the $m$ exits [17]:

$$P_n(u_i|e,n) = \frac{P_n(e|u_i)\,P_n(u_i)}{\sum_{u_j \in U} P_n(e|u_j)\,P_n(u_j)} \quad (1)$$

$P_n(u_i)$ and $P_n(e|u_i)$ are derived from users' mobility profiles. Applying Equation 1, the attacker computes the likelihood of all combinations of users and exit events. The attacker now computes a maximum weight matching between users and exit nodes using the likelihoods as weights, which results in the most probable assignment of users to exit events. Finally, we measure for each user how far along her contributed trace an attacker is able to track her in the released aggregate. We then define our overall privacy metric as the fraction of all users the attacker can track over a certain distance.

TABLE I
OVERVIEW OF THE DATASETS USED IN THE EVALUATION.

| Datasets | Trucks | Geolife | Aachen |
|---|---|---|---|
| Total traces | 276 | 1341 | 5229 |
| Total users | 50 | 96 | 3410 |
| Total nodes | 61 181 | 472 052 | 7 219 405 |
| Timespan in days | 303.13 | 1631.06 | 100.58 |
| Mean nodes per trace | 443.34 | 352.28 | 1380.91 |

## V. EVALUATION

We first evaluate performance, utility and privacy and conclude with a qualitative security and privacy discussion.

**Implementation.** We implemented a complete prototype of *TraceMixer*. To this end, we realize the user-side component as an Android application and the anonymization mechanism as well as the attacker in Python. However, for the performance-critical private set intersection protocol, we use the efficient C++ implementation provided within the *ABY* secure two-party computation framework [33]. Public-key cryptography for the trace preparation on the client-side and for the release step on the privacy peers is implemented using *libsodium*.

**Experimental setup.** We measure the performance of the user-side component on an LG Nexus 5 smart phone and execute the anonymization mechanism between two desktop machines (Ubuntu 14.04, Intel i7-4770 @ 3.10 GHz, 16 GB RAM) that communicate over a Gigabit LAN. All results are given as mean and standard deviation (SD) over 5 runs.

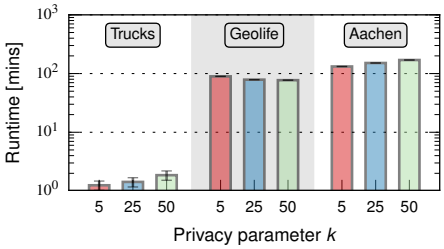


Fig. 4. Runtime of the anonymization mechanism on the whole datasets in minutes.

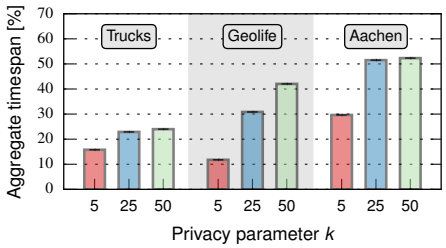


Fig. 5. Average timespan per aggregate normalized over the timespan of the whole dataset.

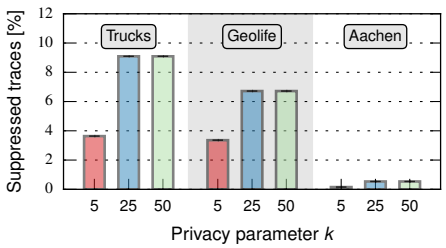


Fig. 6. Percentage of traces that could not be anonymized and have to be suppressed.

**Datasets.** We evaluate all three components of *TraceMixer*, i.e., the user-side application, the oblivious anonymization mechanism, and the attacker, on three (popular) real-world datasets: i) Trucks [37], ii) a subset of Geolife [38], and iii) a large dataset of sports activities we collected around the city of Aachen. We choose these datasets because they feature largely different characteristics (as summarized in Table I) that allow us to evaluate *TraceMixer* in very different settings.

### A. Performance Evaluation

We measure the overheads for i) trace preparation, ii) oblivious aggregation, and iii) shuffling and releasing fragments.

**Trace preparation.** We first study performance on the user-side. To this end, we measure the time to encrypt $100$ to $5000$ trace fragments. As expected, the overhead is linear in the number of fragments and requires only $4.49\,\mathrm{s}$ (SD $0.06\,\mathrm{s}$) even for very large traces with $5000$ fragments. Note that the longest trace in our dataset has only $4747$ nodes and the most other traces are significantly smaller. Thus, the overheads for trace preparation are clearly manageable for smart phones.

**Oblivious aggregation.** The runtime of our oblivious aggregation algorithm on the three datasets is shown in Figure 4. We process traces in their chronological order and measure the total processing time. *TraceMixer* requires only few minutes on the small dataset, Trucks, while runtimes range in the order of a few hours for the large datasets, Geolife and Aachen. At least $93\,\%$ of these efforts can be precomputed. Even if datasets are anonymized in one batch (as in our evaluation), these overheads are manageable. More importantly, for all three datasets the presented runtimes only constitute far less than $0.001\,\%$ of the timespan over which the dataset was collected (cf. Table I). Thus, the presented runtime overheads are almost negligible when datasets are anonymized on the fly.

Interestingly, the difference between the Geolife dataset and the much larger Aachen dataset are not as big as their sizes suggest (cf. Table I). This is due to the different densities of the datasets. Intuitively, traces intersect more frequently in denser datasets such that our approach needs less tries to find an aggregate to merge a new trace into. The Trucks and Aachen datasets are rather dense and hence already after an average of 3 intersections an intersecting aggregate is found. Geolife is sparse in comparison and requires on average 12 intersections per trace before the trace can be merged. In consequence, although much larger, the Aachen dataset requires a similar total amount of intersections as the Geolife dataset.

**Shuffle and release.** Finally, we measure the performance for shuffle and release. To this end, we measure the time to shuffle and decrypt aggregates with up to $250\,000$ trace fragments, which corresponds to the number of fragments in an aggregate of $k = 50$ traces where each trace has maximum length of $5000$ nodes. As expected, the overhead is again linear in the number of fragments and even for the largest aggregates of $250\,000$ fragments requires only $10.64\,\mathrm{s}$ (SD $0.32\,\mathrm{s}$). Thus, the overheads for shuffle and release are clearly feasible on the desktop machines that $PP_1$ and $PP_2$ are run on.

### B. Utility Evaluation

We discuss data utility along three criteria: i) spatial accuracy, ii) temporal accuracy, and iii) suppressed data.

**Spatial accuracy.** The most important utility criterion is the spacial accuracy of the data (cf. Sect. II). In our approach, only the discretization step (cf. Sect. IV-A) introduces small spatial inaccuracies but may also correct errors. Since discretizing locations is not the focus of this work, we did not evaluate these effects but instead set a small discretization threshold of $2\,\mathrm{m}$ which introduces only little inaccuracies in comparison to the error in most GPS receivers of today's smart phones. Notably, our approach does not use spatial obfuscation to anonymize traces other than most related works. Thus, besides the slight effects of discretization, our approach achieves maximum utility in terms of spatial accuracy.

**Temporal Accuracy.** Traces uploaded to *TraceMixer* are annotated with the current time. Further temporal information is stripped off to prevent reconstruction of individual traces through temporal correlation of the nodes. *TraceMixer* only releases the timespan of the traces in a released aggregate. Thus, temporal accuracy increases when aggregates are filled up faster. Figure 5 plots the average timespan per aggregate *normalized* by the timespan over which the dataset was collected, e.g., a timespan of $50\,\%$ means that on average each aggregate is filled and released after half of the time it takes to collect the whole dataset. As we expect, a smaller privacy parameter $k$ leads to smaller timespans since aggregates are filled up faster. How fast aggregates of given size $k$ are filled then depends almost entirely on how fast new traces are uploaded by users which is not determined or limited by our approach but specific to the dataset and crowd-sensing campaign. In conclusion, temporal accuracy is coarse but still practical for our targeted scenarios which require only very coarse or no temporal information at all (cf. Sect. II).

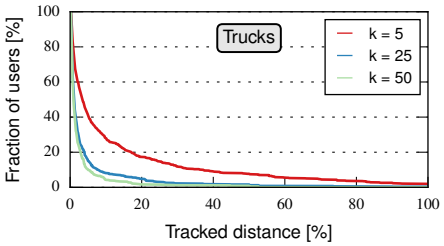


Fig. 7. Attacker's success on the *Trucks* dataset

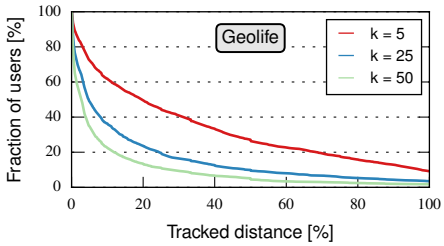


Fig. 8. Attacker's success on the *Geolife* dataset

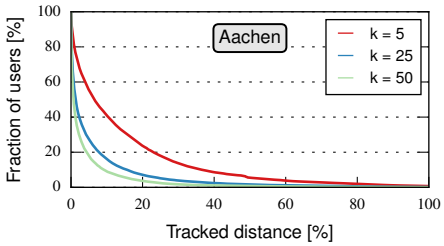


Fig. 9. Attacker's success on the *Aachen* dataset

**Suppressed data.** Finally, the percentage of data that could successfully be anonymized versus the percentage of data that remains suppressed because the privacy criterion could not be fulfilled has an influence on utility. Figure 6 plots the percentage of traces that could not be anonymized using *TraceMixer*. An important insight is that the majority of these traces were collected towards the end of the collection period. These traces are anonymized last and thus suitable aggregates often have already been released. This is confirmed by the fact that the percentages decrease for larger datasets. Thus, when data collection continues, the majority of these traces will eventually be released. The percentage of suppressed traces in Figure 6 must hence be seen as a very pessimistic estimate.

### C. Privacy Evaluation

We evaluate the attacker's success in reconstructing users' traces from the released aggregates. For the attacker's background knowledge, we construct mobility profiles from on up to five randomly selected traces per user. Figures 7 to 9 plot a CDF of the attacker's success measured as the fraction of users (y-axis) he can track over a given fraction of their whole traces (x-axis) for privacy parameters $k = 5, 25, 50$ (lines). We do not plot error bars, since standard deviations are very low, i.e., $2.07\,\%$, $0.72\,\%$ and $0.31\,\%$ on Trucks, Geolife, and Aachen.

On all three datasets, the attacker quickly looses track of users, e.g., even on Geolife he can only track $20\,\%$ of the users longer than $20\,\%$ of their traces for $k = 25$. We further observe that increasing the privacy parameter $k$ also significantly decreases the distance the attacker can track users, i.e., privacy for users increases. As we observed in our performance and utility evaluation, increasing $k$ only slightly increases runtime (Figure 4) and has only moderate impact on utility (Figures 5 and 6). Thus, it is clearly feasible to parameterize *TraceMixer* for high levels of privacy, e.g., $k = 50$ and beyond.

### D. Security and Privacy Discussion

**Security guarantees.** The core of *TraceMixer* is the oblivious aggregation which is based on the PSI protocol proposed in [35]. The original protocol is secure in the semi-honest attacker model which requires that privacy peers do not collude and do not actively cheat. Security in the semi-honest model guarantees that no information is leaked about the inputs except for what is implied in the output. The semi-honest model is a standard choice for STC as it allows for efficient protocols and protects against insider and outsider attacks. For *TraceMixer*, we need to modify the original protocol such that the hashing steps are executed on the client and the hash tables are directly distributed to the privacy peers. Since hash tables are secret shared to the privacy peers and no single privacy peer learns anything from its individual shares, our adaptation is also secure in the semi-honest model.

A second important aspect of the security of *TraceMixer* is the security of shuffle and release. It is important to show that neither privacy peer can reverse the shuffle as this would allow them to recreate the original traces. For $PP_1$ this is impossible, since it obtains the fragments already in completely random order from $PP_2$. For $PP_2$ this is impossible as well, since i) it learns nothing from the fragments due to encryption and ii) cannot relate encrypted fragments to decrypted fragments released by $PP_1$ due to $PP_1$'s own shuffle and the semantic security of the employed encryption scheme. A detailed analysis of similar shuffle schemes was given in [39].

**Privacy guarantees.** It is important to note that the privacy parameter $k$ is *strongly correlated* to the actual achieved privacy level (Figures 7, 8, and 9) but cannot *guarantee* that the attacker's chances are below a certain threshold. Ideally, privacy peers would thus measure the achieved privacy obliviously and release aggregates only when the attacker's success probability falls below a set threshold. However, measuring privacy over encrypted traces is computationally too expensive. Thus, privacy peers need to decrypt aggregates before they can measure the actual level of privacy. Still, they can decide to add further traces obliviously if the aggregate does not achieve the required privacy level. Though, this is not secure against an inside attacker compromising one of the two privacy peers, it protects against outside attackers.

A second limitation are sybil attacks. An attacker can submit $k-1$ traces that all intersect. The first real user who intersects with the fake traces can then be trivially tracked by the attacker. Protection against such sybil attacks can be achieved in *TraceMixer* by requiring user authentication.

## VI. Use Case: Crowd-Sourced Elevation Profiles

To present a concrete use case for *TraceMixer*, we show how to realize a privacy-preserving crowd-sourcing campaign for the creation of high-precision elevation profiles. GPS altitude information is error prone and altitude profiles are often unavailable or very coarse, especially in rural regions. In contrast, most modern smart phones are equipped with barometers which measure altitude very precisely. We observed that the precision of barometric altitude is within centimeters of the real altitude, while GPS altitude may deviate by several meters.

To employ *TraceMixer* in this setting, we implemented the client-side component as an Android application which samples air pressure. The application runs as a background task every $2\,\mathrm{min}$ to minimize energy consumption. We let users report only the difference in air pressure between two nodes. This avoids computing altitude from absolute air pressure measurements, which can vary significantly over time due weather conditions. In contrast, the difference in air pressure between two locations is much less impacted by local weather.

We distributed the application to 9 voluntary users who anonymously collected 3 990 air pressure measurements through *TraceMixer*. Starting from one reference point, we iteratively calculated the altitude of surrounding nodes. To quantify the error, we compare against the altitude data obtained from the the local land-registry which features an altitude resolution of $10\,\mathrm{cm}$ over a $20\,\mathrm{cm} \times 20\,\mathrm{cm}$ grid. For *TraceMixer*, we observe an average error in the altitude of $0.99\,\mathrm{m}$ (SD $0.91\,\mathrm{m}$). This error is higher than our preliminary experiments suggest due to the very low sample frequency which leads to large distances between two sampling locations. Increasing the sample rate increases energy consumption at the client, but allows us to significantly decrease this error.

## VII. Conclusion

We presented *TraceMixer*, a novel location privacy protection mechanisms tailored to the special requirements in crowd-sensing, i.e., providing trajectory privacy protection while preserving high spatial accuracy, that remained hitherto unaddressed by existing location privacy mechanisms. With *TraceMixer*, we achieve these goals through an anonymization mechanism that is inspired by and at the same time reinventing the concept of mix zones. Departing even further from existing protection mechanisms, *TraceMixer* renounces all centralized design patterns that make previous approaches vulnerable to attacks, leaks, and seizure by governments. Instead, *TraceMixer*'s core is implemented using secure two-party computation techniques, allowing *TraceMixer* to operate obliviously: The participants of a crowd-sensing campaign never have to share their data with anyone in clear and even the *TraceMixer* system learns only fully privatized data. As a thorough evaluation on three real-world datasets shows, our approach is feasible even for large datasets and introduces minimal spatial distortion while effectively protecting users' privacy. As a concrete use case, we carried out a crowd-sensing campaign through *TraceMixer*, which demonstrates the privacy-preserving creation of altitude profiles one order of magnitude more precise than GPS. To conclude, *TraceMixer* provides a practical location privacy protection mechanism for a variety of crowd-sensing campaigns ranging from map creation [1], [14], over environmental monitoring [3], [13] to commercial applications [4] and city planning [5].

## Acknowledgments

This work has been funded by the German Federal Ministry of Education and Research (ref. no. 16KIS0443). The responsibility for the content of this publication lies with the authors.

## References

[1] "OpenStreetMap," http://www.openstreetmap.org/.
[2] "OpenSignal," https://opensignal.com/.
[3] P. Mohan *et al.*, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *ACM SenSys*, 2008.
[4] BBC News, "Telefonica hopes 'big data' arm will revive fortunes," http://www.bbc.com/news/technology-19882647, 2012.
[5] "Strava Metro," http://metro.strava.com/.
[6] J. Krumm, "Inference attacks on location tracks," in *PERVASIVE*, 2007.
[7] S. Gambs *et al.*, "De-anonymization attack on geolocated data," *Journal of Computer and System Sciences*, vol. 80, no. 8, 2014.
[8] J. Scheck, "Stalkers Exploit Cellphone GPS," Wall Street Journal, 2010.
[9] D. Christin *et al.*, "A survey on privacy in mobile participatory sensing applications," *Journal of Systems and Software*, vol. 84, no. 11, 2011.
[10] S. Gao *et al.*, "TrPF: A trajectory privacy-preserving framework for participatory sensing," *IEEE TIFS*, vol. 8, no. 6, 2013.
[11] J. H. Ziegeldorf *et al.*, "Privacy in the Internet of Things: Threats and Challenges," *SCN*, vol. 7, no. 12, pp. 2728–2742, 2014.
[12] Privacy Rights Clearinghouse, "Data Breaches," https://www.privacyrights.org/data-breaches.
[13] N. Maisonneuve *et al.*, "NoiseTube: Measuring and mapping noise pollution with mobile phones," in *ICSC Symposium*, 2009.
[14] R. K. Ganti *et al.*, "GreenGPS: A participatory sensing fuel-efficient maps application," in *ACM MobiSys*, 2010.
[15] M. Gruteser and B. Hoh, "On the anonymity of periodic location samples," in *SPC*, 2005.
[16] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive computing*, vol. 2, no. 1, 2003.
[17] J. Freudiger *et al.*, "On the optimal placement of mix zones," in *PETS*, 2009.
[18] X. Liu *et al.*, "Traffic-aware multiple mix zone placement for protecting location privacy," in *IEEE INFOCOM*, 2012.
[19] L. Sweeney, "k-anonymity: A model for protecting privacy," *Int. J. Uncertainty Fuzziness Knowledge Based Syst.*, vol. 10, no. 5, 2002.
[20] C.-Y. Chow and M. F. Mokbel, "Enabling private continuous queries for revealed user locations," in *SSTD*, 2007.
[21] X. Pan *et al.*, "Distortion-based anonymity for continuous queries in location-based mobile services," in *ACM SIGSPATIAL GIS*, 2009.
[22] M. E. Nergiz *et al.*, "Towards trajectory anonymization: a generalization-based approach," in *ACM SIGSPATIAL GIS Workshop SPRINGL*, 2008.
[23] O. Abul *et al.*, "Anonymization of moving objects databases by clustering and perturbation," *Information Systems*, vol. 35, no. 8, 2010.
[24] R. Assam *et al.*, "Differential private trajectory protection of moving objects," in *ACM SIGSPATIAL GIS Workshop IWGS*, 2012.
[25] M. E. Andrés *et al.*, "Geo-indistinguishability: Differential privacy for location-based systems," in *ACM CCS*, 2013.
[26] K. Chatzikokolakis *et al.*, "A predictive differentially-private mechanism for mobility traces," in *PETS*, 2014.
[27] C. Dwork *et al.*, "Calibrating noise to sensitivity in private data analysis," in *TCC*, 2006.
[28] B. Hoh and M. Gruteser, "Protecting location privacy through path confusion," in *SecureComm*, 2005.
[29] B. Hoh *et al.*, "Preserving privacy in GPS traces via uncertainty-aware path cloaking," in *ACM CCS*, 2007.
[30] G. Theodorakopoulos *et al.*, "Prolonging the hide-and-seek game: Optimal trajectory privacy for location-based services," in *WPES*, 2014.
[31] R. Chow and P. Golle, "Faking contextual data for fun, profit, and privacy," in *WPES*, 2009.
[32] M. Terrovitis and N. Mamoulis, "Privacy preservation in the publication of trajectories," in *IEEE MDM*, 2008.
[33] D. Demmler *et al.*, "ABY – A framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.
[34] A. Yao, "How to generate and exchange secrets," in *IEEE FOCS*, 1986.
[35] B. Pinkas *et al.*, "Phasing: Private set intersection using permutation-based hashing," in *USENIX Security*, 2015.
[36] R. Shokri *et al.*, "Protecting location privacy: optimal strategy against localization attacks," in *ACM CCS*, 2012.
[37] E. Frentzos *et al.*, "Nearest neighbor search on moving object trajectories," in *SSTD*, 2005.
[38] Y. Zheng *et al.*, "Mining Interesting Locations and Travel Sequences From GPS Trajectories," in *WWW*, 2009.
[39] J. H. Ziegeldorf *et al.*, "Secure and anonymous decentralized bitcoin mixing," *Future Generation Computer Systems*, 2016.