PAMPAS: Privacy-Aware Mobile Participatory Sensing Using Secure Probes

Dai Hai Ton That[†], Iulian Sandu Popa[†,‡], Karine Zeitouni[†], Cristian Borcea[⋆]
[†]DAVID Laboratory - University of Versailles Saint-Quentin, Versailles, France
[‡]INRIA Saclay-Ile-de-France, Palaiseau, France
[⋆]Department of Computer Science, New Jersey Institute of Technology, Newark, New Jersey, USA
{dai-hai.ton-that,iulian.sandu-popa,karine.zeitouni}@uvsq.fr, borcea@njit.edu

ABSTRACT

Mobile participatory sensing could be used in many applications such as vehicular traffic monitoring, pollution tracking, or even health surveying. However, its success depends on finding a solution for querying large numbers of users which protects user location privacy and works in real-time. This paper presents PAMPAS, a privacy-aware mobile distributed system for efficient data aggregation in mobile participatory sensing. In PAMPAS, mobile devices enhanced with secure hardware, called secure probes (SPs), perform distributed query processing, while preventing users from accessing other users' data. A supporting server infrastructure (SSI) coordinates the inter-SP communication and the computation tasks executed on SPs. PAMPAS ensures that SSI cannot link the location reported by SPs to the user identities even if SSI has additional background information. In addition to its novel system architecture, PAMPAS also proposes two new protocols for privacy-aware location-based aggregation and adaptive spatial partitioning of SPs that work efficiently on resource-constrained SPs. Our experimental results and security analysis demonstrate that these protocols are able to collect the data, aggregate them, and share statistics or derived models in real-time, without any location privacy leakage.

CCS Concepts

•Information systems → Mobile information processing systems; •Security and privacy → Security in hardware;

Keywords

Location privacy; secure protocol; distributed architecture; mobile participatory sensing; spatial aggregates

1. INTRODUCTION

There is an increasing interest in mobile participatory sensing for urban monitoring, which appears to be a better alternative to traditional infrastructure-based sensing to cope with the high installation and maintenance costs, as well as the coverage limitation. Many projects have been conducted recently around the world - or are still ongoing - in the area of environmental participatory sensing [15], such as Citi-Sense in Oslo, CamMobSens at Cambridge, MetroSense at Dartmouth, and OpenSense in Switzerland. Also, many applications that exploit the sensing features of smartphones are already available. Examples include community based traffic monitoring (e.g., Waze[1], or Navigon[2]), finding available parking spaces or noise mapping [9]. In addition, the emerging lightweight low-cost sensors are changing the paradigm of environmental and health monitoring[3], and allow measuring in real-time the individual exposure to environmental risk factors or the propagation of an epidemic.

In these scenarios, the community members act as mobile probes and contribute to spatial aggregate statistics, which in turn, benefit the whole community, e.g., dynamic traffic navigation or air quality mapping and alerts. Various statistics are of interest: basic count and density, average of reported measures by location and time, or more complex geo-statistical operations such as spatial interpolation [14]. Unfortunately, most current mobile participatory sensing systems (MPSS) require users to reveal their locations to trusted monitoring servers, which raises serious privacy concerns because user identity could be determined based on several locations that are linked to the same user [7]. We should stress that, even if users might trust a centralized service, privacy violation examples are legions (see for example DataLossDB.org) coming from negligence, abusive use, internal or external attacks, and such violations affect even the most secured servers. In addition to location, sensing data reported by users could be privacy-sensitive as well. These privacy issues prevent a wide adoption of MPSS.

Several works consider the MPSS privacy problem such as [9], [5], [8], [17], [18]. However, most approaches require trusting a proxy server [8], [18], while others are too costly [9], [5], or sacrifice sensing accuracy for privacy [17].

Hence, providing a high-quality MPSS, while protecting the users' privacy, is still a challenge. Recently, the emergence of personal secure devices has opened new perspectives in personal data protection. Be it a secure portable token [1], [19] communicating with the user's smartphone or plugged inside it (e.g., Google Vault[4]), a tamper-resistant hardware

SSDBM '16, July 18-20, 2016, Budapest, Hungary

DOI: http://dx.doi.org/10.1145/2949689.2949704

[1]http://www.waze.com
[2]http://navigon.com
[3]http://www.epa.gov/heasd/airsensortoolbox/
[4]http://www.cnet.com/news/googles-project-vault-is-a-

security module securing the on-board computer of a vehicle [10], or the secure TrustZone CPU [2] of the ARM cortex-A series equipping most of mobile devices today, all such secure devices offer tangible, hardware-based security guarantees. We leverage their secure data processing capability in a distributed, privacy-by-design architecture, providing an alternative to the traditional server-centric architecture. Our belief is that, similar to TrustZone CPU, such secure devices will become ubiquitous in the near future, equipping by default users' mobile devices. As such, there will be no need for users to buy and connect external hardware to participate in MPPS applications.

This paper presents PAMPAS, a Privacy-Aware Mobile Participatory Sensing system for efficient mobile distributed query processing in the context of MPSS. The novelty of PAMPAS is two-fold: (1) it provides a system architecture that protects users' location privacy by preventing location tracking from any third-party server; and (2) it provides efficient aggregation protocols that satisfy the MPSS real-time constraints in spite of the resource limitations of secure devices. The privacy guarantee gives users strong incentives for participation [11], in addition to the benefits they get from MPSS applications. In PAMPAS, all participants have a mobile device enhanced with secure hardware (i.e., any of the types described above), called a secure probe (SP). SPs act as probes for the target phenomenon, perform distributed query processing, and share the results with the users. The secure hardware prevents users from accessing other users' data during the distributed computation. Secure probes exchange data in encrypted form with help from a supporting server infrastructure (SSI). To provide real-time results, PAMPAS employs efficient, parallel, location-based aggregation protocols which partition the probes according to their geographic distribution. The construction and the maintenance of these partitions aim at reducing and balancing the workloads on worker SPs, while precluding the SSI from doing location-based inference attacks against the participants.

We implemented and validated PAMPAS using representative secure hardware platforms. We used two applications for experiments, traffic and noise monitoring, with both real and synthetic datasets representing small and medium-size cities. Using these applications, we compared PAMPAS with a state-of-the-art secure aggregation protocol described in [19]. The experimental results show that PAMPAS outperforms this protocol in terms of latency and scalability, which translates to much lower resource utilization at the user side.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes the system architecture of PAMPAS, the threat model, and the protocol requirements. Section 4 presents the location-based global aggregation protocol, and Section 5 describes the privacy-aware probe partitioning protocol. The security analysis is presented in Section 6, while the experimental results are shown in Section 7. We conclude the paper in Section 8.

2. RELATED WORK

Traditional system architectures used in MPSS such as [8], [18] rely on a centralized server to collect data from mobile participants, process it, and publish the results. This server-centric model is straightforward and easy to deploy, run, and maintain. However, this basic approach also raises serious privacy concerns and prevents a wide adoption of MPSS. An attacker who is able to link several location reports to the same user can then determine the identity of the user by leveraging, for example, background information (e.g., user home address). Thus, an attacker can identify the MPSS participants and infer their personal habits and activities [7]. In addition to location which is normally included in MPSS reports, the sensing data reported by users could be privacy-sensitive as well. The works that address this issue belong to three classes: (1) server-centric architecture and (2) cryptographic protocols for MPSS, and (3) secure hardware devices in other contexts.

Server-centric approaches. Virtual trip lines (VTLs) [12] deal with the privacy issue by distributing the traffic monitoring service implementation across several specialized servers and by providing a spatio-temporal cloaking of the users under the VTLs. Although the attack of a single system component prevents linking the identity and location of the users, choosing privacy-insensitive locations for VTLs is tricky and limits the traffic information to a part of the road network. Also, the problem of multi-component attack (or collusion) remains, as well as the high cost of building such a complex system distributed over several components. SpotMe [17] proposes a different approach consisting in mixing real user's location with fake locations before posting them to a central server. Then, the server estimates the aggregated user locations by using probability theory. However, the estimation errors can be important (around 20%), while the number of observed spatial units cannot exceed a few hundreds. Also, SpotMe involves higher communication costs because of the large number of fake locations, while linkability may still be a problem for users who send many consecutive location updates, which limits the usability of this approach to sporadic updates.

By employing a fully decentralized architecture for computation, PAMPAS avoids all the above listed problems. Moreover, the trust is enforced by using cheap but highly secure, tamper-resistant hardware at the user side.

Cryptographic approaches. Another way to protect the users' privacy is to use secure cryptographic protocols [5], [9], [16]. Typically, the cryptographic solutions are based on homomorphic encryption schemes allowing a central-server [16] or the users [9] to aggregate the samples directly on the cyphertext. However, the cryptographic methods have to face two major limitations. First, homomorphic encryption only allows the computation of basic aggregate functions (e.g., count, average, standard deviation), while more advanced functions require fully homomorphic encryption schemes, which are not computationally feasible today. Second, even with the basic aggregate functions, the cryptographic protocols can incur a large computation and communication cost [5], [16]. Hence, the existing works typically limit the size of the monitored space (e.g., the number of roads) and the monitoring frequency. Therefore, such solutions cannot meet the scalability and the real-time requirements of MPSS at the same time, and are not generic w.r.t. the type of aggregate function.

Secure hardware approaches. Recent works have also proposed the use of secure hardware at the user-side [1], [19]. The trust in such a distributed architecture in which all computation is done by user devices arises from two sources: (i) the decentralization, i.e., there is no central-server to be

security-chip-disguised-as-an-micro-sd-card/

trusted or to be exposed to attacks having a large benefit/cost ratio; (ii) the (tamper-resistant) secure hardware at the user-side, which protects the devices against physical attacks (even from the device holder).

In [1], Allard et al. propose METAP, a privacy-preserving data publishing protocol in the context of an architecture composed of low power secure devices and a powerful but un-trusted server in order to release sanitized data to third parties. However, this data publishing protocol does not consider the case of spatiotemporal sensed values and cannot be used in participatory sensing aggregations. To et al. [19] propose a similar architecture, but consider the problem of executing SQL queries over a distributed database without revealing any sensitive information to central servers. Considered in our context, this protocol incurs high computation and communication costs because of the specificity of MPSS aggregates (e.g., the aggregate groups are locations, there is a high number of such groups, the computation is continuous and should follow the data generation frequency, the aggregate functions can be complex such as spatial interpolation).

PAMPAS shares the idea of employing a user-centric decentralized architecture with the above mentioned works. However, unlike existing protocols, its secure aggregation protocol is adapted to MPSS requirements, i.e., high dynamicity of the participants, real-time constraints for computation, complexity of the aggregate statistics, and low resource utilization.

A centralized solution based on secure hardware could also be devised using recent proposals to ensure shielded application execution over untrusted servers. For example, Haven [3] extends the hardware level protection features provided by the Intel SGX architecture from code snippets to the entire OS. But there are limitations: this solution slows down the computation substantially; the entire security architecture depends on the chip manufacturer's ability to protect the secret keys; programmers will miss certain features, such as process creation, that are not supported.

3. SYSTEM OVERVIEW

This section presents the system architecture of PAMPAS, the threat model in our system, and the data and computation model of the system. Based on these elements, we derive the requirements for the PAMPAS protocols.

3.1 System Architecture

PAMPAS relies on a hybrid architecture combining secure elements at the user side (secure probes – SP) and a supporting server infrastructure (SSI) that enables secure exchange of messages between the mobile users. SPs and SSI jointly run two protocols to exchange sensed sample updates, continuously compute the spatially aggregated results, and periodically partition SPs according to their location. This architecture fully protects the users' privacy w.r.t. the SSI. Figure 1 shows the general architecture of our system in the context of traffic monitoring.

Compared to a purely decentralized peer-to-peer (P2P) architecture, this hybrid architecture has the salient advantage of not consuming any resources from the participants to maintain the P2P overlay, which is important given the low resources and availability of the user devices. In addition, it exchanges messages between SPs in $O(1)$ hops as opposed to the typical $O(logN)$ hops in P2P networks.

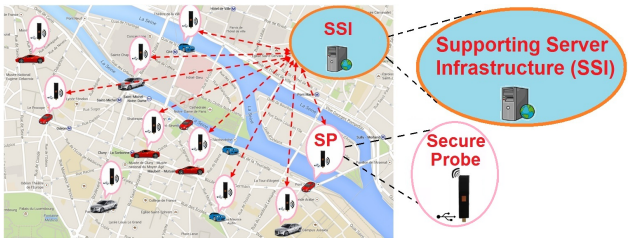


Figure 1: System architecture

Secure Probe (SP). Each user holds a secure portable device, which can be implemented by any type of (tamper-resistant) secure devices flourishing today (see Figure 2) and described in Section 1. Whatever its commercial name and form factor, a secure device, called secure probe (SP) hereafter, embeds at least a secure micro-controller (MCU) for computation (e.g., a smart card chip) connected to a large NAND Flash memory for data storage (e.g., an SD card). An SP plays three roles: (i) mobile probe, (ii) processing node, and (iii) query issuer. The SP sends encrypted samples (containing spatiotemporal sensed measures) to SSI, participates in the data aggregation, and receives the final results from other SPs with the help of SSI. Given their high-level of security, SPs are considered trusted in our system. However, this feature comes at a price. The MCU usually has a low power CPU and a tiny RAM (a few tens of KB). In addition, SPs have low availability since they can be connected/disconnected as required by the users. Therefore, all the computation and communication with the SPs have to be highly optimized.

Figure 2: Examples of secure tokens

Supporting Server Infrastructure (SSI). Different from the typical server-centric architecture, the SSI in our system acts only as a coordinator for exchanging messages between the SPs and for temporary storage purposes. Since the SSI is not trusted, all the temporary results stored in the SSI are encrypted using non-deterministic encryption.

In conclusion, the security and privacy in PAMPAS arise from the combination of secure hardware with a high degree of distribution of the architecture (i.e., all computations are executed by some of the SPs). The challenge is then to be able to continuously compute any type of aggregate functions in real-time in this user-centric architecture given the low resources of the SPs.

3.2 Threat model

The attackers in PAMPAS could be either users or the owners of SSI. Their goal is to collect private user information (e.g., location or sensing data). Using this private information, attackers can determine the user identities and learn their activities and behaviors. Our goal is to ensure that users cannot read the raw data reported by other users. The SSI must not be able to read the user raw data. Also, the SSI must not be able to infer any additional location in-

formation about the participants more than it already knows or could be inferred from the aggregate result. Hence, the scope of PAMPAS is to fully protect the raw data and the aggregation process, and does not consider the privacy exposure risks that arise from analysing the aggregate results, which is a complementary aspect of this work.

Even though the users are untrusted, we assume that all the SPs are trusted, which is reasonable considering that the tamper-resistance of the MCU provides a high level of protection against physical and side-channel attacks, and in particular for the data residing in RAM since the RAM memory is located inside the MCU. We also assume that the hardware manufacturer is trusted and protects the secrets embedded in SPs. In addition, all the persistently stored data in the NAND Flash is cryptographically protected.

Furthermore, we assume an honest-but-curious threat model, i.e., the SSI obeys the protocol it is supposed to do, but may try to infer anything it can from the data or behaviors is sees. Considering a malicious SSI (i.e., the server tampers with the protocol, e.g., by dropping messages to infer more information) is of little interest, since a malicious SSI can be easily detected (e.g., the SPs that aggregate the data verify if their own samples are present in the data sent by the server) leading to critical financial/legal consequences for the service.

Finally, we assume that the communication between SPs and SSI is anonymous, e.g., by using a proxy forwarder or an anonymization network (e.g., Tor) We assume such systems are able to hide the packet origin from an adversary, so that privacy cannot be compromised by a malicious server searching to recognize the origin of the uploaded messages. Let us emphasize that IP anonymity is not enough to protect the user privacy in MPSS because identity information could be determined from the location and sensing data.

3.3 Data and Computation Models

Data model. PAMPAS is designed to be generic with respect to the type of computation required by participatory sensing applications. In most cases, such applications require the aggregation of geo-localized and time-stamped sensed values collected by the sensing devices of the participants. Therefore, a participant's device periodically generates an update in the form *sample* = (*location, time, value*), which is encrypted and sent to the SSI. PAMPAS does not impose any restriction on the generation frequency of samples, which may depend on the application sample generation policy. However, the system should be efficient and scalable for a large number of participants and a high generation frequency of samples. Also, the participants' privacy should be fully protected independent of the number and spatiotemporal distribution of the samples. Furthermore, PAMPAS considers two types of locations corresponding to the two typical types of movements of users: (i) free movements in the two-dimensional space, i.e., *location* = (*x, y*); (ii) movements constrained by a transportation network (e.g., road or railroad network), i.e., *location* = (*rid, pos*), where *rid* is the road identifier and *pos* is the relative position on the road. Finally, the *value* corresponds to the sensed measure (e.g., traffic speed, noise level, etc.).

Query model. Given a stream of *samples* and an aggregate function, PAMPAS produces a spatiotemporal aggregation of the sample stream such as the stream-SQL-like [13] query formulation in Figure 3. The aggregation is temporal since the result is computed *continuously* over time as long as it is required or whenever the number of participants exceeds some predefined threshold. In this way, the spatiotemporal evolution of the measure of interest is monitored over time. To this end, PAMPAS divides the stream using a sliding time window (see Figure 3) and computes an aggregate result based on all the samples generated in the time window. The final aggregation result is a spatial function representing the evolution in space of the observed measure in the respective time window. For instance, the result can be the noise heat-map in the covering area of a city or the average travel time in a road network. As with the duration of observation, we do not impose any restrictions regarding the extent of the observed space.

```
SELECT SpatialAggregate (value), [COUNT(*)]
FROM ParticipatorySensingStream
      (WINDOW x seconds SLIDE x seconds)
[WHERE condition]
GROUP BY spatialUnit
[HAVING predicateOnSpatialAggregate]
```

Figure 3: Spatial-temporal aggregates in PAMPAS

Spatial units. As shown in the above query, spatial aggregates are based on a discrete referential space, i.e., a finite set of spatial units. Without loss of generality, we consider two types of referential spaces corresponding to the two types of users' movements. In the case of free movement, we consider a uniform grid and each grid cell corresponds to a *spatial unit*. The size of the units is determined based on the application requirements, space size, number of participants, etc. In the case of constrained movement by a transportation network, we consider that a *spatial unit* corresponds to a network (road) segment, i.e., the network path connecting two adjacent network nodes (e.g., the road segment between two intersections). In both cases, the number of *spatial units* can be large (e.g., hundreds of thousands). The COUNT in the query model is optional and is required in the aggregation protocol to check the probes partitioning.

Aggregate functions. PAMPAS can compute most types of aggregate statistics required by participatory sensing applications. Practically, our system can compute in real-time any type of function having reasonable time and space complexity given the relative low CPU power and little RAM of the SP. For illustrative purpose, we consider three classes of functions in this paper: (i) *Typical algebraic functions*: count, sum, average, standard deviation. Such aggregate functions are the most popular in the works related to participatory sensing [9], [5], [16]. These functions allow for example to compute the average travel time or the traffic density in a road network; (ii) *Specific functions*: inverse distance weighting (IDW). For instance, an application monitoring the noise pollution in the city could use the IDW function to compute a heat-map of the noise level in the entire space [14]; (iii) *Holistic functions*: median, percentile, top-k. Such functions are also frequently used in statistical computations. Their particularity is that the computation of the result requires accessing the entire sample set and cannot be achieved incrementally by accessing only subsets of samples as with the previous two classes of functions.

An important observation is that cryptographic solutions based on homomorphic encryption cannot be applied for spe-

Table 1: Notations used in the algorithms

Notation	Description
G_i	Identifier of probe group i
E_k	Symmetric deterministic encryption
nE_k	Symmetric non-deterministic encryption
E_k^{-1}	Symmetric deterministic decryption
nE_k^{-1}	Symmetric non-deterministic decryption
$P_{G_i}^{fake}$	Probability to send a fake message in group i
N	Number of spatial units
N_G	Number of probe groups
QI_{comm}	Degradation factor of the communication time
QI_{comp}	Degradation factor of the computation time
$Comp\_time_i$	Computation time for the group i

cific or holistic functions (see Section 2). Also, the holistic functions cannot be computed efficiently in a distributed architecture by the secure protocol proposed in [19] (as shown in Section 7).

3.4 Protocol Requirements

In the light of the above description of the proposed user-centric architecture, the PAMPAS protocols have to deal with the following challenges: (i) *Privacy*: By keeping all the sensitive data in the SPs, the adopted user-centric architecture matches this requirement in contrast with a server-centric architecture. In short, the computation protocol should not reveal to the SSI any additional information about the participants' paths besides what the SSI can infer from the aggregate result. (ii) *Generality*: the protocols should be able to compute any type of function over the spatiotemporal sensed measures by the mobile users and covering a large observation space. This is different from the works based on cryptographic approaches in which, typically, only basic computation (e.g., simple aggregates like sum, average) can be achieved and only in specific locations over limited periods of time. (iii) *Efficiency*: the protocols should be highly efficient to be able to *continuously* compute the aggregate results in *real-time* with very *limited resources*. Indeed, for economic and security reasons, the SPs used for data processing have low resources and limited availability. Hence, it is necessary to minimize the computation and communication costs of the PAMPAS protocols. (iv) *Scalability*: the protocols should allow PAMPAS to scale to a large number of participants (e.g., up to millions of users), high sampling frequencies, and very large regions. (v) *Accuracy*: PAMPAS should continuously reflect the sensed measures with good precision. In other words, protecting the users' privacy should not impact the accuracy of the aggregate result computed by the protocols.

4. GLOBAL AGGREGATION PROTOCOL

The global privacy-preserving protocol in PAMPAS consists in three phases that are repeatedly executed in pipeline (see Figure 4). First, the SSI collects all the sensing updates sent by the participants for a period equal to the sliding time window (i.e., the *collection period*). Each update is encrypted using symmetric non-deterministic encryption so

Algorithm 1: PAMPAS Protocol at the SSI-side

```
1  collection_period():
2      /* Receive encrypted updates from SPs */
3      while (true) do
4          message = (E_k(G_i), nE_k(sample))
5          store(message) →
           list[E_k(G_i)][currentTimeWindow]
6  processing_period():
7      foreach i in {E_k(G_i)} do
8          /* feed in parallel the randomly selected SPs */
9          randomly select SP_i ∈ E_k(G_i)
10         while
           message ← list[E_k(G_i)][lastTimeWindow] do
11             send(message, SP_i)
12     foreach i in {E_k(G_i)} do
13         /*Receive the final results from worker SPs*/
14         enc_result_i^final = (E_k(G_i), nE_k(result))
15 delivery_period():
16     foreach i in {E_k(G_i)} do
17         /*Push result_i to all requesting SPs*/
18         multicast(enc_result_i^final, {SP_k})
```

that the SSI cannot gain any knowledge from these updates. All the SPs share a secret key, which is renewed periodically to increase security. The key is generated randomly by a randomly chosen SP. To distribute the secrete key, we assume the users authenticate using a typical PKI infrastructure, i.e., a certificate is embedded in each user secure hardware. Whenever a new SP connects to the system, it authenticates using its certificate. Then, the SP randomly contacts another connected SP, which sends back the current shared secret key encrypted with the public key of this newly connected SP.

The shared secret key is used by the SPs to symmetrically encrypt the update messages (e.g., by using AES encryption) so that any SP can decrypt messages and aggregate the data. Note that, although an SP can decrypt the updates, a user is not allowed to access the decrypted data in her SP and that the tamper-resistant hardware protects the transiting data from the user. Therefore, as for the SSI, the users have access only to the final results and not to the raw data.

At the end of the collection period, the SSI triggers the *processing period*. In this phase, only a small subset of SPs, which are randomly selected by the SSI, are involved. The SSI partitions the collected samples such that the number of updates in a partition can fit the RAM resources of an SP (otherwise, the persistent Flash storage of the SP has to be used incurring a much higher computation cost). Then, each sample partition is sent to an SP, which computes a partial aggregate result for the received updates. The encrypted results are sent back to the SSI. Finally, the *delivery period* consists of delivering the current partial aggregate results to the queriers. Each querier needs to perform the final aggregation of these partial results, which is merely a concatenation of the demanded partial results.

Algorithms 1 and 2 give the detailed description of the operations executed by the SSI and the SPs respectively. In the following, we denote by E_k and nE_k the symmetric

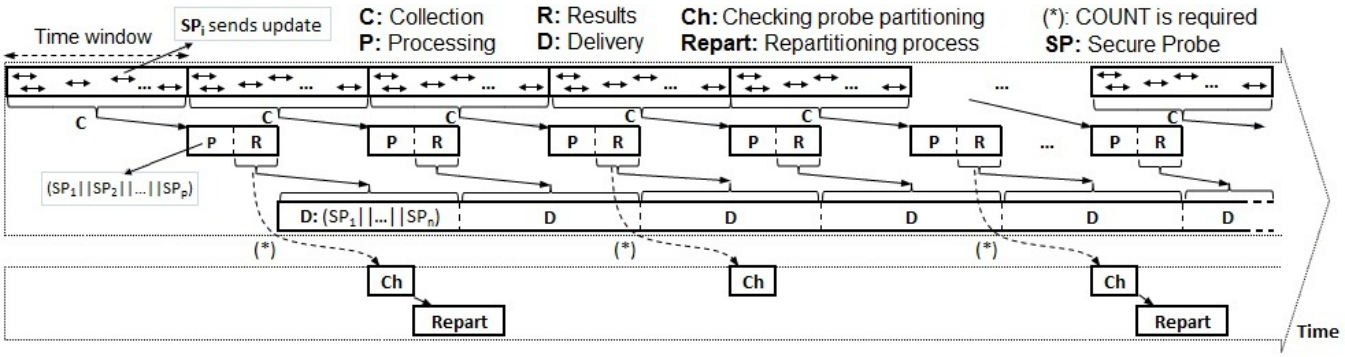


Figure 4: Workflow representation of the global protocol in PAMPAS

Algorithm 2: PAMPAS Protocol at the SP-side

1 ***collection_period()*: /* for all SPs */**
2 /* Generate and send the sensing update: $update(G_i, sample)$ */
3 $message = (E_k(G_i), nE_k(sample))$
4 $send(message, SSI)$
5 /* Send a fake sample to the SSI with probability $P_{G_i}^{fake}$ */
6 **if** $rand(0, 100) >= P_{G_i}^{fake}$ **then**
7 $fake\_message = (E_k(G_i), nE_k(fake\_sample))$
8 $send(fake\_message, SSI)$
9 ***processing_period()*: /* only for the selected SPs, one for each G_i */**
10 **while** *message = receive(SSI)* **do**
11 $sample = nE_k^{-1}(message)$
12 $result = result \oplus sample$
13 $enc\_result_i^{final} = (E_k(G_i), nE_k(result))$
14 $send(enc\_result_i^{final}, SSI)$
15 ***delivery_period()*: /* for all SPs */**
16 /* Pull the results for required $\{G_i\}$ from the SSI */
17 **foreach** *i in* $\{E_k(G_i)\}$ **do**
18 $send\_request(E_k(G_i), SSI)$
19 $result_i^{final} = nE_k^{-1}(receive(SSI))$

deterministic and non-deterministic encryption with the key k, and by E_k^{-1} and nE_k^{-1} the opposite decryption operations while G_i indicates the identifier of group i. All the notations used in the algorithms are listed in Table 1.

To address the performance limitations of the existing protocols [19] (see Section 2), the aggregation protocol in PAMPAS groups the participants based on their location, which permits processing together the generated samples in a group by a single SP. To this end, the users also send the deterministically encrypted value of the spatial unit they are currently located in, in addition to the non-deterministically encrypted value of the sample, i.e., $message = (E_k(groupID), nE_k(sample))$ (Algorithm 1, lines 4-5 and Algorithm 2, lines 3-4). Consequently, the SSI can group the messages based on the encrypted unit number and then send each group of samples to a different SP for aggregation (lines 7-11 in Algorithm 1 and lines 10-12 in Algorithm 2). By doing so, the advantage is manifold. First, the processing period is guaranteed to terminate in a single iteration, since each involved SP produces directly the aggregation result corresponding to a spatial unit. This greatly improves both the computation and the communication cost of the aggregation process. Second, data processing by an SP is also efficient since only one aggregate is computed, which greatly reduces the RAM requirements and avoids/reduces the usage of the persistent storage. Third, the final aggregate result is also partitioned and the queriers can demand the results only for specific spatial units, which further improves the communication cost. Furthermore, in order to avoid leaking information regarding the spatial distribution of users, the SPs also generate and send fake messages to the SSI (see Algorithm 2, lines 6-8). The rational and detailed explanation for this technique are discussed in the next section.

However, despite all these benefits, the above approach has one fundamental shortcoming originating from the skewed spatial distribution of the participants. Although the exact location of the updates and the unit ID are hidden, the SSI knows the number of participants in each spatial unit. If the SSI has access to side information about the spatial distribution of the users (e.g., global traffic density information), it may use this information to infer the (approximate) location of the participants and compromise their privacy.

5. PROBE PARTITIONING PROTOCOL

To counter the privacy threats that are rooted in the skewed spatial distribution of the participants, PAMPAS continuously partitions the set of probes based on their current location and the spatial units of the query. Similar to the global aggregation protocol, this privacy-aware partitioning protocol is executed by SPs. The idea is to group SPs located in adjacent spatial units such that the resulted probe groups have approximately the same size. Therefore, in PAMPAS a group G_i *covers several spatial units (as defined in Section 3.3) and includes all the SPs in these units.*

The probe partitioning has to be recomputed periodically to keep the groups balanced since the users' distribution in space changes over time. Moreover, the groups should contain users located in closely situated spatial units to maximize the lifetime of a partitioning. The challenge is to implement a partitioning algorithm that can be executed periodically at SPs because the typical spatial partitioning algorithms are much too costly to be considered in our context (i.e., limited-resources SPs).

Our algorithm is based on the following idea. We use a

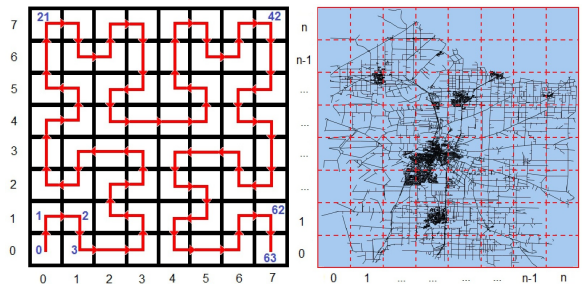


Figure 5: Hilbert indexing of spatial units

space-filling curve to index the spatial units of the application query. A space-filling curve has the property to map a multidimensional space to a one-dimensional space such that, for two objects that are close in the original space, there is a high probability that they will be close in the mapped target space. Then, we sort the spatial units on the space-filing curve index. Once sorted, an approximate balanced grouping can be checked and computed in $O(N_G)$ space complexity and $O(N)$ time complexity, where N_G is the number of probe groups, and N is the number of spatial units.

Indexing the spatial units. In our system, we use Hilbert curves, but other types of space-filling curves can be used as well to index the spatial units considered by the participatory sensing application (e.g., z-curves). In the case of free movement, the indexing is straightforward since the space is already partitioned with a uniform grid (see Figure 5 left). Then, we cover the grid cells with the Hilbert curve corresponding to the grid granularity and label each cell with the obtained Hilbert index. In the case of constrained movement, the indexing requires two steps. First, we cover the transportation network with a uniform grid (see Figure 5 right). The grid granularity is chosen such that the number of network segments (see Section 3.3) intersecting with a grid cell is low for most of the cells. Then, the grid cells are indexed with a Hilbert curve and each network segment is labeled with the Hilbert index of the cell containing the segment center. In case several segments are contained by a cell, the segments are sorted by the x-coordinate and the y-coordinate of their centers and labeled accordingly. Once the spatial units are indexed, they are sorted on the index value and the sorted unit vector is broadcasted to all the participants to be used in the probe partitioning phase.

Checking and repartitioning the probe grouping. Periodically, our system verifies if the current probes partitioning is still balanced with respect to the number of probes in each group. The verification frequency depends on the dynamicity of users in space. In PAMPAS, the checking and repartitioning processes can be executed often (i.e., every few seconds) due to their low cost. When a partitioning checking is triggered, the system computes a *count* aggregate in addition to the application aggregate function (see Figure 3), which gives the actual number of users (SPs) in all the spatial units. The count aggregate result is then pushed to an SP randomly chosen by the SSI. The checker SP decrypts the results and updates the weights[5] of the sorted spatial unit vector (lines 4-7 in Algorithm 3). This operation has $O(N)$ complexity assuming that a small index containing the partitions frontiers is kept in memory by the SP (which requires only N_G Flash addresses to be kept in RAM). At the same time, the SP computes in memory the count by group (since the groups are sent one by one by the SSI, line 8 in Algorithm 3) and compares the counts. If the balancing of the current probes partitioning is within the predefined limits, the checker SP sends the current values to all the other SPs (i.e., exchanged encrypted through SSI), which update the weights of the spatial units with the new count values. Otherwise, the checker SP computes a new partitioning.

[5]The weight is the number of probes in a spatial unit.

Algorithm 3: Checking probe partitioning (SP-side)

```
1  check_probe_partitioning():/* one randomly
   selected SP */
2      /* Pull all the results from the SSI */
3      foreach i in {E_k(G_i)} do
4          send_request(E_k(G_i), SSI)
5          enc_result_i^final = receive(SSI)
6          allCounts[G_i][] ← E_k^-1(enc_result_i^final)
7          update localy stored counts for spatial units
           /* also required to compute the probability to
           generate fake samples */
8          compute weights[G_i] = SUM(allCounts[G_i][])
9      compute standard_deviation(weights)
10     if standard_deviation(weights) < threshold then
11         send_for_broadcast(nE_k(allCounts), SSI)
12     else
13         execute probes_repartitioning()
```

Once the sorted vector of spatial units is updated with the new weight values, the probe repartitioning can be efficiently computed in $O(N)$ and $O(N_G)$ time and space complexity respectively (see Algorithm 4). To set the partition borders we use a greedy algorithm, which adds spatial units to a group as long as the total weight of the group is lower than a threshold value (lines 12-16 in Algorithm 4). The threshold is computed as the ratio between the total number of probes and the number of groups (line 10 in Algorithm 4), and represents the average number of users per group. The partitioning result is a list of N_G *milestones* indicating the group borders in the sorted index of spatial units (line 15 in Algorithm 4). The result is then encrypted and delivered, through SSI, to all users, which update their partitioning data and generate new samples accordingly starting from the next computation window.

The proposed probes partitioning algorithm has low complexity and can be efficiently executed even with the low resources of an SP. However, the partitioning algorithm cannot guarantee a certain degree of balancing of the partition weights. Yet, the partitioning balancing is required to avoid leaking any information regarding the spatial distribution of users. To deal with this problem, the SPs generate fake samples in all the probe groups having a number of users lower than the maximum size group. Therefore, in the collection period of each computing time window, an SP sends probabilistically a dummy sample in addition to the real sample. The probability to send a fake sample is proportional to the difference between the maximum size group and the number of users in the SP's group, and inversely proportional to the

Algorithm 4: Repartitioning process (SP-side)

```
1  PROBE_REPARTITIONING():/* one randomly selected SP */
2      compute QI_comp and QI_comm for current N_G
3      while true do
4          /* adjust the number of groups N_G */
5          if QI_comp > QI_comm then
6              tN_G = 2 * N_G
7          else
8              tN_G = N_G/2
9          /* repartition for tN_G */
10         avgGroupWeight = SUM(allCounts[])/tN_G
11         currentGroupWeight = 0
12         for i = 0 to N − 1 do
13             currentGroupWeight+ = allCounts[i]
14             if currentGroupWeight ≈ avgGroupWeight
               then
15                 newPartitionMilestones[].add(i)
16                 currentGroupWeight = 0
17         /* check if the new partitioning for tN_G is
           better than for N_G*/
18         compute tQI_comp and tQI_comm for tN_G
19         if tQI_comp + tQI_comm < QI_comp + QI_comm
           then
20             N_G = tN_G; QI_comp = tQI_comp
21             QI_comm = tQI_comm
22         else
23             break
24     message = allCounts[]||newPartitionMilestones[]
25     send_for_broadcast(nE_k(message), SSI)
```

number of users in the group (see Algorithm 2, lines 6-8). The same approach is used to hide the number of spatial units in each group. At the end of the aggregation phase, each aggregating SP adds to the result a number of fake values equal to the difference between the maximum number of units in the groups and the number of units in the current group. In this way, all the partial aggregate results received by the SSI have the same size and the SSI cannot infer the number of cells in any group. Note that the fake values are filtered out by the worker or querier SPs and therefore have no impact on the accuracy of the results.

$$QI_{comp} = Max_{i=1,N_G}[Comp\_time_i] - Max_{j=1,N}[Comp\_time_j] \quad (1)$$

$$QI_{comm} = \frac{size(sample)}{bandwidth}\sum_{i=1}^{N_G}\{Max_{j=1,N_G}[Count_j(probes)] - Count_i(probes)\} + \frac{size(sample)}{bandwidth}\sum_{i=1}^{N_G}\{Max_{j=1,N_G}[Count_j(spatialUnits)] - Count_i(spatialUnits)\} \quad (2)$$

Choosing the Number of Probe Groups. The cost of the aggregation protocol is composed of the computation cost at the SP side and the communication cost between the SSI and the SP. The number of probe groups impacts both the computation and the communication costs. Specifically, the computation cost decreases with the increase in the number of groups and attains the minimum value when the number of groups is equal to the number of spatial cells, i.e., an SP is used to aggregate the samples for each spatial unit. But increasing the number of groups leads to a higher imbalance in the groups' weights, which in turn requires injecting more fake samples and enlarges the communication cost. Therefore, modifying the number of groups has opposite effect on the computation and the communication cost.

PAMPAS computes two quality indicators to measure the impact of the number of groups on the computation and communication costs, i.e., QI_{comp} and QI_{comm}, as defined by Formulas (1) and (2). QI_{comp} estimates the degradation of the computation time at the SP side generated by the fact that several spatial cell aggregates are delegated to one SP instead of using one SP for each cell. Estimating the computation time is fairly simple since the time is typically linear with the number of samples to be processed by the SP, assuming that the aggregation can be entirely processed in RAM. However, the cost model can be extended to the case in which it is required to access the secondary storage. QI_{comm} estimates the degradation of the communication cost caused by the imbalance of the group weights. The first term indicates the overhead incurred by the fake samples generated to balance the groups, while the second term measures the overhead of generating fake results to balance the number of aggregate results in each group.

Each time an SP computes the probe partitioning, it also computes the values of QI_{comp} and QI_{comm} (line 2 in Algorithm 4). If $QI_{comp} > QI_{comm}$, the SP multiplies by two the number of groups and re-partitions the probes. If $QI_{comp} < QI_{comm}$ the SP divides by two the number of groups and repartitions the probes (lines 5-8 in Algorithm 4). The SP continues to adjust the number of groups until $QI_{comp}+QI_{comm}$ has minimum value (lines 19-23 in Algorithm 4), meaning that the aggregation cost is near optimal. Thus, this process allows adapting the number of groups to the number and the spatial distribution of the probes.

6. SECURITY AND PRIVACY ANALYSIS

6.1 Security Analysis

The users cannot read the raw data of other users because the data stored in memory is protected by the secure MCU (i.e., the RAM is located inside the MCU) and the data stored in NAND Flash are cryptographically protected.

The SSI does not have the encryption key, so it cannot access the transiting data. In addition, the non-deterministic encryption protects the data against frequency-based attacks. The SSI may also try to buy an SP and pass for a user to gain access to the shared encryption key. However, this would be useless since the tamper-resistance of an SP protects the key. The SSI could collude with a querier, but it will gain access only to the aggregate result. Finally, since the samples are communicated through anonymizers, the SSI cannot identify the senders or link consecutive messages from the same user.

The SSI could try to infer information from the deterministically encrypted group ID values. Nevertheless, the SSI cannot perform a frequency-based attack using the en-

crypted group ID, since all the groups contain approximately the same number of messages. Therefore, the SSI cannot infer the corresponding (approximate) location of a group or the topological neighborhood of the groups (which would be the first step to attack the users' privacy). Hence, the only knowledge the SSI acquires is the number of groups and its evolution over time, which does not endanger the users' privacy. Note that even if the SSI has somehow access to the full partitioning information and the corresponding encrypted ID, a user is still hidden under the corresponding group area and within the crowd in the same group (let us recall that the messages are sent anonymously so it is hard for the server to link the messages coming from the same user). Hence, the protocols are secure and fully protect the privacy of the users.

Although, protecting the privacy of users beyond the aggregate results is out of the scope of this paper (as discussed in Section 3.2), one can easily integrate basic protection mechanisms in PAMPAS for such cases. For example, to avoid the risk of exposure for the users situated in very sparse areas (e.g., a single user or very few users located in a spatial unit), we can simply add a predicate in the HAVING clause of the aggregate query (see Figure 3) indicating the minimum number of users in a spatial unit. In this way, the sparse spatial units are eliminated from the aggregate results. Another solution is to increase the size of the sliding window, or of the spatial units accordingly.

6.2 Privacy Analysis

To underline the high level of privacy protection of PAMPAS w.r.t. the SSI, we consider an entropy-based metric and apply it in the context of two scenarios that are related to our architecture. We then compare the privacy leakage in these two scenarios with the privacy leakage in PAMPAS.

Entropy is a popular metric to describe location privacy in general [6], and it is also appropriate to describe the privacy-preserving mechanism of PAMPAS. Commonly, entropy is used to quantify the average degree of uncertainty associated with a set of events. In the case of location privacy, the idea is to prevent user identification by obfuscating her exact location in a spatial region containing a certain number of individuals. Therefore, the level of privacy is directly related to the popularity (i.e., number of individual footprints) of the region. This means the higher the popularity, the higher the privacy level of the users in that region. Then, entropy is used to quantify the degree of popularity of a region. Formally, let reg be a spatial region and let $U(reg) = \{u_1, u_2, \dots u_p\}$ be the set of users in region reg. Let f_i (with $1 \leq i \leq p$) be the number of sample updates (i.e., footprints) that user u_i sends from reg and $F = \sum_{i=1}^{p} f_i$ be the total number of sample updates sent from reg.

Definition 1. Entropy of a region: the entropy of region reg is defined as: $E(reg) = -\sum_{i=1}^{p} \frac{f_i}{F}.log\frac{f_i}{F}$

Definition 2. Popularity of a region: the popularity of region reg is defined as: $Pop(reg) = 2^{E(reg)}$

Definition 3. Privacy leakage: the privacy leakage for each $update_k$ sent by user u_i is defined as:

$$priv\_leak_{u_i}(update_k) = \frac{1}{Pop(location\ of\ update_k)}$$

To compute the privacy leakage in different cases, we consider a simple numerical example inspired by the datasets used in our experimental evaluation (see Section 7.1). Let us consider that 200 thousand users participate in a mobile sensing application that aggregates data over 20 thousand spatial units (e.g., road segments in a road network). To keep the formulas tractable (but without loss of generality), let us consider that each user produces 50 samples from 50 distinct spatial units. This implies that on average, there are 500 footprints (i.e., updates) in each spatial unit.

Scenario 1: there is no grouping of the probes. Each participant sends the non-deterministically encrypted value of a sample together with the deterministically encrypted value of the spatial unit identifier to allow an efficient aggregation of the data. However, no fake sample is inserted by the probes. Although the spatial unit identifiers are encrypted, the SSI could easily determine the location of the spatial units if it has access to the global spatial distribution of the probes (i.e., a frequency-based attack). In this case, the average entropy of a spatial unit by applying Definition 1 is $E(s.unit) = -\sum_{i=1}^{500} \frac{1}{500}.log\frac{1}{500} = log(500)$, which gives a popularity of $Pop(s.unit) = 500$ and an average privacy leakage of $priv\_leak = 0.002$ for each update. Clearly, the privacy leakage can be lower or higher for each spatial unit depending on the popularity value compared with the average value.

Scenario 2: there is a static partitioning of probes, i.e., the spatial units are statically partitioned into a number of groups containing closely located spatial units. As in the previous case, the probes send the deterministically encrypted value of the spatial group and are also exposed to a frequency-based attack from the SSI. However, grouping many spatial units leads to decreasing the privacy leakage risk (but at the cost of increased aggregation time). For instance, partitioning the spatial units in 200 groups (i.e., 100 spatial units per group), leads to an average popularity $Pop(group) = 10^3$ and thus an average privacy leakage $priv\_leak(update) = 10^{-3}$, which is smaller than in the previous scenario. Also, the obfuscation region is much larger since it corresponds to 100 spatial units instead of one.

PAMPAS goes even further in the protection of the participants' privacy by using a dynamic partitioning of the probes based on their location and spatial distribution. The adaptive partitioning produces nearly balanced groups of probes. In addition, the eventual imbalance of the groups is corrected by injecting fake tuples, which precludes the SSI doing frequency-based attacks. This means that it is extremely difficult for the SSI to estimate even the approximate corresponding area of each group. Therefore, in our case, the entropy applies indistinguishably to all the participants leading the a popularity $Pop(group) = 2 \cdot 10^6$ and an average privacy leakage $priv\_leak(update) = 5 \cdot 10^{-7}$. Practically, in PAMPAS, the privacy leakage depends only on the total number of participants, while the obfuscation area corresponds to the entire observation space. Besides, the number of groups is adaptively chosen such as to minimize the aggregation cost.

7. EXPERIMENTAL EVALUATION

The goals of our experimental evaluation are twofold: (i) compare the execution time and scalability of PAMPAS with those of a state-of-the-art protocol described in [19]; (ii) quantify the cost and scalability of our partitioning pro-

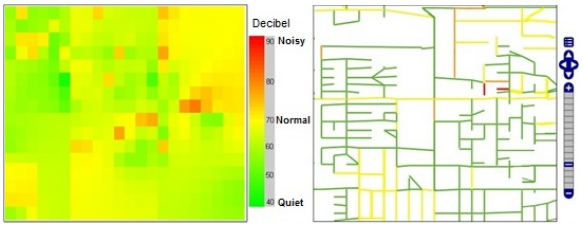


Figure 6: Aggregation maps for two applications: noise monitoring (left) and traffic monitoring (right)

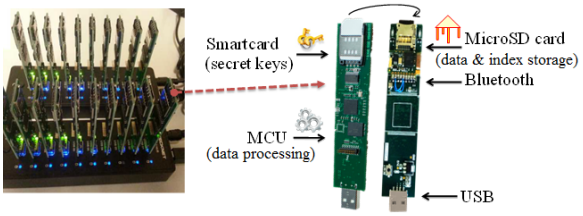


Figure 7: Secure tokens

tocol. We implemented and validated PAMPAS through emulations using secure tokens which have a hardware configuration representative for secure hardware platforms. As applications for our experiments, we used traffic monitoring and noise monitoring with both real and synthetic datasets representing small and medium-size cities. Figure 6 illustrates our graphic interface for these applications; it shows the aggregate results for the noise heat-map and the average travel time for the road network. A demo of our prototype was presented in [20] using a traffic monitoring scenario.

7.1 Experimental Setting

Hardware platform. In our experimental evaluation, the SSI is hosted on a PC (3.1 GHz i5-2400, 8GB RAM, running Windows 7) which also displays the aggregate results in a graphical form for validation purpose. The SPs are implemented by representative secure hardware devices (see Figure 7) which includes an MCU with a 32-bit RISC CPU at 120MHz, a cryptographic co-processor implementing AES and SHA, 128KB of static RAM and 1MB of NOR Flash to store the software stack, a smartcard chip hosting the cryptographic credentials (i.e., the secrete encryption keys) and an SD card reader allowing for a large storage capacity. We used a commodity SD card (Samsung SDHC Essential Class 10 of 32GB) as secondary Flash storage. The SSI in our testing system manages a multi-channel Ethernet connection with a global bandwidth of 100Mbps. Importantly, on the SP's side, our implementation limits the RAM consumption to only 30KB and the maximum communication bandwidth to 200Kbps to validate the proposed protocols with less powerful secure devices. Thus, in our experiments, all the SPs have this minimalist configuration. To emulate a very large number of SPs, we execute sequentially on an SP the aggregate computations and communications for all the worker SPs and measure the "parallel" execution time as the maximum aggregation time in the execution sequence.

Baseline system. To underline the importance of the PAMPAS protocols, we implemented the *secure protocol* proposed in [19] and took it as the baseline. This protocol can be applied without modification to aggregate the samples collected in each time window. Since PAMPAS offers the same level of security and privacy as the baseline protocol, our experimental evaluation focuses on the efficiency part. Note that in [19], two more protocols are proposed that are even more expensive than the secure protocol if considered in our context.

Datasets and aggregate functions. We use both synthetic and real datasets to test the efficiency and scalability of PAMPAS. We employed the well-known Brinkhoff generator [4] to generate mobility traces on two real road networks of the cities of Oldenburg (Germany) and Stockton (San Joaquin County, CA). Oldenburg is a small size network having 7035 road segments, while Stockton is a medium size road network having 24123 segments. Depending on the network size, we generated traces corresponding to a medium and large number of users. With Oldenburg, the medium and large datasets contain 47 thousand and 270 thousand users respectively. With Stockton, the medium and large datasets contain 200 thousand and 1.35 million users respectively. The spatial distribution of the traces follows the network spatial density. Compared to the existing real datasets, the synthetic datasets have the prominent advantage of having excellent spatial and temporal coverage. However, it is also important to validate the proposed protocol with real datasets. To this end, we used the T-Drive Taxi trajectory dataset [21]. This dataset contains around 15 million trajectory units collected from 10357 taxis over a period from Feb. 2 to Feb. 8, 2008 in Beijing. Because the density of taxis is too low compared to the synthetic dataset, we extracted and merged a period of one hour in our tests, in order to generate a dataset containing 191 thousand trajectories covering 32800 road segments.

To show the generality of PAMPAS, we selected three aggregate functions, i.e., *average*, *IDW* [14] and *median*, corresponding to the three aggregate types described in Section 3.3. We associate the average and median aggregates with the traffic monitoring application, i.e., compute the average travel time and the median speed for each road segment in a road network. Hence, these two scenarios consider the constrained movement type. The IDW aggregate is associated to the noise-level monitoring application and a free movement type. In this case, we use the same generated mobility traces, but consider them in the 2D space instead of the network space. Also, we use a 64x64 grid to divide the observed 2D space into 4096 spatial units for the free movement scenario. The speed sample values are directly generated by the moving objects generator, while the noise values are generated by us proportionally to the number of probes in the spatial unit.

7.2 Performance Evaluation

Execution time. Figure 8 shows the aggregation time (in a logarithmic scale) for the three functions of both Baseline and PAMPAS protocols with 191 thousand probes in Beijing and with 200 thousand probes in Stockton. The aggregation time is global, i.e., it includes both the computation and communication time. The results indicate that PAMPAS is very efficient since it requires only a few seconds to compute the aggregate results for all the tested functions in both datasets. Also, the aggregation times of PAMPAS are similar between the real and the synthetic datasets. However, in both cases the baseline protocol is much more costly (especially for complex aggregate functions) leading

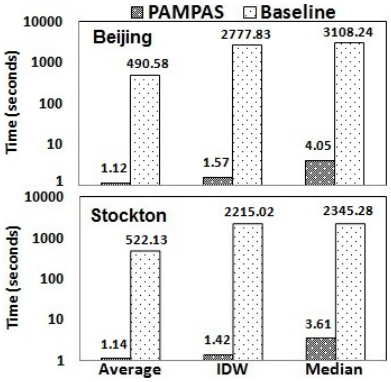


Figure 8: Aggregation time of PAMPAS and Baseline protocols in real dataset (top) and synthetic dataset (bottom)

to aggregation times up to three orders of magnitude higher than PAMPAS. Moreover, the aggregation times with the baseline protocol are larger for the Beijing dataset. The explanation is that the number of spatial units is larger in Beijing (i.e., 32800) than in Stockton (i.e., 24123). On the other hand, PAMPAS is scalable with respect to both the aggregation function and the number of spatial units in the query.

Scalability. We further test the scalability of the protocols with different number of probes, spatial units, and aggregate functions. Figure 9a shows the aggregation time for the two protocols for the average (top graph) and median (bottom graph) functions with medium and large number of users on both road networks. The results confirm that only PAMPAS is scalable w.r.t. all the varying input parameters. In the worst case, the computation time attains 14 seconds to compute the median speed for 1.35 million samples covering 24123 spatial units.

The baseline protocol does not scale with the number of samples and especially with the number of spatial units. Practically, the baseline can provide real-time aggregation only for a small number of spatial units (i.e., 7000 in Oldenburg) and basic aggregate functions (e.g., average). The very limited RAM of the SPs and the impossibility to efficiently parallelize the aggregate computation make the baseline inadequate for the requirements of participatory sensing applications.

Cost and scalability of partitioning protocol. Figure 9b (top) presents the partitioning computation time for both Oldenburg and Stockton networks. A new partitioning can be computed in a few seconds by an SP. This means that the checking and probes re-partitioning can be executed frequently, which allows PAMPAS to adapt to even fast changes in the spatial distribution of the probes. Most of the partitioning cost resides in reading and writing the partitioning data to the secondary Flash storage. This also explains the increase of the partitioning time with the number of partitions, since in this case the I/O operations are executed at a smaller granularity, which is more costly.

Figure 9b (bottom) indicates that the partitioning unbalance factor, i.e., the ratio between the maximum and the average partition size, increases with the number of partitions. The unbalance factor is an important indicator in PAMPAS since the higher the unbalance, the higher the number of fake injected samples and, therefore, the communication cost.

Figure 9c shows the impact of the number of partitions on the global aggregation time as well as on the computation and communication cost, which compose the total time. The computation time decreases with the increase of the number of partitions since the amount of work done by the aggregation SPs also decreases. Conversely, the communication time increases with more partitions since more fake samples are injected into the system as explained above. Globally, the near-optimal aggregation time is obtained with a number of partitions that minimizes the cumulated degradation of the computation and communication costs (see Section 5). We obtained similar results with the real dataset, for which the optimal number of partitions is 100 while the network partitioning is computed in just 2 seconds. The aggregation costs are partially shown in Figure 8 (top). Given the space limitation and the similarity of the results with the synthetic datasets, we omit here the details of the results with the real dataset.

Discussion. It is worth mentioning that the aggregation time can be greatly improved by increasing the processing power and the communication bandwidth of the SSI. For example, increasing the server bandwidth from 100Mbps to 1 GBps, makes the maximum aggregation time (i.e., median function with the large Stockton dataset) to drop from 14 seconds to less than 7 seconds. Also, in some scenarios, pushing the computation in the user devices may be problematic (e.g., battery powered devices, concurrent applications running in the device). However, PAMPAS minimizes this type of problem thanks to its design and its high efficiency. For instance, in our tests, a user participating in the system for one hour, has a probability between 3.5% and 8.7% to participate once to an aggregate computation assuming that aggregates results are produced every 30 seconds, and a probability between 0.004% and 0.12% to do a repartitioning assuming that the probes partitioning is checked every 1 minute. In all cases, the computation is done in a few seconds at most and requires only modest resources. Moreover, the computation effort is inversely proportional to the probability to be picked.

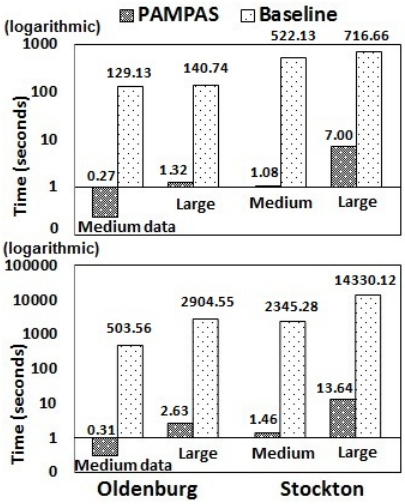
8. CONCLUSION

This paper proposes PAMPAS, a privacy-aware mobile participatory sensing system based on a distributed architecture and personal secure hardware. This combination allows PAMPAS to achieve the same level of privacy as cryptographic solutions without having to sacrifice generality, scalability, and accuracy. The proposed aggregation solution is, to the best of our knowledge, the first proposal of a distributed protocol that is secure, efficient, and scalable and that fits both the strict hardware constraints of secure personal devices and the real-time constraints of participatory sensing applications. The experimental evaluation based on representative hardware for secure platforms validates the proposed solution.

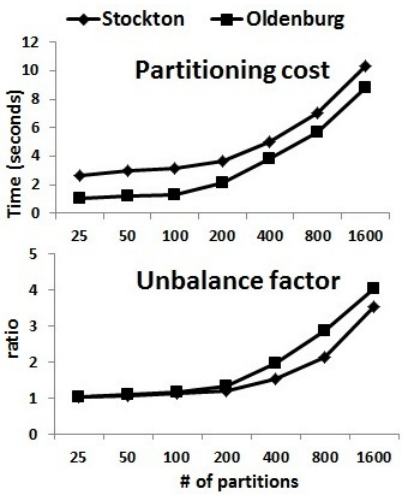
9. REFERENCES

[1] T. Allard, B. Nguyen, and P. Pucheral. METAP: revisiting privacy-preserving data publishing using secure devices. *Distributed and Parallel Databases*, 32(2):191–244, 2014.

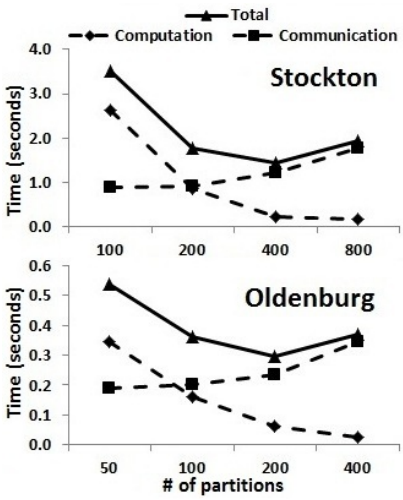
[2] ARM. *ARM Security Technology - Building a Secure System using TrustZone Technology.* ARM Technical White Paper, 2009.



(a) Scalability of the PAMPAS and Baseline protocols with Average function (top) and Median function (bottom)



(b) The partitioning costs (top) and the partitioning imbalance factor (bottom) with different number of partitions



(c) Communication and computation costs of Median function with different number of partitions

Figure 9: Performance evaluations

[3] A. Baumann, M. Peinado, and G. Hunt. Shielding applications from an untrusted cloud with haven. In *OSDI*, pages 267–283, 2014.

[4] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, June 2002.

[5] J. W. S. Brown, O. Ohrimenko, and R. Tamassia. Haze: privacy-preserving real-time traffic statistics. In *ACM SIGSPATIAL*, pages 540–543, 2013.

[6] M. L. Damiani. Location privacy models in mobile applications: conceptual view and research directions. *GeoInformatica*, 18(4):819–842, 2014.

[7] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3, 2013.

[8] E. D'Hondta, M. Stevens, and A. Jacobs. Participatory noise mapping works! an evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, 9(5):681–694, October 2013.

[9] G. Drosatos, P. S. Efraimidis, I. N. Athanasiadis, and M. Stevens. A privacy-preserving cloud computing system for creating participatory noise maps. In *COMPSAC*, pages 581–586, 2012.

[10] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli. Progress and challenges in intelligent vehicle area networks. *Magazine Communications of the ACM*, 55(2):90–100, 2012.

[11] H. Gao, C. H. Liu, W. Wang, J. Zhao, Z. Song, X. Su, J. Crowcroft, and K. K. Leung. A survey of incentive mechanisms for participatory sensing. *IEEE Comm. Surveys and Tutorials*, 17(2):918–943, 2015.

[12] B. Hoh, T. Iwuchukwu, Q. Jacobson, D. Work, A. M. Bayen, R. Herring, J. C. Herrera, M. Gruteser, M. Annavaram, and J. Ban. Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines. *IEEE Tran. on Mobile Computing*, 11(5):849–864, 2012.

[13] N. Jain, S. Mishra, A. Srinivasan, J. Gehrke, J. Widom, H. Balakrishnan, U. Çetintemel, M. Cherniack, R. Tibbetts, and S. B. Zdonik. Towards a streaming sql standard. In *PVLDB 1(2)*, pages 1379–1390, 2008.

[14] S. Nittel, J. C. Whittier, and Q. Liang. Real-time spatial interpolation of continuous phenomena using mobile sensor data streams. In *ACM SIGSPATIAL*, pages 530–533, 2012.

[15] M. Penza. Cost action TD1105: New sensing technologies for environmental sustainability in smart cities. In *IEEE SENSORS*, 2014.

[16] R. A. Popa, A. J. Blumberg, H. Balakrishnan, and F. H. Li. Privacy and accountability for location-based aggregate statistics. In *CCS*, pages 653–666, 2011.

[17] D. Quercia, I. Leontiadis, L. Mcnamara, C. Mascolo, and J. Crowcroft. Spotme if you can: Randomized responses for location obfuscation on mobile phones. In *ICDCS*, pages 363–372, 2011.

[18] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *ACM SenSys*, pages 85–98, 2009.

[19] Q.-C. To, B. Nguyen, and P. Pucheral. Privacy-preserving query execution using a decentralized architecture and tamper resistant hardware. In *EDBT*, pages 487–498, 2014.

[20] D.-H. Ton-That, I. Sandu-Popa, and K. Zeitouni. PPTM: Privacy-aware participatory traffic monitoring using mobile secure probes. In *IEEE MDM*, 2015. Demo paper.

[21] J. Yuan, Y. Zheng, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *SIGSPATIAL*, pages 99–108, 2010.