Mobile participatory sensing with strong privacy guarantees using secure probes

Iulian Sandu Popa, Dai Hai Ton That, Karine Zeitouni, Cristian Borcea

Mobile Participatory Sensing with Strong Privacy Guarantees Using Secure Probes

Iulian Sandu Popa · Dai Hai Ton That · Karine Zeitouni · Cristian Borcea

Abstract Mobile participatory sensing (MPS) could benefit many application domains. A major domain is smart transportation, with applications such as vehicular traffic monitoring, vehicle routing, or driving behavior analysis. However, MPS's success depends on finding a solution for querying large numbers of smart phones or vehicular systems, 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 MPS. 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. Moreover, an enhanced version of the protocol, named PAMPAS$^+$, makes the system robust even against advanced hardware attacks on the SPs. Hence, the risk of user location privacy leakage remains very low even for an attacker controlling the SSI and a few corrupted SPs. Our experimental results demonstrate that these

I. Sandu Popa
DAVID Laboratory - University of Versailles Saint-Quentin, Université Paris-Saclay, Versailles, France & INRIA Saclay-Ile-de-France, Université Paris-Saclay, Palaiseau, France
E-mail: iulian.sandu-popa@uvsq.fr

D. H. Ton That
College of Computing and Digital Media, DePaul University, Chicago, Illinois, USA
E-mail: dtonthat@depaul.edu

K. Zeitouni
DAVID Laboratory - University of Versailles Saint-Quentin, Université Paris-Saclay, Versailles, France
E-mail: karine.zeitouni@uvsq.fr

C. Borcea
Department of Computer Science, New Jersey Institute of Technology, Newark, New Jersey, USA
E-mail: borcea@njit.edu

protocols work efficiently on resource constrained SPs being able to collect the data, aggregate them, and share statistics or derive models in real-time.

Keywords Location privacy · secure protocol · distributed architecture · mobile · participatory sensing · real-time urban monitoring

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 [33], such as Citi-Sense in Oslo, CamMobSens at Cambridge, MetroSense at Dartmouth, OpenSense in Switzerland or Urban Civics in Paris[1]. Many applications that exploit the sensing features of smart phones are already available. Examples include community based traffic monitoring (e.g., Waze[2], or Navigon[3]), evaluating the quality of road infrastructures[4], finding available parking spaces or noise mapping [15]. As we can see from these examples, smart transportation applications such as vehicular traffic monitoring, vehicle routing, or driving behavior analysis, benefit greatly from mobile participatory sensing. In addition, the emerging lightweight low-cost sensors are changing the paradigm of environmental and health monitoring[5], 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 (e.g., their smart phones or vehicular systems) 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 [31]. 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 [30]. 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 represent a serious obstacle that can prevent a wide adoption of MPSS.

[1] https://goflow.ambientic.mobi/
[2] http://www.waze.com
[3] http://navigon.com
[4] http://www.streetbump.org/
[5] http://www.epa.gov/heasd/airsensortoolbox/

Several works consider the MPSS privacy problem such as [15], [6], [13], [37], [40]. However, most approaches require trusting a proxy server [13], [40], while others are too costly [15], [6], or sacrifice sensing accuracy for privacy [37]. 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], [41], [25] communicating with the user's smartphone or plugged inside it (e.g., Google Vault[6]), a tamper-resistant hardware security module securing the on-board computer of a vehicle [16], or the secure TrustZone CPU [3] of the ARM cortex-A series equipping most of mobile devices today, all such secure devices offer tangible, hardware-based security guarantees under the form of what is generally called a Trusted Execution Environment (TEE) [38]. We leverage their secure data processing capability in a distributed, privacy-by-design architecture, providing an alternative to the traditional server-centric architecture. Following current technology trends (e.g., Intel SGX [36], TrustZone CPU [21]), we expect that such secure devices will become ubiquitous in the near future, equipping by default users' mobile devices and devices embedded in smart vehicles. 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[7] [44] 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 [18], 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.

[6] http://www.cnet.com/news/googles-project-vault-is-a-security-chip-disguised-as-an-micro-sd-card/

[7] This paper is an extended version of [44]. The new material covers three significant contributions. First, we design a new, more robust aggregation protocol that is resilient to advanced hardware attacks. Second, we provide an alternative, more effective partitioning algorithm that offers a different tradeoff in terms of efficiency and partitioning quality than the base partitioning algorithm. Third, we provide a thorough analysis of the privacy protection and also an extensive evaluation of the new proposed protocols.

Furthermore, to increase the security of PAMPAS and make it robust against advanced hardware attacks (called lab attacks in the paper[8]), we design an enhanced aggregation protocol named PAMPAS$^+$. To this end, we consider first an extended threat model in which we take into account the possibility that some of the SPs could be corrupted. We then introduce the PAMPAS$^+$ protocol which, by isolating the risk, is able to avoid a complete leakage of sensitive user information even if some SPs are corrupted and thus to continue protecting the privacy of the participants. Hence, PAMPAS$^+$ drastically reduces the benefits of a lab attack and through this largely increases the system security. At the same time, the improved privacy protection of PAMPAS$^+$ has a minimal impact on the efficiency and scalability of the protocol as the most costly operations can be done offline.

We implemented and validated PAMPAS/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 [41]. 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 base 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. Building up on the foundation laid by the PAMPAS protocol, we present PAMPAS$^+$ in Section 6, an efficient privacy improvement of PAMPAS which addresses the important lab attack problem. The security analysis of PAMPAS and PAMPAS$^+$ is presented in Section 7, while the experimental results as well as the performance comparison of PAMPAS and PAMPAS$^+$ are shown in Section 8. Finally, we conclude the paper in Section 9.

2 Related Work

Traditional system architectures used in MPSS such as [13], [40] 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

[8] We use the terminology of ARM [3] which designates as lab attacks the most advanced, comprehensive and invasive hardware attacks for which the attackers have access to laboratory equipment and the knowledge to perform reverse engineering of a device and also monitor analog signals to perform attacks such as cryptographic key analysis.

infer their personal habits and activities [30]. 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 and user-centric architectures.

Server-centric approaches. Virtual trip lines (VTLs) [22] deal with the privacy issue by distributing the traffic monitoring service implementation across several specialized servers and by providing a spatiotemporal 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. A similar approach is AnonySense [10] which proposes a system composed of many entities such as the mobile nodes, the registration authority, the task services, the report service, the access point and the mix network. Each component takes responsibility for a specific role so that a single malicious entity in the system cannot have access to too much private information.

Shi et al. [39] propose PriSense, a people-centric urban sensing system to tackle the problem of the privacy concerns of participating individuals. The architecture of PriSense comprises two main components: powerful aggregation severs (ASs) and participating individuals (called nodes). Each AS is in charge of a certain region and provides network access services for the nodes in that region. The AS broadcasts queries in their region and the nodes contribute with their data to the AS if they are relevant. To protect the user privacy, a node randomly splits its data and sends the shares to other nodes in its region. Then the nodes send the shares to the AS which is able to compute base aggregation functions over the collected data. PriSense offers probabilistic privacy protection against malicious ASs and nodes. However, there are several drawbacks: the deployment cost is high since it requires an AS for each observed region, the communication cost is important since the data is split before being sent to the AS, the type of aggregation is limited to base statistical functions (e.g., average, min, max, percentile, variance) while the accuracy is low since a spatial unit corresponds to an AS region.

SpotMe [37] proposes a different approach consisting in mixing the 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. PoolView [17] tackles the problem of user privacy in participatory sensing applications by applying data perturbation techniques on the user-side to perturb private measurements before sharing. In PoolView,

there is a tradeoff between the user's privacy level and the accuracy of the system, i.e., the higher the required privacy level is, the higher the error in the computed statistics is. Also, no matter the perturbation method which is applied, there is no guarantee that the user's privacy is always protected. Besides, recent proposals address location privacy in different contexts, such as geo-indistinguishability in location-based services, by applying the differential privacy principle to geo-location [2], [8], [29], [26] or geo-obfuscation for the anonymization of workers in spatial task allocation [46], [26], [23]. However, depending on the chosen degree of obfuscation, on the one hand, the quality of service in such systems may suffer from the information accuracy loss, and on the other hand, temporal correlation [7] or background knowledge can still be used to infer the user's location[9].

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 [6], [15], [27], [35]. Such solutions can offer formal guarantees on location privacy and accountability to protect against users trying to upload large amounts of fake samples. Typically, the cryptographic solutions are based on homomorphic encryption schemes allowing a central-server [35], [27] or the users [15] 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, sum, 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 [6], [35]. 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. Moreover, depending on the encryption protocol, the accuracy of the aggregate result may also be impacted since only a (low) number of discrete range values can be computed with such protocols [6], [15], [27], [35].

Secure hardware and user-centric approaches. Recent works have also proposed the use of secure hardware at the user-side [1], [41], [42]. 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 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).

[9] https://github.com/chatziko/location-guard

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. [41], [42] propose a similar architecture, but consider the problem of executing SQL queries over a distributed database without revealing any sensitive information to central servers. In particular, the work in [41] proposes a secure protocol to evaluate GROUP BY SQL-like-queries, which are similar to some extent to mobile participatory sensing aggregates. In a nutshell, the secure devices at the user-side push their query relevant tuples in an encrypted form to a centralized server infrastructure. After collecting the tuple data, the central server randomly partitions the tuples and sends each partition to a secure device which decrypts it and produces partial aggregations (i.e., it aggregates the data belonging to the same group in a partition). The partial aggregations are then sent back to the server, which repeats the process (i.e., repartitions the intermediate results and resends them to some secure devices for further aggregation). Hence, several (or many) iterations are required before all tuples belonging to each group are aggregated. 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). To improve the efficiency of the secure protocol, an equi-depth histogram-based protocol is presented as an alternative in [41]. The idea is to partition the aggregation groups into buckets before the query processing, which leads to a uniform distribution of the data in the obtained buckets, similar to an equi-depth histogram. This allows the server to assemble the data tuples belonging to the same bucket into the same partition, which in turn allows the secure devices to compute the final aggregation result in one iteration. However, building an equi-depth histogram efficiently requires either knowing in advance the data distribution or having a nearly-static distribution and then computing it using the costly secure protocol introduced above. Unfortunately, these hypotheses are not verified in MPSS apps since the spatial distribution of the participants is not known in advanced and also rapidly evolves over time.

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.

Recently, a distributed vehicular rerouting system, called DIVERT, was proposed in [32] to optimize vehicle travel time through proactive traffic congestion avoidance. Similar to PAMPAS, DIVERT employs a hybrid architecture since, although the rerouting computation is done at the vehicles, a central

server is still needed to determine an accurate global view of the traffic. DIVERT proposes a privacy-aware traffic reporting scheme to protect the user privacy against attackers at the server side. However, applying this solution to generic participatory sensing applications (i.e., outside the context of path rerouting computation for congestion avoidance) is impractical since its privacy mechanism only allows users to upload their location information when located in low sensitive areas, i.e., road segments with significant traffic density since such roads naturally offer a higher degree of anonymity to the participants. Hence, vehicles may only send traffic reports to the server if the road density is higher than a predefined threshold. Moreover, in DIVERT the users are considered trusted. Different, PAMPAS employs secure hardware at the user side to prevent unauthorized access to private data and also provides secure protocols preventing a massive leakage of private data in case of a lab attack and collusion with the server.

Acknowledging users' privacy concerns as one of the main obstacles for long-term participation in MPSS, PriMe [28] defines a human-centric privacy measurement method based on users' perception. The user preferences related to sharing certain types of data are transformed into privacy metrics to measure the individual's inherent sensitivity and her sensitivity toward data items based on data features. However, on the one hand, pushing the privacy permission settings at the user side may lead to privacy issues, since not all participants are aware of the sensitivity of the data they want to share. Also, once shared, users lose control of their data. On the other hand, highly protective privacy settings may lead to very low accuracy levels of aggregation results. In PAMPAS, we take a different approach, i.e., all the user raw data is protected using secure hardware and secure aggregation protocols; only the aggregation results can be accessed by the participants or other third parties. Furthermore, our data protection has no impact on the result accuracy.

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 [4] 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. More important, such an approach exhibits the intrinsic problem of single point of failure and hence cannot protect against lab attacks on the secure hardware at the server side, a problem which we address in PAMPAS$^+$.

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. All acronyms used in this paper are described in Table 1.

Table 1: Acronyms used in the article

Acronym	Description
AES	The Advanced Encryption Standard symmetric cryptographic algorithm
IDW	Inverse Distance Weighting
MPS	Mobile Participatory Sensing
MPSS	Mobile Participatory Sensing System
PAMPAS	Privacy-Aware Mobile Participatory Sensing system and protocol
PAMPAS+	Enhancement of the PAMPAS protocol dealing with corrupted probes
PKI	Public Key Infrastructure
RSA	The Rivest-Shamir-Adleman asymmetric cryptographic algorithm
SHA	Secure Hashing Algorithm
SP	Secure Probe
SSI	Supporting Server Infrastructure
TEE	Trusted Execution Environment

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 a privacy-preserving protocol to exchange sensed sample updates, continuously compute the spatially aggregated results, and periodically partition SPs according to their location. This protocol can be either PAMPAS or an enhanced version of PAMPAS called PAMPAS$^+$ (but not both at the same time). 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.

Secure Probe (SP). Each user holds a secure portable device, which can be implemented by any type of (tamper-resistant) secure devices flourishing today and described in Section 1. In this paper, we consider the more challenging case of cheap, low-end secure devices (see Figure 2); extending the proposed solution to more powerful TEEs such as TrustZone or Intel SGX is straightforward in terms of computation efficiency and resource consumption. Whatever its commercial name and form factor, such a basic secure device, called secure probe (SP) hereafter, usually embeds at the very least a secure micro-controller (MCU) for computation (e.g., a smart card chip) connected

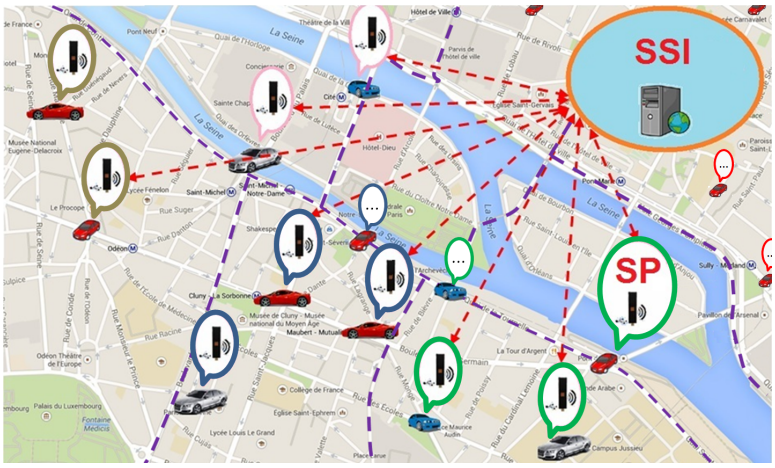

Fig. 1: Overview of PAMPAS's system architecture: mobile secure probes (SPs) leverage an untrusted Supporting Server Infrastructure (SSI) for efficient message passing among SPs. SPs continuously compute a spatial probe partitioning (depicted by the purple dashed lines) allowing for efficient and secure sample data aggregation

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. However, the high-level of security of SPs 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 in our context 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.

Fig. 2: Examples of cheap, low-end secure tokens that can be used in PAMPAS

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. In our architecture the SSI is not trusted. Therefore, all the temporary results stored in the SSI have to be encrypted, e.g., using non-deterministic encryption.

In conclusion, the foundation for the security and privacy in PAMPAS arises 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, while limiting the private information leakage to SSI or potentially corrupted SPs.

3.2 Base Threat Model

In this section, we present the base threat model considered in PAMPAS. 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 threat model is based on the following assumptions.

Assumption 1. Even though the users are untrusted, we assume in the base threat model that all the SPs are trusted. This is reasonable considering that a Trusted Execution Environment provides a high level of protection against hack attacks [3] (i.e., local or remote software attacks on the secure device) or shack attacks [3] (i.e., low budget hardware attacks). In addition, all the persistently stored data in the NAND Flash is cryptographically protected. Thus, SP owners cannot access the private data in their SPs.

In Section 6.1 we relax this assumption, i.e., we consider that some SPs could be exposed to lab attackss [3] (i.e., high budget hardware attacks) and thus be corrupted. Then, we modify the PAMPAS protocols accordingly.

Assumption 2. Each SP is supplied with a trustworthy certificate, e.g., from an offline trusted PKI. Without this assumption, an attacker can easily emulate nodes in the network, and conduct a Sybil attack [14], mastering a large proportion of SPs, thus defeating any countermeasure. We also assume that the hardware manufacturer is trusted and protects the secrets embedded in SPs.

Assumption 3. We assume an honest-but-curious (or covert adversary) SSI, i.e., it 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.

Assumption 4. We assume that the communication channel through which the SPs transmit (encrypted) raw data to 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.

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 information 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 analyzing the aggregate results, which is a complementary aspect of this work (e.g., see [29]).

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 [24] 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]
```

Fig. 3: The stream-SQL-like query formulation of the spatio-temporal aggregates supported 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 [15], [6], [35]. 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 [31]; (iii) *Holistic functions*: median, percentile, top-k. Such functions are also frequently used in statistical computations (e.g., the median speed of vehicles in a road network). 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 specific or holistic functions (see Section 2). Also, the holistic functions cannot be computed efficiently in a

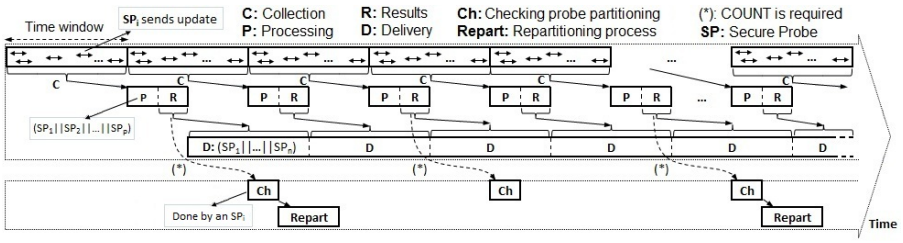

Fig. 4: Workflow representation of the global protocol in PAMPAS

distributed architecture by the secure protocol proposed in [41] (as shown in Section 8).

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 of 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 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 (cf. Assumption 2 in Section 3.2). 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 in delivering the current partial aggregate results to the queriers. Each querier needs to perform the final aggregation of these partial results, which is merely the union 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 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 2.

To address the performance limitations of the existing protocols (such as [41] as described in Section 2), the aggregation protocol in PAMPAS groups the participants based on their location, which permits processing by a single aggregating SP the reported sample updates from all the SPs in a spatial unit (forming a group) for a time window. Thus, the SSI pushes these samples to one aggregating SP which can aggregate the received data in one iteration. To put that in perspective, the secure protocol in [41] incurs a costly aggregation process in our context mainly because it requires a large number of iterations, as confirmed by our experimental evaluation (see Section 8). Note that an aggregating SP does not have to belong to the probe group for which it aggregates the samples. Indeed, the SSI randomly selects from all the participants as many aggregating SPs as there are probe groups and then assigns the samples in each group to an aggregating SP (as indicated in Algorithm 1).

Table 2: Notations used in the algorithms

Notation	Description
E_{sk}	Symmetric deterministic encryption using the shared key
nE_{sk}	Symmetric non-deterministic encryption using the shared key
E_{sk}^{-1}	Symmetric deterministic decryption using the shared key
nE_{sk}^{-1}	Symmetric non-deterministic decryption using the shared key
CK_{ij}	The couple key shared between SP_i and SP_j
nE_ck_{ij}	Symmetric non-deterministic encryption using the couple key shared between SP_i and SP_j
$nE\_ck_{ij}^{-1}$	Symmetric non-deterministic decryption using the couple key shared between SP_i and SP_j
AE_PK_i	Asymmetric encryption using the public key of SP_i
AE_SK_i	Asymmetric encryption using the secret key of SP_i
$AE\_SK_i^{-1}$	Asymmetric decryption using the secret key of SP_i
G_i	Identifier of probe group i
$P_{G_i}^{fake}$	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

Algorithm 1: PAMPAS aggregation protocol at the SSI-side

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

To this end, the probes 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

Algorithm 2: PAMPAS aggregation protocol at the SP-side

1 ***collection_period()*: /* for all SPs */**
2 /* Generate and send the sensing update: $update(G_i, sample)$ */
3 $message = (E_{sk}(G_i), nE_{sk}(sample))$;
4 $send\_anonymous(message, SSI)$;
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 $fake\_message = (E_{sk}(G_i), nE_{sk}(fake\_sample))$;
8 $send\_anonymous(fake\_message, SSI)$;
9 ***processing_period()*: /* only for the selected SPs, one for each G_i */**
10 **while** *message = receive(SSI)* **do**
11 $sample = nE_{sk}^{-1}(message)$;
12 $result = result \oplus sample$;
13 $enc\_result_i^{final} = (E_{sk}(G_i), nE_{sk}(result))$;
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* $\{E_{sk}(G_i)\}$ **do**
18 $send\_request(E_{sk}(G_i), SSI)$;
19 $result_i^{final} = nE_{sk}^{-1}(receive(SSI))$;

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, which greatly improves both the computation and the communication cost of the aggregation process. This is possible since an aggregating SP receives all the samples inside a spatial unit and can thus produce in one iteration the final aggregated results corresponding to the unit. This is valid for all the considered aggregated functions. However, the benefits in term of efficiency compared to a protocol requiring several iterations (such as the one in [41]) are greater for holistic functions. 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 explanations 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 spatial 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), then it may use this information to infer the approximate location of the participants (e.g., the road segment where a group of samples have been generated). This translates into larger privacy leakage for the users (which we quantify in Section 7.2), which indicates an increase of the probability with which the SSI could link subsequent sample updates and thus endanger the participants' 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.* Note that in this paper we opted for selecting spatially co-located units to constitute probe groups mainly for 'semantical' reasons. Typically, the privacy location metrics (see Section 7.2) are defined with respect to some spatial region. In general, a region refers to a connected subspace in the related work. That being said, we believe that the choice between regions formed by spatially co-located units or dispersed units has little if no impact both on the efficiency and the security of the protocols. Indeed, as long as the employed partitioning algorithm produces relatively balanced groups in terms of contained number of SPs (while the remaining differences are compensated by fake sample injections), the overall cost and security of the protocols are preserved.

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 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. To allow for an efficient probe partitioning in PAMPAS, we associate an index value to each spatial unit of the query and

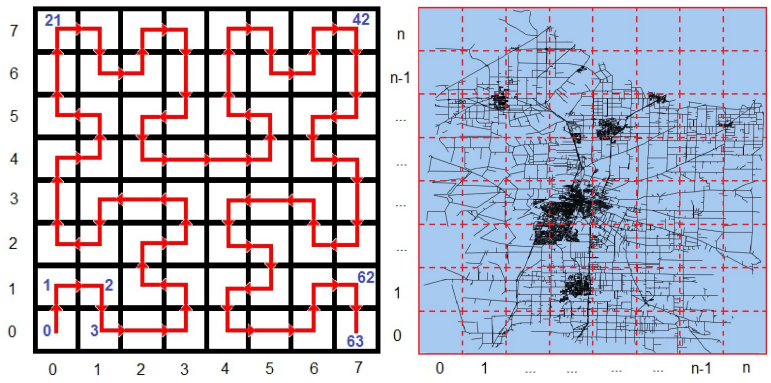

Fig. 5: Basic examples showing the indexing of spatial units in the case of free movement (left) or constrained movement (right)

then sort the spatial units on the index values. In our system, we use Hilbert curves to index the spatial units considered by the participatory sensing application. The index values correspond to the indices of the Hilbert curve covering the spatial units.

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 labeled with the Hilbert index, we sort them on the index value to obtain a sorted unit vector. This sorted unit vector is broadcasted to all the SPs and used, in the probe partitioning phase, as reference by the partitioning algorithms (i.e., in both the linear partitioning algorithm - see Algorithms 3 and 4 - and the binary partitioning algorithm - see Algorithm 9).

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.

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_sk(G_i)} do
4          send_request(E_sk(G_i), SSI);
5          enc_result_i^final = receive(SSI);
6          allCounts[G_i][] ← E_sk^-1(enc_result_i^final) ;
7          update localy stored counts for spatial units ;
8          compute groupWeights[G_i] = SUM(allCounts[G_i][])/* also required to
               compute the probability to generate fake samples */;
9      compute standard_deviation(weights);
10     if standard_deviation(weights) < threshold then
11         send_for_broadcast(nE_sk(groupWeighs), SSI);
12     else
13         execute probes_repartitioning();
```

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[10] 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.

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 12-17 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

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

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(groupWeights[])/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                 newGroupWeights[].add(currentGroupWeight);
17                 currentGroupWeight = 0;
18         /* check if the new partitioning for tN_G is better than for N_G*/
19         compute tQI_comp and tQI_comm for tN_G;
20         if tQI_comp + tQI_comm < QI_comp + QI_comm then
21             N_G = tN_G; QI_comp = tQI_comp;
22             QI_comm = tQI_comm;
23         else
24             break;
25     message = newGroupWeights[]||newPartitionMilestones[];
26     send_for_broadcast(nE_sk(message), SSI);
```

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 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 units 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.

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 units, 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.

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

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

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. 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 re-partitions 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 20-24 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 PAMPAS+

In this section, we present an improved version of the proposed PAMPAS protocol that we name PAMPAS+. The objective of PAMPAS+ is twofold. First, PAMPAS+ enforces the security of the base protocol by taking into account an enlarged threat model compared to the threat model introduced in Section 3.2. Our second objective is to limit as much as possible the impact of the increased security on the protocol requirements that were defined in Section 3.4. Specifically, the increased security should not be achieved at the price of a significant tradeoff with the desirable features of PAMPAS, i.e., privacy, generality, efficiency, scalability and accuracy.

We introduce first the extended threat model of PAMPAS+. Then we present the modified aggregation and partitioning protocols of PAMPAS+. The analysis of the privacy level and efficiency of PAMPAS+ in comparison with the basic PAMPAS protocol are presented in Section 7 and Section 8 respectively.

6.1 Extended threat model

The PAMPAS protocol is based on four assumptions enunciated in Section 3.2. In particular, Assumption 1 considers that the SPs are inviolable, which is reasonable since the SPs integrate tamper-resistant hardware. Hardware protection has the salient advantage of precluding software and simple (low-budget) hardware attacks. Yet, no hardware security can be described as unbreakable. For this reason, in many cases the security of a system is rather evaluated by comparing the estimated cost of a successful attack with the potential benefit. If the benefit of a successful attack is much lower than its cost (e.g., financial or invested time), the system can be reasonably assumed to be secure. In the opposite case, the system can be considered at risk. In the light of this consideration, we revisit below the first assumption of the base threat model. The rest of the assumptions (2 to 4) defined in Section 3.2 remain unchanged.

Assumption 1'. We assume that one or a few SP owners have the means to conduct an advanced lab attack on the devices in their possession. A lab attack is usually described as an invasive hardware attack for which the attacker has the knowledge to perform reverse engineering of a device and also has access to the necessary laboratory equipment. Such an attack, if successful, can offer the attacker access to secret cryptographic material (e.g., the secret encryption keys) stored inside the device.

This new assumption has a major impact on the threat model. Let us consider that following a successful attack, the attacker colludes with the SSI by revealing to the SSI the shared secret key used to encrypt the messages that are exchanged between SPs. In this case, the SSI can decrypt all the messages in transit. In particular, the SSI has access to the update messages from SPs containing the location of each user. Therefore, the privacy of all the participants can be entirely compromised.

Although the cost of the above presented attack is hard to estimate, this shows that the risk of successfully corrupting one SP is to reveal to the attackers all the sensitive information that are managed by the system. Therefore, the first objective of PAMPAS$^+$ is to protect the participants against this type of attack. To this end, we extend the threat model defined in Section 3.2 by taking into account the possibility that a few participating SPs could be corrupted, while the vast majority of the SPs remain trusted. Specifically, our extended threat model is based on two main assumptions. First, even if a lab attack on one or a few SPs is costly, it can be interesting for an attacker if the attack benefit is high. In our case, a high benefit is equivalent to getting access to a significant amount of the private information. Second, we consider that an attacker can directly access, at most, a very limited number of SPs and therefore could control only a few SPs in the system. In other words, we do not consider the case of a powerful state-size attacker that would be able to control a significant proportion from the total number of SPs. In this context, PAMPAS$^+$ searches to minimize the information leakage by the corrupted SPs and to avoid a complete "meltdown" of the system as it would be the case for the base protocol. The idea is to ensure that the amount of privacy leakage remains proportional with the ratio of corrupted SPs in the system and therefore impose to an attacker to corrupt many SPs to gain access to a significant amount of private data, which contradicts the assumption above.

6.2 Aggregation Protocol in PAMPAS$^+$

The main idea in PAMPAS$^+$ is to avoid using a single shared key to encrypt all the messages that the SPs exchange and that are relayed by the SSI. An alternative to using the SSI as a message dispatcher would be to employ direct point-to-point communications between the SPs. However, this latter solution is not feasible in our context since it implies large resource consumption at the SP side. For instance, in the aggregation phase the aggregating SPs would have to manage a high number of connections (e.g., hundreds or thousands) in a very short time window (i.e., during the duration of the aggregation phase), which is not reasonable. For this reason, we still employ the SSI in our system architecture.

Nevertheless, to avoid the risk of complete meltdown in case of a successful lab attack on an SP, we propose in PAMPAS$^+$ that each message transmitted by an SP to the SSI to be encrypted such that only the recipient SP can decrypt it. We have basically two options to achieve this. The first is to use an asymmetric public-secret key encryption (e.g., RSA encryption). Typically, in the aggregation phase, the SPs in a group are informed about the identity of the aggregating SP. Therefore, all the messages in the group are encrypted with the public key of the aggregating SP. This public key can be broadcasted at the beginning of aggregation phase for the SPs that do not have it. The problem with this approach is the relatively large cost of the asymmetric encryption. For instance, with our secure tokens, the cost of encrypting/decrypting a message

Algorithm 5: PAMPAS+ couple keys exchange protocol at the SSI-side

```
1  SSI-side():
2      /* Execution thread at SSI managing system join requests from new SPs */
3      while true do
4          joinMessage(SP_i) = (certificate_SP_i, certifiedID_SP_i) ;
5          newDeliveryList_i = {allRegisteredSPs} ;
6          associate joinMessage(SP_i) to newDeliveryList_i ;
7          add newDeliveryList_i to {allDeliveryLists} ;
8          foreach list_k in {allDeliveryLists} do
9              /*Send join messages to the currently connected SPs*/
10             targets = list_k → getDeliveryList() ⋂ {allConnectedSPs};
11             broadcast(joinMessage(newSP_k), targets);
12             remove targets from list_k;
13             if isEmpty(list_k) then
14                 delete list_k;
15         {allRegisteredSPs} = {allRegisteredSPs} ⋃ SP_i ;
16     /* Execution thread at SSI for dispatching couple key messages to SPs */
17     while true do
18         newCoupleKeyMessage = (target_j, certificate_SP_i,
           certifiedID_SP_i, AE_PK_j(CoupleKey_ij)) ;
19         add newCoupleKeyMessage to {allCoupleKeyMessages};
20         foreach message in {allCoupleKeyMessages)} do
21             /* Push couple key messages to the currently connected SPs*/
22             target = message → getTarget();
23             if isConnected(target) then
24                 unicast(message, target);
25                 remove message from allCoupleKeyMessages;
```

using RSA with a key length of 256 bytes is around 0.4/0.6 seconds. This leads to a large aggregation time, which is mostly explained by the high decryption time that dominates the overall cost of the aggregation at the aggregating SP, and to not respecting the efficiency requirement of the protocol.

The second and retained option is to use the very efficient symmetric encryption (e.g., AES 256) as in the base protocol. This requires that for any two SPs participating in the system, there has to be a shared symmetric encryption key only known by the two SPs (called *couple key* hereafter) and used to encrypt/decrypt all the messages that are exchanged between the two SPs.

Couple keys exchange protocol. Let us note that the secret couple keys cannot be generated on the fly in real-time. For example, at the aggregation time, the aggregating SP would have to contact and establish a secret key with all the SPs in its partition with which a secret key has not been established before. This is much too costly since it requires exchanging asymmetrically encrypted messages which incur an important processing time as described above.

Therefore, the secret couple keys have to be generated "offline" (i.e., outside the aggregation process) (see Algorithms 5 and 6). Typically, when a new SP registers in the system, it sends a "Hello" message to the SSI containing its

Algorithm 6: PAMPAS$^+$ couple keys exchange protocol at the SP-side

1 ***startRegistration()*: /* for each newly arrived SP */**
2 /* Build a certified ID to be send to all the other registered SPs*/
3 $certifiedID\_SP_i = AE\_SK_i(Hash(info\_SP_i))$ /*Where $info\_SP_i = (PublicKey\_SP_i, alias\_SP_i/IPaddress\_SP_i)$ */;
4 /* Send a couple-key generation request to all the other registered SPs*/
5 $joinMessage\_SP_i = (certificate\_SP_i, certifiedID\_SP_i)$;
6 $send\_for\_broadcast(joinMessage\_SP_i, SSI)$;
7 ***coupleKeyCreation()*: /* for the connected SPs selected by the SSI */**
8 $joinMessage = receiveJoinMessage(SSI)$;
9 $certificate\_SP_i = joinMessage \rightarrow certificate\_SP_i$;
10 $certifiedID\_SP_i = joinMessage \rightarrow certifiedID\_SP_i$;
11 $authentic = verifyCertificate(certificate\_SP_i, certifiedID\_SP_i)$;
12 **if** *authentic* **then**
13 $newCoupleKey = AESRandomKeyCenerator()$;
14 $certifiedID_{self} = AE\_SK_{self}(Hash(info_{self}))$;
15 $newCoupleKeyMessage = (target\_SP_i, certificate_{self}, certifiedID_{self}, AE\_PK_i(newCoupleKey))$;
16 $send\_for\_unicastcast(newCoupleKeyMessage, SSI)$;
17 add $(newCoupleKeyMessage, tag_{ij})$ to $CoupleKeyIndex$;
18 /* the tag_{ij} is computed separately by SP_i and SP_j as $Hash(PK_i \| PK_j)$ such as $PK_i < PK_j$ */
19 ***coupleKeyRegistration()*: /* for the newly registering SP */**
20 $coupleKeyMessage = receiveCoupleKeyMessage(SSI)$;
21 $certificate\_SP_i = coupleKeyMessage \rightarrow certificate\_SP_i$;
22 $certifiedID\_SP_i = coupleKeyMessage \rightarrow certifiedID\_SP_i$;
23 $authentic = verifyCertificate(certificate\_SP_i, certifiedID\_SP_i)$;
24 **if** *authentic* **then**
25 $encryptedCoupleKey = coupleKeyMessage \rightarrow AE\_PK_{self}(newCoupleKey)$;
26 $newCoupleKey = AE\_SK^{-1}_{self}(encryptedCoupleKey)$;
27 add $(newCoupleKeyMessage, tag_{ij})$ to $CoupleKeyIndex$;
28 /* the tag_{ij} is computed separately by SP_i and SP_j as $Hash(PK_i \| PK_j)$ such as $PK_i < PK_j$ */

identification (i.e., signed certificate proving that it is an SP and containing its public key, see lines 1-6 in Algorithm 6 and lines 4-7 in Algorithm 5). This message is broadcasted by the SSI to all the already registered SPs (lines 8-14 in Algorithm 5). At receiving a "Hello" message, each SP generates a secret random key used to communicate with the newly arrived SP, encrypts the secret key with the public key of the new SP and sends the encrypted message (lines 7-16 in Algorithm 6) to the SSI that will push it to the new SP. Both the SP generating the couple key and the newly registered SP add the new couple key to their couple key index using a couple key tag (lines 17-18 and 25-28 in Algorithm 6). More details about the usefulness of the couple key tags are given in Algorithms 8 below.

This process can be costly (i.e., the number of exchanged messages is proportional with the number of registered SPs) but it does not have to cope with

real-time constraint of the main aggregation process. The major inconvenient is that a new SP entering the system cannot communicate with another SP until the corresponding couple key is not created.

The idea of generating pairwise shared AES keys is a standard method to increase cryptographic efficiency (instead of using asymmetric encryption based on the public/private key). The same goes for the couple key exchange protocol which is based on the classical approach of using the PKI certificates to authenticate the two nodes to each other and also permit the secure couple key exchange. Note also that the security practices require to periodically update the couple key to prevent an attacker that obtains the key to gain access to the entire history of exchanged messages. However, in the context of distributed data aggregation using secure hardware, the idea to use a dedicated secret key to secure the data exchange between each couple of peers is novel. For instance, [41, 42] use a single secrete key shared among all peers as in the basic PAMPAS protocol. Similarly, in the context of distributed privacy-preserving data publishing with secure hardware, [1] uses the traditional approach that divides the peers into several clusters and employs one key for each cluster. Thus, in case of a lab attack, the data leakage is contained to the participants belonging to the same cluster as the compromised peer. In our context, the clusters correspond to the probe groups obtained through partitioning. Nevertheless, this approach is not feasible in PAMPAS because the high mobility of the participants leads to frequent probe repartitioning, which in turn requires the costly operation of renewing the group keys at each repartitioning.

By using a dedicated key for each pair of SPs, PAMPAS$^+$ achieves a much stronger security level, while protecting the privacy of the participants even in case of successful lab attacks (as discussed in Section 7.3). On the other hand, the increased security introduces a computational overhead since the probes have to manage a large set of keys instead of a single key. The second major contribution of PAMPAS$^+$ is to propose optimizations to alleviate this problem and keep the protocol efficient and scalable. There are two significant optimizations. The first optimization is the tagging and indexing of the shared keys by each SP, which allows an optimal retrieval of the encryption key for each received update sample. The second optimization is a new partitioning algorithm able to compute a more balanced partitioning (see Section 6.3), which reduces the number of inserted fake samples and thus improves the aggregation time (see Sections 8.3 and 8.4).

Shared encryption key. The couple keys allow a secure data exchange between SPs through the SSI during the collection and processing periods (see Figure 4). Nevertheless, the global PAMPAS protocol also includes a delivery period and a probe partitioning process. The delivery period consists in delivering the aggregation results to the participating SPs. The partitioning process consists in checking the current partitioning based on the spatial distribution of the SPs and in computing and delivering to all the SPs a new probe partitioning, if the current one is unbalanced. To keep the delivering of the aggregation and the partitioning results efficient, we continue to use in

PAMPAS$^+$ a shared encryption key as in the initial PAMPAS protocol (see Section 4). Otherwise, the aggregating SP or the probe partitioning SP would have to individually encrypt and send the results to the other SPs, which would consume too many resources of the worker SP and take significant time.

Using a shared key in the delivery phase of the aggregation protocol has no impact on the privacy guarantees of our protocol since as indicated in our threat model (see Section 3.2), our protocol focuses on the protection of the raw data and the aggregation process and not on the exposure risks that arise from analyzing the aggregate results. However, using a shared key for delivering the partitioning results has the inconvenient of reducing the privacy guarantees of the protocol in case of a successful lab attack. This is thoroughly analyzed in Section 7.3. For now, let us note that this does not call into question the complete security of the PAMPAS$^+$ protocol, but merely the level of the privacy offered by the proposed protocol.

Global aggregation in PAMPAS$^+$. Algorithms 7 and 8 present the aggregation protocol in PAMPAS$^+$ at the SSI-side and SP-side respectively. The new protocol is similar with the basic protocol of PAMAPS and consists of a collection period, a processing period and a delivery period. Compared with the basic protocol, there are two major modifications in PAMPAS$^+$. First, the selection of the aggregating SPs is done at the beginning of the collection period and not during the processing period. The reason is that the list of the worker SPs has to be broadcasted to all the SPs beforehand so that each participating SP can choose the appropriate couple key (i.e., the couple key corresponding to the aggregating SP in its partition) to encrypt its sample data (lines 2-7 in Algorithm 7 and lines 2-5 in Algorithm 8).

Second, each sample is encrypted using a couple key instead of the shared key (lines 6-10 in Algorithm 8). Beside, each SP holds a hash index allowing to efficiently retrieve the couple key shared with another SP (line 7 in Algorithm 8). The couple key is also associated with a unique tag generated at the creation of the key and shared by the two SPs holding the key. The tag is required in the processing period to allow an aggregating SP to know which couple key to use to decrypt a sample (lines 18-20 in Algorithm 8) since the samples are anonymously sent to the SSI (line 11 in Algorithm 8). A second hash index is equally built on the couple keys to allow the efficient retrieval of a couple key given the associated tag (line 19 in Algorithm 8). As in the basic protocol, the SPs probabilistically send fake samples to compensate for the unbalanced number of probes in the groups. Finally, we note that the security of the protocol, and in particular the case of corrupted aggregating SPs, is analyzed in Section 7.3.

6.3 Partitioning Protocol in PAMPAS$^+$

In this section, we propose a new probe partitioning algorithm to be used with PAMPAS$^+$. We note from the beginning that the initial partitioning protocol proposed in Section 5 can be applied without any change in the context of

Algorithm 7: PAMPAS$^+$ aggregation protocol at the SSI-side

```
1  collection_period():
2      /* Randomly select an SP for each group and broadcast the list to all connected
       SPs */
3      aggSPList = EMPTY ;
4      foreach i in {E_sk(G_i)} do
5          randomly select SP_i;
6          aggSPList.addPair(SP_i, E_sk(G_i)) ;
7      broadcast(aggSPList) ;
8      /* Receive encrypted updates from SPs */
9      while true do
10         message = (E_sk(G_i), nE_ck_ij(sample), tag_ck_ij) ;
11         store(message) → list[E_sk(G_i)][currentTimeWindow];
12 processing_period():
13     foreach i in {E_sk(G_i)} do
14         /* feed in parallel the selected aggregating SPs */
15         while message ← list[E_sk(G_i)][lastTimeWindow] do
16             send(message, SP_i);
17     foreach i in {E_sk(G_i)} do
18         /*Receive the final results from the worker SPs*/
19         encResult_i^final = (E_sk(G_i), nE_sk(result));
20 delivery_period():
21     foreach i in {E_sk(G_i)} do
22         /*Send result_i to all requesting SPs*/
23         multicast(encResult_i^final, {SP_k});
```

PAMPAS$^+$. Thus, the new partitioning algorithm proposed in this section is not motivated by a lack of compatibility but by the need of increasing the effectiveness. Specifically, our initial partitioning algorithm (lines 9 to 17 in Algorithm 4) is very efficient, its cost being linear with the number of spatial units (i.e., $O(N)$ and $O(N_G)$ time and space complexity respectively, where N is the number of spatial units and N_G is the number of groups). However, this high efficiency is counterbalanced by the quality of the obtained partitioning (i.e., the balance factor with respect to the number of probes in the obtained partitions) since the partitioning decisions are taken based on local knowledge of the probe distribution in the spatial units.

The main objective of the new partitioning algorithm is to propose an alternative to the base algorithm which offers a different tradeoff between the efficiency and the quality of the partitioning. The new algorithm (see Algorithm 9) can increase the partitioning quality by making partitioning decisions that are based on the global knowledge of the probe distribution in the spatial units. The idea behind the proposed algorithm is to recursively split the sorted vector of spatial units such that the total number of probes is balanced as most as possible in the two obtained sub-vectors. This is equivalent

Algorithm 8: PAMPAS$^+$ aggregation protocol at the SP-side

***collection_period()*: /* for all SPs */**
  /* Get the list of aggregating SPs */
  $aggSPList = receive(SSI)$;
  /* Get the aggregating SP for this SP's current group */
  $aggSP = getSP(aggSPList, E_{sk}(G_i))$;
  /* Get the couple key and the key tag corresponding to this SP and the aggregating SP */
  $CK_{ij} = getCK(aggSP)$;
  $tag_{ij} = CK_{ij} \rightarrow tag$;
  /* Generate and send the sensing update: $update(G_i, sample)$ */
  $message = (E_{sk}(G_i), nE\_ck_{ij}(sample), tag_{ij})$;
  $send\_anonymous(message, SSI)$;
  /* Send a fake sample to the SSI with probability $P_{G_i}^{fake}$ */
  if $rand(0, 100) \geq P_{G_i}^{fake}$ **then**
    $fakeMessage = (E_{sk}(G_i), fakeSample, fakeTag)$;
    $send\_anonymous(fakeMessage, SSI)$;
***processing_period()*: /* only for the selected SPs, one for each G_i */**
  while *message = receive(SSI)* **do**
    $tag_{ij} = message \rightarrow tag_{ij}$;
    $CK_{ij} = getCK(tag_{ij})$;
    $sample = nE\_ck_{ij}^{-1}(message)$;
    $result = result \oplus sample$;
  $encResult_i^{final} = (E_{sk}(G_i), nE_{sk}(result))$;
  $send(encResult_i^{final}, SSI)$;
***delivery_period()*: /* for all SPs */**
  /* Pull the results for required $\{G_i\}$ from the SSI */
  foreach *i in* $\{E_{sk}(G_i)\}$ **do**
    $send\_request(E_{sk}(G_i), SSI)$;
    $result_i^{final} = nE_{sk}^{-1}(receive(SSI))$;

to building a binary partitioning tree on the sorted vector of spatial units of height $h = \log(N_G)$.

The binary partitioning algorithm is implemented as a recursive function (lines 23-32 in Algorithm 9). The function *isLeaf* (line 24) decides if a node is internal (and therefore needs to be split) or not, while the *split* function determines the optimal position in the current node to do the split. The price to pay for the potential higher quality of the partitioning is an increased computational complexity. Nevertheless, the complexity of the new algorithm remains low and as we show in Section 8.3, it can still be used in real-time in our context. Specifically, the new algorithm has $O(N \times \log(N_G))$ and $O(N_G)$ time and space complexity respectively.

Parallel partitioning. Beside, the partitioning algorithm can be easily parallelized (i.e., executed in parallel by several SPs), which can drastically reduce its execution time. As opposed to the base partitioning, several SPs can be involved in the partitioning process of PAMPAS$^+$. Specifically, there is a

Algorithm 9: Repartitioning process in PAMPAS$^+$ (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         curNode = 0;
11         mileStone1 = 0; mileStone2 = N − 1;
12         treeTable = new TreeTable();
13         BinaryPartitioning(treeTable, mileStone1, mileStone2,
                              curNode, tN_G);
14         /* check if the new partitioning for tN_G is better than for N_G*/
15         compute tQI_comp and tQI_comm for tN_G;
16         if tQI_comp + tQI_comm < QI_comp + QI_comm then
17             N_G = tN_G; QI_comp = tQI_comp;
18             QI_comm = tQI_comm;
19         else
20             break;
21     send_for_broadcast(nE_sk(treeTable), SSI);
22 BinaryPartitioning(treeTable, mileStone1, mileStone2, curNode, tN_G)
23     treeTable[curNode].Info = [mileStone1, mileStone2];
24     if not isLeaf(treeTable[curNode], tN_G) then
25         treeTable[curNode].Type = Internal_Node;
26         splitPoint = split(treeTable, mileStone1, mileStone2,
                              curNode);
27         child1 = new Child();
28         chile2 = new Child();
29         BinaryPartitioning(treeTable, mileStone1, splitPoint,
                              child1, tN_G);
30         BinaryPartitioning(treeTable, splitPoint + 1, mileStone2,
                              child2, tN_G);
31     else
32         treeTable[curNode].Type = Leaf_Node;
```

coordinator SP that triggers the partitioning and computes the top levels of the binary partitioning tree. Then, the coordinator SP distributes the remaining partitioning of the bottom levels of the tree to other SPs. The final partitioning results are in the end aggregated by one selected SP (e.g., the coordinator SP). The benefit of this approach is that the bottom levels of the partitioning tree are computed in parallel by several SPs. On the other hand, a parallel partitioning also entails a communication cost between the participating SPs. We study these aspects in detail and compare the efficiency and effectiveness of the two proposed partitioning algorithms in Section 8.3.

Let us finally note that the checking of the probe partitioning in PAMPAS$^+$ is done exactly the same as in the base protocol (see Algorithm 3) and therefore further discussions on this aspect are omitted in this section.

7 Security and Privacy Analysis

In this section, we discuss the security and privacy of the proposed protocols. We analyze first the security and privacy of the base PAMPAS protocols in Section 7.1 and 7.2 respectively. Then, we analyze the additional privacy protection offered by PAMPAS$^+$ in comparison with the base PAMPAS in Section 7.3.

7.1 Security Analysis

The security analysis presented in this section is valid if all the four assumptions presented in Section 3.2 hold. 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 encrypted 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 raw data generated by 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.

7.2 Privacy Analysis

As discussed in the previous section, neither the SP users, nor the SSI have access to the samples (containing the exact locations of the probes) or the intermediary results generated by the SPs. Hence, the privacy leakage can only be described in terms of statistical information exposure or location obfuscation level. To underline the high level of privacy protection of PAMPAS w.r.t. the SSI, we consider two metrics, i.e., an entropy-based metric and a location obfuscation metric, and apply them 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. We focus on the privacy leakage only at the SSI side since the SSI is untrusted and constitutes the transit point of all the probe generated data, while an SP gets external probe data sporadically (i.e., only when it is selected as aggregating SP).

Entropy-based metric. Entropy is a popular metric to describe location privacy in general [11], and it is also appropriate to describe the privacy-preserving mechanism of PAMPAS, similar to the work in [32]. 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 *Entropy privacy leakage*: the entropy privacy leakage for each $update_k$ sent by user u_i is defined as:

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

The value of the entropy leakage for an update varies between 0 and 1, with 0 indicating that there is no entropy leakage and 1 indicating a maximum entropy leakage. The minimum value is obtained when the popularity of the considered region is extremely high, i.e., there is a very large number of users inside the region. The maximum value is obtained in the opposite case, i.e., the user is alone inside the region.

Location obfuscation-based metric. Location obfuscation is a widely used technique in mobile applications to protect the user privacy [11], e.g., to protect the identity [9] or the behavior [12] of the user. The idea is simple: the exact user location reported in an update is replaced with a region containing the user location [19], [20]. The larger the region is, the larger the obfuscation and hence the protection. The protocols in PAMPAS do not reveal exact probe locations to the SSI but leak some information regarding the group to which each sample belongs to. Since each group is associated with a spatial partition in the observed space (representing a region or a set of road segments), the location obfuscation becomes a relevant metric in our context, e.g., if the SSI knows the spatial extent of the partitions. Let obs_space denote the total observed space (two-dimensional or network) of an MPSS app. Let obf_region be the obfuscation area of an update with $obf\_region \subseteq obs\_space$.

Definition 4 *Location privacy leakage*: the location privacy leakage for each $update_k$ sent by user u_i is defined as:

$$location\_leak_{u_i}(update_k) = \begin{cases} 1 - \frac{size(obf\_region)}{\alpha \cdot size(obs\_space)}, & \text{if } size(obf\_region) \\ & \leq \alpha \cdot size(obs\_space) \\ 0, & \text{if } size(obf\_region) \\ & > \alpha \cdot size(obs\_space) \end{cases}$$

where α is a real value indicating the relative location sensitivity w.r.t. the size of observed space.

Definition 4 offers a simple way to quantify location leakage, with normalized values between 0 and 1 as for the entropy leakage. It indicates that the location privacy leakage for an update depends on the area of the obfuscation region hiding the real location in the case of non-constrained movement or on the total length of network segments in the region in the case of constrained movement. If the obfuscation space is greater than the entire application space multiplied by the location sensitivity factor α then the location leakage is minimum, i.e., 0. If the update contains the exact location of the user (i.e., a point in space), the leakage has maximum value, i.e., 1. The location sensitivity factor α is a parameter indicating the region size (relative to the observed space

size) from which there is a location privacy leakage. For instance, for $\alpha = 0.01$, there is a location leakage only when the obfuscation region is smaller than 1% of the observed space.

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 8.1). 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. Finally, we consider that there is a location privacy leakage only when the obfuscation region is smaller than 200 times the average road segment length, i.e., $\alpha = 0.01$.

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 entropy leakage of $entropy\_leak = 0.002$ 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. Similarly, the average location leakage is $location\_leak_{u_i} = 1 - \frac{Average\ road\ segment\ length}{0.01 \cdot Total\ road\ segment\ length} = 0.995$.

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 $entropy\_leak(update) = 10^{-3}$, 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, i.e., $location\_leak_{u_i} = 1 - \frac{1}{0.01 \cdot 200} = 0.5$.

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 $entropy\_leak(update) = 5 \cdot 10^{-7}$, while the location leakage is $location\_leak_{u_i} = 0$. 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.

7.3 Privacy Benefits of PAMPAS$^+$

The first observation is that PAMPAS$^+$ offers the same privacy level as PAMAPS if the base threat model (see Section 3.2) is considered. The explanation is that the global mechanism of both the aggregation and the partitioning protocols remains the same in PAMPAS$^+$ compared to PAMPAS. In particular, the insertion of fake tuples balances the number of messages generated by the probe groups and precludes the SSI from inferring even the approximate location of an update as explained in Section 7.1 and 7.2.

Besides, the PAMPAS$^+$ aggregation protocol is designed to offer strong privacy guarantees to the participants in the context of an extended threat model. The objective is to make the protocol robust to the (unlikely) situation in which one or a few SPs are compromised as described in Assumption 1' in Section 6.1. We analyze hereafter the potential privacy leakage for a participant in the worst case scenario with respect to the extended threat model, i.e., a few SPs are compromised and at least one compromised SP colludes with the SSI.

PAMPAS and lab attacks. Let us first consider the basic PAMPAS protocol. In this case, a corrupted SP that colludes with the SSI is equivalent with the SSI obtaining the shared secret key. This allows the SSI to decrypt all the messages that are exchanged between the SPs. Typically, the SSI gets access to the sample data periodically sent by each participating SP, i.e., $sample = (location, time, value)$. If the location data is approximated to the spatial unit that contains the actual location of the SP (e.g., only the road identifier is reported and not the exact location on the road), then the privacy leakage for each update is the same with the value computed in Scenario 1 in Section 7.2 (i.e., $entropy\_leak(update) = 0.002$ and $location\_leak_{u_i} = 0.995$). However, to achieve high accuracy it is likely that the applications would require the exact location of the probes, which leads to $location\_leak_{u_i} = 1$. Even if the updates are sent anonymously, having access to the exact location of each update makes it easier for the SSI to link the updates coming from the same probe (e.g., in relatively sparse areas) and thus compromise the privacy of the participants.

PAMPAS$^+$ and lab attacks. Let us have the same privacy analysis in the context of PAMPAS$^+$. Corrupting an SP and colluding with the SSI is equivalent to offering the SSI the couple keys and the shared secrete key stored in that SP. The access to the couple keys of an SP allows the SSI to decrypt the messages that are destined to that SP. This happens in the

aggregation phase whenever the corrupted SP is chosen as aggregating SP. However, since the aggregating SP has to change at each aggregation time window, the SSI will get access to the raw data in a single time window. For a high number of participants, the probability to leak the data collected in a time window is equal to the ratio between the number of corrupted SPs and the total number of SPs, i.e., very low. Therefore, although the privacy leakage can be occasionally significant (e.g., $entropy\_leak = 0.002/location\_leak_{u_i} = 1$ as discussed in the previous paragraph), the very rare periodicity of this situation makes that the overall privacy leakage to remain very low, i.e., similar to the leakage in which none of the SPs is corrupted. To increase significantly the privacy leakage and therefore the probability to reconstruct the trajectories of the SPs, an attacker would have to corrupt a significant number of SPs, which has a prohibitive cost. This shows the major advantage of the PAMPAS$^+$ protocol over the basic protocol, i.e., having a dedicated encryption key for the communication between each two SPs.

On the other hand, PAMPAS$^+$ uses a single shared key for the efficient transmission of the partitioning and aggregation results. Gaining access to the shared key would allow the SSI to decrypt the partitioning information and hence to know the exact spatial extent of each partition. In addition, the SSI knows to which partition belongs each update since the partition value is deterministically encrypted with the shared key to allow the SSI to group the messages partition-wise (line 11 in Algorithm 7). Hence, the entropy leakage of each update transmitted by an SP can be estimated as done in Scenario 2 in Section 7.2 (i.e., $entropy\_leak(update) = 10^{-3}$). At the same time, the location leakage is still small since the footprints of each participant are spatially hidden within the partition and among the footprints of the other participants in the same group, i.e., $location\_leak_{u_i} = 0.5$.

Thus, PAMPAS$^+$ offers good privacy protection even in the context of the extended threat model and definitely much better than compared with the basic PAMPAS protocol, in which the privacy leakage is associated with the exact location of the probe. We recall that in order to degrade the privacy protection to this level, an attacker has to corrupt at least one SP and collude with the SSI, which is costly. We recall also that the choice of keeping a shared secret key in PAMPAS$^+$ is determined by the efficiency and scalability requirements of the proposed protocol and therefore it is a tradeoff with the privacy protection level.

A final observation is that in the case of a lab attack, the privacy leakage in PAMPAS$^+$ depends on the probe partitioning, i.e., the number of probes in a partition for the entropy leakage, and the spatial size of a partition for the location leakage. Our current partitioning algorithms choose the number of probe groups strictly based on data aggregation performance criteria (see "Choosing the number of probe groups" in Section 5). However, extending the partitioning model to include privacy related criteria is easy. For instance, we can impose a minimum value for the number of probes in a group and/or a minimum spatial extent of a partition, which in turn limits the potential

entropy leakage and/or location leakage, respectively. Surely, this can have an impact on the aggregation efficiency.

7.4 Other Attacks

The attacks studied in the previous section can be considered as "passive" attacks since, letting aside the act of corrupting the SPs and colluding with the SSI, the attacker observes the private information to which he gains access after breaking the data confidentiality barrier, but does not derive from the original protocol. However, a lab attack can offer the attacker access to the cryptographic material protected by the secure hardware, which in turn could allow him to insert forged data into the system with the objective to infer or gain access to even more private data. We discuss in this section two important "active" attacks and explain the way the system handles them.

The first type of active attack from the owner of a corrupted SP would be to try to create a large number of fake identities that are all associated with the corrupted SP to gain a disproportionate influence in the system, i.e., the Sybil attack [14]. In this case, the benefit of the attacker is multiplied with the number of fake identities while the cost is minimized to corrupting a single SP. To deter Sybil attacks we rely on the fact that a user cannot easily obtain multiple user IDs because the user ID is derived from a user certificate and obtaining digital certificates is not cheap. Specifically, we consider that a PKI certificate is associated with the secure hardware of each SP and to gain access to multiple certificates, an attacker would have to possess and corrupt as many SPs, which contradicts the assumptions in our threat model (see Assumption 2 in Section 3.2). We recall that the certificate of an SP is verified at each couple key exchange with another SP. Hence, a "fake" SP, even if it succeeds registering in the system (at the SSI) and be chosen as aggregating SP, it will not be able to aggregate any data since this requires a prior couple key exchange with all the other "real" SPs in the system. Further, techniques such as [34], [45], complementary to our protocol, can be used to address these attacks.

The second possible active attack is related to the fact that the owner of a corrupted SP may detain the secret cryptographic keys, which allows him to either produce as many forged samples as needed (in the collection phase) or forge a fake aggregation result (in the aggregation phase in case it is chosen as aggregating SP). Tampering with the accuracy of the aggregation results may be used by an attacker as a way to infer more private information. Yet, it is reasonable to think that the attacker will make use of such actions only if he is sure he cannot be detected since otherwise he can be banned from the system. In the case of a corrupted SP forging many sample, an honest aggregating SP can easily detect that there are many samples coming from a same SP (since the samples have to be encrypted using the same couple key). Therefore, a corrupted SP could forge a maximum of one sample for each partition to be sure it cannot be detected. But this has a negligible influence on the accuracy of

the results knowing also that samples containing extreme values can be filtered out as being outliers. In the case of a corrupted aggregating SP, if the forged aggregation result is very different from the real result, this can be detected in the next protocol iterations when the results for the same partition are computed by honest SPs. A corrupted aggregating SP could still marginally perturb the result, but the benefit of doing so is equally small.

8 Experimental Evaluation

The goals of our experimental evaluation are twofold: (i) compare the execution time and scalability of PAMPAS and PAMPAS$^+$ with those of a state-of-the-art protocol described in [41]; (ii) quantify the cost and scalability of our partitioning and aggregation protocols. We implemented and validated PAMPAS and PAMPAS$^+$ through emulations using secure tokens which have a hardware configuration representative for secure hardware platforms. For PAMPAS$^+$, we consider that the couple keys (see the protocols in Section 6.2) have been exchanged between any two SPs participating in aggregation process. 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 [43] using a traffic monitoring scenario.

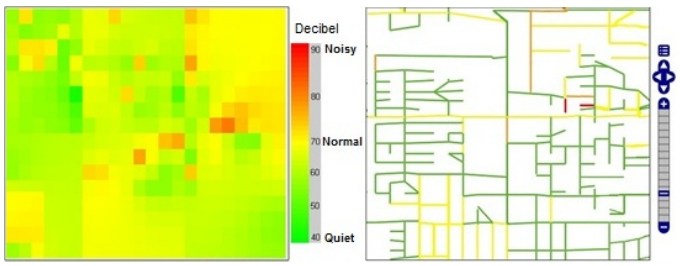

Fig. 6: Basic graphic visualization of aggregation maps for two applications: noise monitoring (left) and traffic monitoring (right) [43]

8.1 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

include 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.

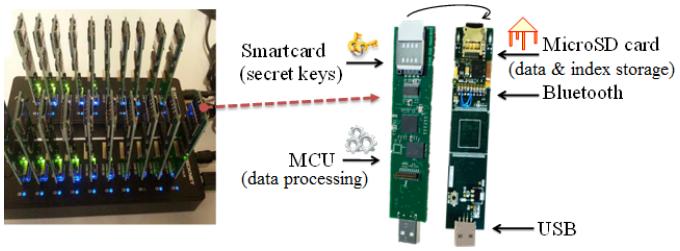

Fig. 7: Secure tokens used in our experimental evaluation

Baseline system. To underline the importance of the PAMPAS and PAMPAS$^+$ protocols, we implemented the *secure protocol* proposed in [41] and took it as the baseline. This protocol can be applied without modification to aggregate the samples collected in each time window. Since the basic PAMPAS offers the same level of security and privacy as the baseline protocol, while PAMPAS$^+$ offers even stronger privacy protection (see Section 7.3), our experimental evaluation focuses on the efficiency part. Note that in [41], two more protocols are proposed that are even more expensive than the secure protocol considered in our context.

Datasets and aggregate functions. We use both synthetic and real datasets to test the efficiency and scalability of PAMPAS and PAMPAS$^+$. We employed the well-known Brinkhoff generator [5] 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 [47]. 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 and PAMPAS+, we selected three aggregate functions, i.e., *average*, *IDW* [31] 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.

8.2 Performance Evaluation

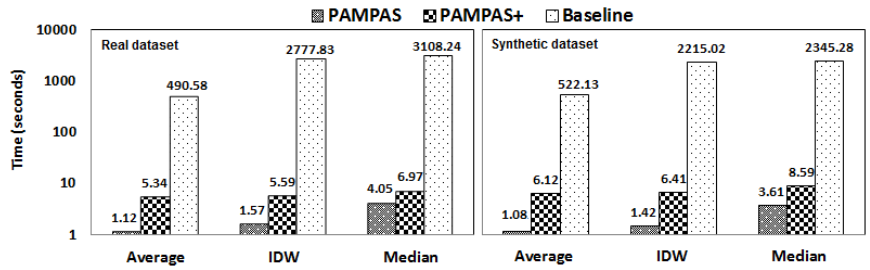

Fig. 8: Aggregation time of PAMPAS, PAMPAS+ and Baseline protocols with the real dataset (left) and the synthetic dataset (right)

Execution time. Figure 8 compares the aggregation time (in a logarithmic scale) of PAMPAS, PAMPAS+ and baseline protocols for the three considered functions with 191 thousand probes in Beijing (real dataset) and with 200 thousand probes in Stockton. The aggregation time is global, i.e., it includes

both the computation and communication time. The number of partitions was selected as the optimal value for the average function (i.e., 128 partitions). 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. The aggregation time in PAMPAS$^+$, although higher than in PAMPAS, is still small enough to respect the real-time requirement of the protocol. The increased time is expected since in PAMPAS$^+$ there is the additional cost of accessing the couple key from the Flash memory for each sample during the aggregation processes. Nevertheless, the fixed number of partitions (i.e., 128) used in these tests plays mostly in the disadvantage of PAMPAS$^+$. As we show in the scalability tests here below, the difference between PAMPAS and PAMPAS$^+$ remains acceptable in all cases when the number of partitions is tuned correctly. Also, the aggregation times are similar with the real and synthetic datasets for both PAMPAS and PAMPAS$^+$. However, in both cases the baseline protocol is much more costly (especially for complex aggregate functions) leading 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 and PAMPAS$^+$ are 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 9 shows the aggregation time for the three protocols for the average (top graph) and median (bottom graph) functions with medium and large number of users on both road networks. The number of partitions in this evaluation is selected as the optimal value for each case. Specifically, the number of partitions ranges from 128 to 512 and from 128 to 1024 in PAMPAS and PAMPAS$^+$ respectively depending of the data size and the aggregation function, i.e., a larger data size leads to a higher number of partitions as well as complex computations such as the IDW or median functions. The results confirm that only PAMPAS and PAMPAS$^+$ are scalable with respect to all the varying input parameters. In the worst case, the computation time of PAMPAS and PAMPAS$^+$ attains 14 seconds and 18 seconds respectively 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 the partitioning protocol of PAMPAS. Figure 10 (left) presents the partitioning computation time in PAMPAS for both Oldenburg and Stockton networks. A new partitioning can be computed

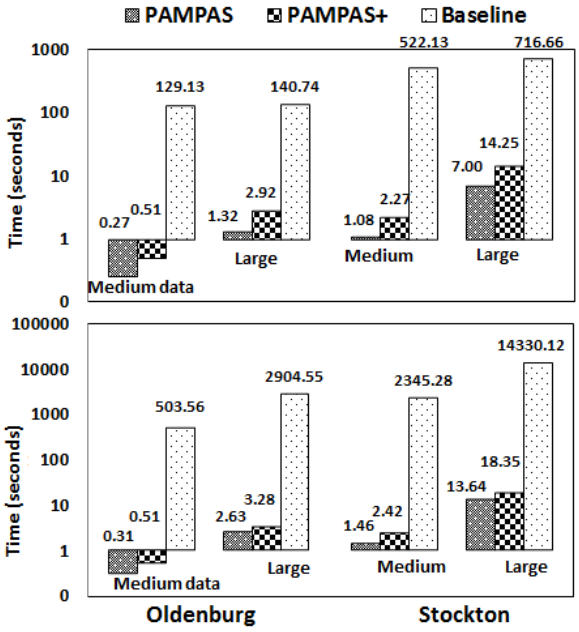

Fig. 9: Scalability of the PAMPAS, PAMPAS+ and Baseline protocols with Average function (top) and Median function (bottom)

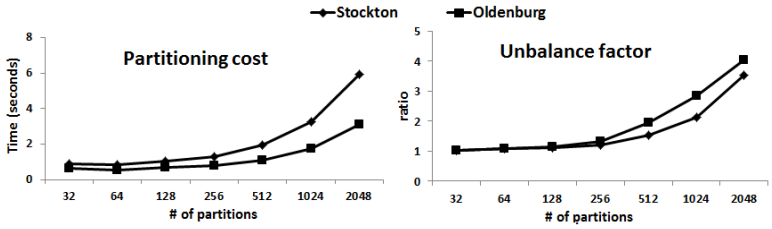

Fig. 10: The partitioning costs (left) and the partitioning unbalance factor (right) in PAMPAS with different number of partitions

in a few seconds by an SP[11]. This means that the checking and probes repartitioning 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 (see the detail in Section 8.3). 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.

[11] We note that in this paper we employed an optimized implementation of the base partitioning algorithm compared to the version used in [44]. While the general algorithm remains the same (see Algorithm 4), we optimized the number of Flash IOs through a better usage of the 30KB of RAM available for data processing at the SP side.

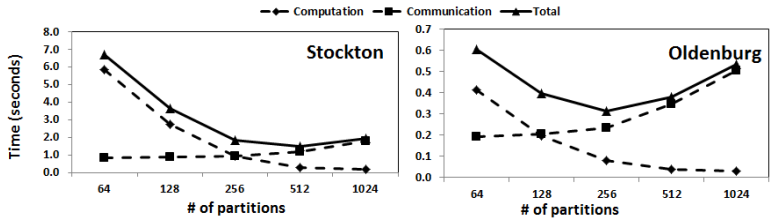

Fig. 11: Communication and computation costs of Median function in PAMPAS with different number of partitions and road networks

Figure 10 (right) indicates that the partitioning unbalance factor of PAMPAS, 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 11 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 aggregating 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 128 while the network partitioning is computed in just 2 seconds. The aggregation cost is partially shown in Figure 8 (left). For the sake of brevity and also due to the similarity of the results with the synthetic datasets, we omit here the details of the results with the real dataset.

8.3 Partitioning efficiency of PAMPAS$^+$ versus PAMPAS

In this section, we analyze the efficiency and effectiveness of the partitioning algorithm of PAMPAS$^+$ compared with the base algorithm. We compare the overall cost and the unbalance factor with different number of partitions for the two methods. Then, we detail the global partitioning cost by separating the CPU and the secondary memory access costs. Finally, we evaluate the parallel execution cost of the PAMPAS$^+$ partitioning method (see Section 6.3) with different number of probes.

Figure 12 shows the partitioning cost (left) and the unbalance factor (right) in PAMPAS and PAMPAS$^+$ for the Stockton large dataset. As expected, the partitioning algorithm of PAMPAS$^+$ is more costly but produces more balanced partitions especially for large numbers of partitions. Overall, both methods have reasonable partitioning cost and acceptable unbalance factor, which

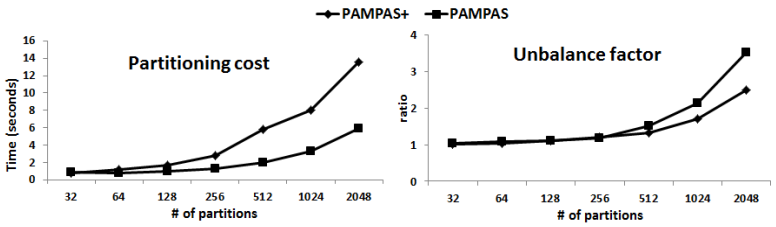

Fig. 12: The partitioning costs (left) and the partitioning unbalance factor (right) in PAMPAS and PAMPAS$^+$ with different number of partitions

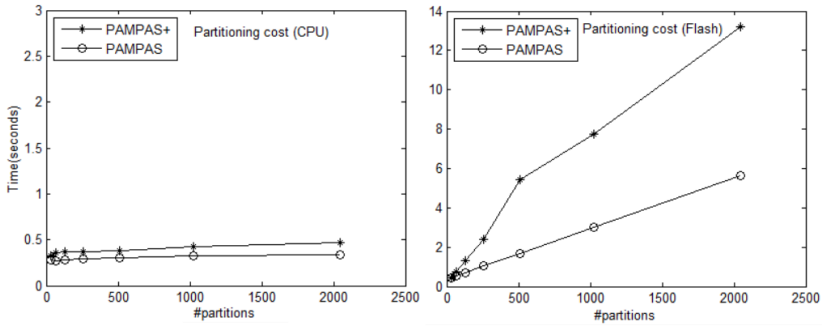

Fig. 13: The detailed partitioning costs - CPU (left) and Flash IOs (right) - in PAMPAS and PAMPAS$^+$ with different number of partitions

allows both algorithms to be employed in real-time in mobile participatory sensing applications. Given the higher cost but better quality of the partitioning algorithm of PAMPAS$^+$, this method will be preferred for large networks and high number of participants. Also, the aggregation process of PAMPAS$^+$ is more costly than in PAMPAS. Therefore, a more effective partitioning is required to reduce the aggregation cost (see Section 8.4), since a lower unbalance factor reduces the volume of fake messages and hence the communication cost while allowing to involve a larger number of SPs in the aggregation phase.

For a better understanding of the behavior of the two partitioning methods, we detail in Figure 13 the CPU cost (left) and the IO cost (right). We can observe that in terms of CPU cost the difference is only marginal between the two algorithms. The performance difference is largely explained by the IO cost. The recursive nature of the partitioning algorithm in PAMPAS$^+$ requires repeated reads/writes from/to the Flash storage especially given the low RAM memory limit imposed in the protocol (i.e., 30KB), which explains the larger number of IO compared with the base method.

Parallel partitioning. As described in Section 6.3, the partitioning algorithm of PAMPAS$^+$ can be easily parallelized. We validated experimentally the parallel execution of the partitioning algorithm. Figure 14 shows the partitioning cost for different numbers of partitions for the large Stockton dataset when the partitioning process is executed by a different number of SPs varying

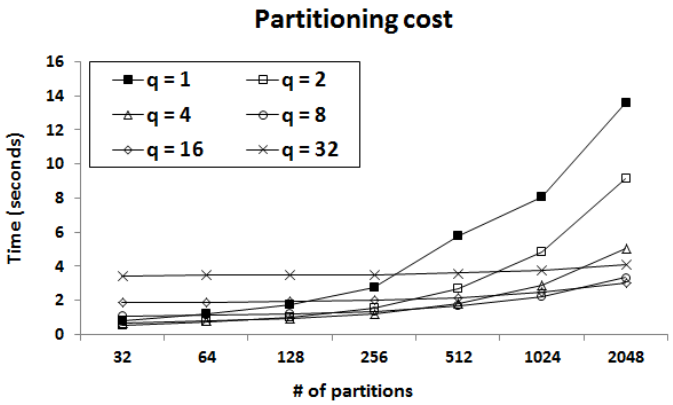

Fig. 14: The parallel partitioning costs in PAMPAS$^+$ with different number of partitions

from 1 to 32. The first observation is that delegating the process to several SPs can drastically reduce the partitioning cost for large number of partitions (i.e., above 256). At the same time, increasing the number of worker SPs above a certain limit results is augmenting the partitioning cost. The reason is twofold. First, the communication cost increases with increased number of SPs. Second, when a large number of SPs are used, the coordinator SP (see Section 6.3) has to do more work at the beginning of the process before delegating the tasks to the other SPs.

Typically, a number of 4 to 8 SPs offers the best performance in our experiments. More importantly, if parallelized, the partitioning algorithm in PAMPAS$^+$ becomes faster than the equivalent algorithm in PAMPAS.

8.4 Aggregation efficiency of PAMPAS$^+$ versus PAMPAS

Similar to PAMPAS, the number of partitions plays an important role in global aggregation time of PAMPAS$^+$ and has to be tuned correctly to obtain low aggregation cost. Figure 15 shows the variation of the communication and computation costs with different number of partitions for the median function and the large Stockton and Oldenburg datasets. As in the base protocol, increasing the number of partitions decreases the computation cost but augments the communication cost. However, due to a more effective partitioning (see Figure 12 (right)), the increase of the communication cost is slower in PAMPAS$^+$ then in PAMPAS. This also makes that, in some cases, the optimal number of partitions (i.e., minimizing the overall cost) to be higher in PAMPAS$^+$ then in PAMPAS for the same dataset and aggregation function. For instance, PAMPAS reaches the best performance with 256 partitions while PAMPAS$^+$ requires 512 partitions for the large Oldenburg dataset and the median function.

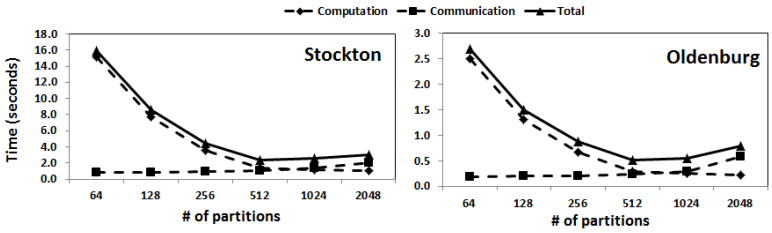

Fig. 15: Communication and computation costs of Median function in PAMPAS+ with different number of partitions and road networks

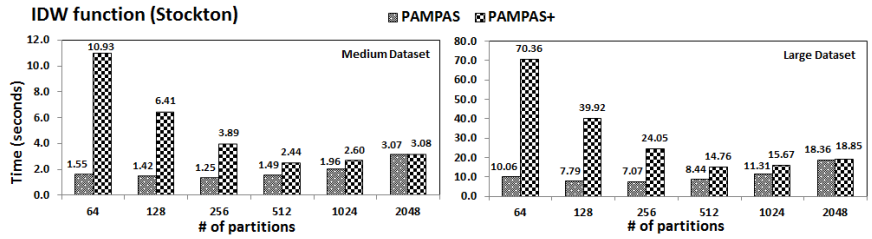

Fig. 16: Overall costs of IDW function in PAMPAS and PAMPAS+ with different number of partitions

Figure 16 compares the overall aggregation cost of PAMPAS and PAMPAS+ with different number of partitions for the IDW function and the medium and large Stockton datasets. In general, both PAMPAS and PAMPAS+ offer a reasonable cost in all cases. For PAMPAS+, the aggregation time drops rapidly with the increase of the number of partitions before attaining the near-optimal value. Compared with PAMPAS, the aggregation cost of PAMPAS+ is much higher for a small number of partitions, while the cost difference reduces to practically none for large number of partitions. If we compare the near-optimal costs of the two protocols, we can observe that the aggregation time of PAMPAS+ is about twice larger. However, the aggregation time of PAMPAS+ remains sufficiently small to be still considered as a real-time protocol even for very large number of users (i.e., 1.35 millions).

8.5 Discussion

This section wraps up the experimental evaluation. The first observation is that the existing data aggregation protocols built on secure hardware [41, 42] have not been designed for applications requiring real-time processing such as in the area of participatory sensing and therefore cannot be used in this context. On the other hand, both protocols proposed in this paper are secure, efficient, scalable, accurate and generic with respect to the aggregation function. PAMPAS is extremely efficient and builds its efficiency on the reasonable premise of the inviolability of secure hardware. PAMPAS+ goes one

significant step forward from the security point of view and trades efficiency to increase the security level of the aggregation protocol. Thus, PAMPAS$^+$ offers stronger privacy guarantees to the participants since it protects their privacy even if some SPs are successfully compromised by their owner (see Section 6.1). PAMPAS$^+$ minimizes the information leakage by the corrupted SPs and hence drastically reduces the benefit of a successful lab attack (see Section 7.3). The price to pay for the increased security level is larger aggregation time. Also, PAMPAS$^+$ requires an additional couple-key exchange phase in which each pair of SPs exchanges a secret key, but this process does not have to meet the real-time constraint imposed to the main aggregation process.

Selecting the partitioning algorithm. Both proposed protocols include a specific partitioning algorithm. Nevertheless, we note that both the linear partitioning algorithm of PAMPAS and the recursive binary partitioning of PAMPAS$^+$ can be interchanged in the two protocols. For the sake of clarity we did not provide any results for the aggregation cost of the two protocols using interchanged partitioning methods. The rationale is simple. Using the binary partitioning in PAMPAS leads to decrease the aggregation cost of about 9.64% (on average), but at the same time significantly increases the cost of the partitioning phase. Since PAMPAS is already very efficient in the aggregation phase using the linear partitioning, we believe that a better partitioning is superfluous in this case. Oppositely, using the linear partitioning in PAMPAS$^+$ leads to increase the aggregation cost of about 12.53% (on average), but decreases the cost of the partitioning phase. Given that the aggregation cost is significantly larger in PAMPAS$^+$ than in PAMPAS, any increase could be perceived as being too much and hence the binary partitioning should be preferred in this case in our opinion. We also recall that the binary partitioning can be executed in parallel which can drastically reduce the execution time but requires several SPs to do the computation.

Selecting the query spatial units. In PAMPAS, the probe groups are built on top of the spatial units requiring each group to include entirely the spatial units it covers. This is done to increase the data aggregation efficiency as explained in Sections 4 and 5. However, the way the spatial units are chosen for a query can have an impact on the query computation efficiency. Choosing units of smaller size is better since a finer decomposition of the observed space may allow for a more balanced group partitioning (especially for highly skewed spatial distribution of participants), which in turn increases the aggregation efficiency (since less fake tuples need to be generated to balance the data in the groups). On the other hand, smaller units may increase the probability of filtering out more aggregation results corresponding to the units with low number of participants (as discussed in Section 7.1). In conclusion, while PAMPAS is agnostic to the selection of the query spatial units, their configuration can impact its efficiency and to some extent its accuracy. Studying this issue is an interesting perspective of future work.

Dealing with parallel MPS queries. In this paper, our focus is on having a framework as generic as possible from the viewpoint of possible types of spatio-temporal aggregation and then providing efficient and secure proto-

cols to achieve such aggregations. Therefore, the spatial units are defined per query (see Section 3) and thus can change for different queries. Maintaining the partitioning for each query leads to a higher overhead, but this is the cost for privacy and flexibility. One solution to reduce this overhead is to consider the same spatial units for different queries. However, this raises a number of issues. First, sharing the same spatial unit decomposition would, obviously, make sense only if the physical referential space is the same (e.g., the geographical area of a city). However, even if the physical space is the same, the type of movement can completely change the spatial unit representation (i.e., road segments for vehicle movements versus grid cells for pedestrian movements). So considering the same spatial units is possible only for the same type of movement. Second, assuming that the previous elements are verified, there is the question regarding the participants to the queries. If several queries are run in parallel, the users may opt to participate to only part of them. Hence, for any two parallel queries, there is no guarantee that the respective sets of participants are identical. However, the distribution of the participants over the spatial units is an essential parameter of the partitioning algorithm. Without knowing the accurate spatial distribution, the system can no longer guarantee the optimum privacy level. Also, verifying the intersection between two lists of participants in a distributed and privacy-preserving way is not trivial (and may even exceed the cost of partitioning). We should finally note that in PAMPAS the partitioning cost is low (a few seconds in most cases) and can be effectively done in parallel by only a few SPs (see Section 8.3). Moreover, its cost is much lower than the aggregation cost which dominates the overall cost of PAMPAS since the aggregation involves a few hundred SPs at each iteration as indicated by our experimental evaluation. However, optimizing the partitioning cost with different queries executed in parallel is another interesting perspective of future work.

Boosting performance. Finally, 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 and 18 seconds to less than 7 seconds and 10 seconds in PAMPAS and PAMPAS$^+$ respectively. 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, our protocols minimize this type of problem thanks to their design and 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 (i.e., MCU-like low power CPU, 30KB of RAM and 200Kbps communication bandwidth). Moreover, the computation effort is inversely proportional to the probability to be picked.

9 Conclusion

This paper proposes PAMPAS and PAMPAS$^+$, two related privacy-aware mobile participatory sensing systems based on a distributed architecture and personal secure hardware. This combination allows PAMPAS and PAMPAS$^+$ to achieve the same level of privacy as cryptographic solutions without having to sacrifice generality, scalability, and accuracy. Moreover, PAMPAS$^+$ protects the participants even against sophisticated hardware attacks without having to sacrifice too much its efficiency. The proposed aggregation solutions are, to the best of our knowledge, the first proposal of distributed protocols that are secure, efficient, scalable and generic w.r.t. the aggregation computation and that fit 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 solutions.

References

1. Allard T, Nguyen B, Pucheral P (2014) METAP: revisiting privacy-preserving data publishing using secure devices. Distributed and Parallel Databases 32(2):191–244
2. Andrés ME, Bordenabe NE, Chatzikokolakis K, Palamidessi C (2013) Geo-indistinguishability: Differential privacy for location-based systems. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM, New York, NY, USA, CCS '13, pp 901–914, DOI 10.1145/2508859.2516735, URL http://doi.acm.org/10.1145/2508859.2516735
3. ARM (2009) ARM Security Technology - Building a Secure System using TrustZone Technology. ARM Technical White Paper
4. Baumann A, Peinado M, Hunt G (2014) Shielding applications from an untrusted cloud with haven. In: OSDI, pp 267–283
5. Brinkhoff T (2002) A framework for generating network-based moving objects. GeoInformatica 6(2):153–180
6. Brown JWS, Ohrimenko O, Tamassia R (2013) Haze: privacy-preserving real-time traffic statistics. In: ACM SIGSPATIAL, pp 540–543
7. Cao Y, Yoshikawa M, Xiao Y, Xiong L (2017) Quantifying differential privacy under temporal correlations. In: Data Engineering (ICDE), 2017 IEEE 33rd International Conference on, IEEE, pp 821–832
8. Chatzikokolakis K, Palamidessi C, Stronati M (2015) Location privacy via geo-indistinguishability. ACM SIGLOG News 2(3):46–69, DOI 10.1145/2815493.2815499, URL http://doi.acm.org/10.1145/2815493.2815499
9. Chow CY, Mokbel MF, Aref WG (2009) Casper*: Query processing for location services without compromising privacy. ACM Trans

Database Syst 34(4):24:1–24:48, DOI 10.1145/1620585.1620591, URL http://doi.acm.org/10.1145/1620585.1620591
10. Cornelius C, Kapadia A, Kotz D, Peebles D, Shin M, Triandopoulos N (2008) AnonySense: Privacy-aware people-centric sensing. In: MobiSys
11. Damiani ML (2014) Location privacy models in mobile applications: conceptual view and research directions. GeoInformatica 18(4):819–842
12. Damiani ML, Bertino E, Silvestri C (2010) The probe framework for the personalized cloaking of private locations. Trans Data Privacy 3(2):123–148, URL http://dl.acm.org/citation.cfm?id=1824401.1824404
13. D'Hondta E, Stevens M, Jacobs A (2013) 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
14. Douceur JR (2002) The sybil attack. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems, Springer-Verlag, London, UK, UK, IPTPS '01, pp 251–260, URL http://dl.acm.org/citation.cfm?id=646334.687813
15. Drosatos G, Efraimidis PS, Athanasiadis IN, Stevens M (2012) A privacy-preserving cloud computing system for creating participatory noise maps. In: COMPSAC, pp 581–586
16. Faezipour M, Nourani M, Saeed A, Addepalli S (2012) Progress and challenges in intelligent vehicle area networks. Magazine Communications of the ACM 55(2):90–100
17. Ganti RK, Pham N, Tsai YE, Abdelzaher TF (2008) PoolView: Stream privacy for grassroots participatory sensing. In: SenSys
18. Gao H, Liu CH, Wang W, Zhao J, Song Z, Su X, Crowcroft J, Leung KK (2015) A survey of incentive mechanisms for participatory sensing. IEEE Comm Surveys and Tutorials 17(2):918–943
19. Ghinita G, Damiani ML, Silvestri C, Bertino E (2016) Protecting against velocity-based, proximity-based, and external event attacks in location-centric social networks. ACM Trans Spatial Algorithms Syst 2(2):8:1–8:36, DOI 10.1145/2910580, URL http://doi.acm.org/10.1145/2910580
20. Goel P, Kulik L, Ramamohanarao K (2016) Privacy-aware dynamic ride sharing. ACM Trans Spatial Algorithms Syst 2(1):4:1–4:41, DOI 10.1145/2845080, URL http://doi.acm.org/10.1145/2845080
21. González J, Hölzl M, Riedl P, Bonnet P, Mayrhofer R (2014) A practical hardware-assisted approach to customize trusted boot for mobile devices. In: Chow SSM, Camenisch J, Hui LCK, Yiu SM (eds) Information Security, Springer International Publishing, pp 542–554
22. Hoh B, Iwuchukwu T, Jacobson Q, Work D, Bayen AM, Herring R, Herrera JC, Gruteser M, Annavaram M, Ban J (2012) Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines. IEEE Tran on Mobile Computing 11(5):849–864
23. Huang KL, Kanhere SS, Hu W (2010) Preserving privacy in participatory sensing systems. Comput Commun 33(11):1266–1280, DOI 10.1016/j.comcom.2009.08.012, URL

http://dx.doi.org/10.1016/j.comcom.2009.08.012
24. Jain N, Mishra S, Srinivasan A, Gehrke J, Widom J, Balakrishnan H, Çetintemel U, Cherniack M, Tibbetts R, Zdonik SB (2008) Towards a streaming sql standard. In: PVLDB 1(2), pp 1379–1390
25. Lallali S, Anciaux N, Popa IS, Pucheral P (2017) Supporting secure keyword search in the personal cloud. Information Systems 72:1 – 26, DOI https://doi.org/10.1016/j.is.2017.09.003, URL http://www.sciencedirect.com/science/article/pii/S0306437916303891
26. Li M, Zhu L, Zhang Z, Xu R (2017) Achieving differential privacy of trajectory data publishing in participatory sensing. Inf Sci 400(C):1–13, DOI 10.1016/j.ins.2017.03.015, URL https://doi.org/10.1016/j.ins.2017.03.015
27. Li Q, Cao G (2012) Efficient and privacy-preserving data aggregation in mobile sensing. In: IEEE ICNP
28. Liu R, Cao J, VanSyckel S, Gao W (2016) Prime: Human-centric privacy measurement based on user preferences towards data sharing in mobile participatory sensing systems. In: 2016 IEEE International Conference on Pervasive Computing and Communications (PerCom), pp 1–8, DOI 10.1109/PERCOM.2016.7456518
29. Maruseac M, Ghinita G, Trajcevski G, Scheuermann P (2017) Privacy-preserving detection of anomalous phenomena in crowdsourced environmental sensing using fine-grained weighted voting. Geoinformatica 21(4):733–762, DOI 10.1007/s10707-017-0304-3, URL https://doi.org/10.1007/s10707-017-0304-3
30. de Montjoye YA, Hidalgo CA, Verleysen M, Blondel VD (2013) Unique in the crowd: The privacy bounds of human mobility. Scientific reports 3
31. Nittel S, Whittier JC, Liang Q (2012) Real-time spatial interpolation of continuous phenomena using mobile sensor data streams. In: ACM SIGSPATIAL, pp 530–533
32. Pan J, Sandu-Popa I, Borcea C (2017) Divert: A distributed vehicular traffic re-routing system for congestion avoidance. IEEE Transactions on Mobile Computing 16(1):58–72, DOI 10.1109/TMC.2016.2538226
33. Penza M (2014) Cost action TD1105: New sensing technologies for environmental sustainability in smart cities. In: IEEE SENSORS
34. Piro C, Shields C, Levine BN (2006) Detecting the sybil attack in mobile ad hoc networks. In: 2006 Securecomm and Workshops, pp 1–11
35. Popa RA, Blumberg AJ, Balakrishnan H, Li FH (2011) Privacy and accountability for location-based aggregate statistics. In: CCS, pp 653–666
36. Priebe C, Vaswani K, Costa M (2018) Enclavedb - a secure database using sgx. IEEE, URL https://www.microsoft.com/en-us/research/publication/enclavedb-a-secure-database-using-sgx/
37. Quercia D, Leontiadis I, Mcnamara L, Mascolo C, Crowcroft J (2011) Spotme if you can: Randomized responses for location obfuscation on mobile phones. In: ICDCS, pp 363–372
38. Sabt M, Achemlal M, Bouabdallah A (2015) Trusted execution environment: What it is, and what it is not. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol 1, pp 57–64, DOI 10.1109/Trustcom.2015.357

39. Shi J, Zhang R, Liu Y, Zhang Y (2010) PriSense: Privacy-preserving data aggregation in people-centric urban sensing systems. In: IEEE INFOCOM
40. Thiagarajan A, Ravindranath L, LaCurts K, Madden S, Balakrishnan H, Toledo S, Eriksson J (2009) Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In: ACM SenSys, pp 85–98
41. To QC, Nguyen B, Pucheral P (2014) Privacy-preserving query execution using a decentralized architecture and tamper resistant hardware. In: EDBT, pp 487–498
42. To QC, Nguyen B, Pucheral P (2016) Private and scalable execution of sql aggregates on a secure decentralized architecture. ACM Trans Database Syst 41(3):16:1–16:43, DOI 10.1145/2894750, URL http://doi.acm.org/10.1145/2894750
43. Ton-That DH, Sandu-Popa I, Zeitouni K (2015) PPTM: Privacy-aware participatory traffic monitoring using mobile secure probes. In: IEEE MDM, demo paper
44. Ton-That DH, Sandu-Popa I, Zeitouni K, Borcea C (2016) PAMPAS: Privacy-aware mobile participatory sensing using secure probes. In: Proceedings of the 28th International Conference on Scientific and Statistical Database Management, ACM, SSDBM '16, pp 4:1–4:12, DOI 10.1145/2949689.2949704, URL http://doi.acm.org/10.1145/2949689.2949704
45. Wang G, Wang B, Wang T, Nika A, Zheng H, Zhao BY (2016) Defending against sybil devices in crowdsourced mapping services. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, ACM, New York, NY, USA, MobiSys '16, pp 179–191, DOI 10.1145/2906388.2906420, URL http://doi.acm.org/10.1145/2906388.2906420
46. Wang L, Yang D, Han X, Wang T, Zhang D, Ma X (2017) Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation. In: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, WWW '17, pp 627–636, DOI 10.1145/3038912.3052696, URL https://doi.org/10.1145/3038912.3052696
47. Yuan J, Zheng Y, Xie W, Xie X, Sun G, Huang Y (2010) T-drive: driving directions based on taxi trajectories. In: SIGSPATIAL, pp 99–108