Data Verification and Privacy-respecting User Remuneration in Mobile Crowd Sensing

Stylianos Gisdakis, Thanassis Giannetsos, Panos Papadimitratos
Networked Systems Security Group
KTH Royal Institute of Technology, Stockholm, Sweden
{*gisdakis*, *athgia*, *papadim*}@kth.se

Abstract—The broad capabilities of current mobile devices have paved the way for Mobile Crowd Sensing (MCS) applications. The success of this emerging paradigm strongly depends on the quality of received data which, in turn, is contingent to mass user participation; the broader the participation, the more useful these systems become. This can be achieved if users are gratified for their contributions while being provided with strong guarantees for the security and the privacy of their sensitive information. But this very openness is a double-edge sword: any of the participants can be adversarial and pollute the collected data in an attempt to degrade the MCS system output and, overall, its usefulness. Filtering out faulty reports is challenging, with practically no prior knowledge on the participants trustworthiness, dynamically changing phenomena, and possibly large numbers of compromised devices. This work presents a holistic framework that can assess user-submitted data and sift malicious contributions while offering adequate incentives to motivate users to submit better quality data. With a rigorous assessment of our system's security and privacy protection complemented by a detailed experimental evaluation, we demonstrate its accuracy, practicality and scalability. Overall, our framework is a comprehensive solution that significantly extends the state-of-the-art and can catalyze the deployment of MCS applications.

Index Terms—Mobile Crowd Sensing, Security, Privacy, Incentive Mechanisms, Homomorphic Encryption

1 Introduction and Background

Mobile Crowdsensing [1, 2] (MCS) leverages the proliferation of modern sensing-capable devices to build a wide-scale information collection network that can provide insights for practically, *anything*, from *anywhere* and at *anytime*. Emerging applications range from environmental monitoring [3, 4], intelligent transportation [5, 6, 7] to assistive health-care [8, 9] and public safety [10, 11]. Overall, MCS will lead to a new understanding of our environment, thus, creating added value for the participating users.

However, MCS systems entail a number of challenges. First, users are increasingly concerned with the *security* and the *privacy* of their sensitive information; revelations of mass surveillance [12] aggravate such anxieties. Users are expected to contribute sensed data tagged with spatio-temporal information (e.g., time and/or location). Misusing such information could reveal sensitive user-specific attributes including their whereabouts, personal activities, health condition, etc. [13]. Therefore, *user privacy is vital for any successful and large-scale deployment of MCS systems*.

A second challenge is user participation. Simply put, MCS systems will be successful if users embrace them in great numbers and provide a sufficient influx of contributions (i.e., information) to the system. Unfortunately, users may opt out at any time for any reason. In fact, user participation is driven either by *intrinsic* (i.e., contributing to applications targeting societal welfare) or *extrinsic* (i.e., reward-based) motivation. The latter one requires incentive mechanisms that can reinforce user motivation [14]. The way incentives are materialized largely depends on the MCS application and the involved stakeholders. However, the common denominator is the provision of incentives in a privacy-preserving manner: users should be gratified without being associated to the data they contribute to the system. Nonetheless, this sets the challenge ahead: *assessing the overall contributions made by anonymous (for privacy-protection) users.*

Besides recruiting users, MCS systems must also *retain* them by offering accurate and useful information. Systems relying on, presumably, trusted information sources (e.g., traffic cameras and/or environmental sensors deployed and managed by the authorities) can offer accurate information but at a high deployment cost and with limited spatial coverage. On the contrary, MCS systems collect information not necessarily from trustworthy sources; they opt for contributions from anyone possessing a sensing-capable device. This very openness, however, renders them vulnerable to malicious users that can *pollute* the data collection process, thus, manipulating (or even dictating) the system output; thwarting such attacks is the main challenge ahead.

Many works aim to secure the data collection [15, 16, 17, 18] from unauthorized adversaries. However, registered MCS users, i.e., insiders, might also misbehave and target the system. More specifically, compromised devices possessing valid cryptographic credentials, can still (easily) pollute the MCS data. Evicting malicious users is possible [19] but it mandates fine-grained and node-specific identification of offending behavior. Unfortunately, this is not straightforward especially in the presence of intelligent, numerous and cooperating adversaries. Relying on reputation schemes [20, 21] is not an option: users must *build* their reputation from the trustworthiness of the data they contribute to the MCS system. But assessing data trustworthiness is exactly the problem at hand: in other words, a circular dependency. What we need is to *assess and sift faulty data without any assumption on the trustworthiness of their source*. Designing such

mechanisms is not easy: it requires fusing (contradictory) data, originating from untrustworthy sources describing dynamic and uncertain phenomena for which we may not have any prior statistical description or model.

Motivation & Contributions: The problems of *privacy-protection*, *data-trustworthiness* and *incentive-provision* have been studied separately in the literature: schemes either focus on the privacy and security without considering data pollution attacks [15, 16, 17, 18]; or they facilitate mass participation by linking incentives to users' contributions and, thus, without considering user privacy [22, 23]. Finally, although SHIELD [24], the state-of-the-art data-trustworthiness framework for MCS systems, can efficiently detect and mitigate pollution attacks, it does not consider extrinsic incentive features. Nonetheless, user remuneration and data verification are strongly connected: malicious users should not be rewarded but instead detected and evicted, whereas benign users should be gratified on the basis of the *quantity* and the *quality* of their contributions.

This work, systematically addresses all the aforementioned challenges by building on SHIELD and extending it into a holistic framework offering: (*i*) a data-trustworthiness framework that can assess user-submitted data and sift malicious contributions, (*i*) a privacy-preserving incentive provision mechanism that is resilient to dishonest users, while (*iii*) holding them accountable for their actions, (*iv*) under weakened assumptions on the trustworthiness of the MCS system entities.

The rest of the paper is organized as follows. In Sec. 2, we first describe the system and adversarial models for our scheme. We then provide an overview of our data verification protocol, followed by a detailed description of its core components (Sec. 3). Sec. 4 presents the user remuneration protocol along with a detailed analysis of the security and privacy properties achieved by our rewarding scheme. The experimental setup, used to evaluate our system, along with the performance results are presented in Sec. 6 and 7, respectively, before we conclude the paper in Sec. 8.

2 PRELIMINARIES

2.1 System Model

Our secure and privacy-preserving MCS system comprises the following entities:

Users: Individuals operating mobile devices (e.g., smartphones, tablets and smart vehicles) equipped with embedded sensors (e.g., inertial and proximity sensors, cameras, microphones and gyroscopes) and navigation modules (e.g., GPS). User mobile devices collect and report sensory data to the MCS infrastructure over any available network (e.g., 3/4G, WiFi). Users (devices) can also query the results of sensing tasks. In case the chosen remuneration method is *bitcoin payments* (Sec. 4), each participating user possess (or generates) a bitcoin address. In the rest of the paper, we refer to users and their mobile devices or clients interchangeably.

Task Initiators (TI): Organizations or individuals initiating data collection campaigns by recruiting users and distributing descriptions of sensing tasks to them [25].

Identity & Credential Management Infrastructure: It supports sensing tasks by registering users, providing cryptographic credentials and enabling or offering Authentication, Authorization and Access Control services. We consider the following credential management entities:

• **Group Manager (GM):** It is responsible for registering and authenticating user devices to sensing tasks. Moreover, the GM enables user privacy protection, during the remuneration protocol, leveraging homomorphic encryption .

Our system utilizes the Benaloh's additive and randomized homomorphic encryption scheme [26]: given a private/public key pair $(p = a, P = g^a)$ the encryption of a message m, with $m \in \mathbb{Z}_p$, is $H_e(m, r) = (g^r, g^{r \cdot \alpha} \cdot h^m)$; where g, h are generators of a cryptographic group G of order q. The additive homomorphic property of the scheme yields that $H_e(m_1, r_1) \cdot H_e(m_2, r_2) = H_e(r_1 + r_2, m_1 + m_2)$, where the multiplication is component-wise. Decryption is $H_d(C_1, C_2) = DiscreteLog_h(C_2/C_1^{\alpha})$.

• **Pseudonym Certification Authority (PCA):** It provides anonymized ephemeral credentials, termed *pseudonyms*, to users. Pseudonymous authentication ensures the integrity and the authenticity of all information submitted by the clients. To ensure *unlinkability*, devices can obtain multiple pseudonyms from the PCA.

We require that the credential management system is *Sybil-proof*: no registered user can obtain multiple identities and credentials valid simultaneously. We assume our system operates on top of a credential management such as SPPEAR [19], the state-of-the-art security and privacy-preserving architecture which offers such properties.

Reporting Service (RS): The RS exposes the interfaces that allow registered users to submit their data and query the results of a sensing task. Furthermore, it also offers the mechanism that enables participants to claim their rewards without being linked to their contributed data. Data verification is performed on this entity: we first assess and remove invalid, faulty data and, then, remunerate users based on the *level* of their participation and the *quality* of information they provide. The RS retrieves (from the GM) the public key, P, of the additive homomorphic scheme instantiated by the GM.

Our system is designed according to the *separation-of-duties principle* [27]: each entity is given the minimum information required to execute a designated task. This way, single system entities have limited access to user information and thus they cannot breach user privacy (Sec. 5.3).

The *area of interest* of a sensing task is the locality within which participating users must contribute data. The area of interest can be defined either explicitly (e.g., coordinates forming polygons on maps) or implicitly (through annotated geographic areas, e.g., Stockholm). In any case, the area of interest is divided into *spatial units*: homogeneous, with respect to the sensed phenomenon, areas [28]. More specifically, the sensed phenomenon has temporal but not significant spatial variations within a spatial unit. Defining spatial units largely depends on the MCS application and, hence, on the underlying monitored phenomenon. For instance, for traffic monitoring systems (e.g., [6, 29]), road links (i.e., road segments between two junctions) serve as spatial units. Similarly, for public transport MCS applications (e.g., [30]) spatial units can be individual bus, metro and train lines. Finally, areas around pollution points (e.g., factories) are spatial units [31] for environmental monitoring tasks.

Users submit to the RS streams of measurements over a

time interval, t, specified in the MCS campaign [25]. These data are submitted in successive *reports*, each containing n measurements, v_i:

$$r_i = \{[v_1, v_2, v_3, ..., v_n] \,||\, t \,||\, loc \,||\, \sigma_{PrvKey} \,||\, Ps\}$$

The number of measurements, n, is subject to the application and is specified by the campaign administrator. Depending on the phenomenon to be monitored, collected data should be of certain quality level which can be influenced by the sampling frequency (other parameters include the overall sensing time, the location accuracy, etc.); higher n may yield a better perception of the monitored phenomenon. loc is the device's location and σ_{PrvKey} is a digital signature with some private key. The corresponding public key is included in the certificate Ps (obtained by the PCA). User-RS communication is done over an authenticated TLS channel.

2.2 Threat Model

We assume malicious users (clients) participating in sensing campaigns that try to compromise the system: having obtained valid credentials, they submit authenticated yet faulty reports to the RS. We do not consider only human operators with malevolent intentions, but also compromised devices (clients), e.g., running a rogue version of the MCS application. Each compromised client can *pollute* the data collection process. This will *distort* the MCS system perception of the sensed phenomenon, and thus degrade the usefulness of the campaign. For instance, consider traffic monitoring campaigns, during which users submit their velocity to the RS: malicious users could try to impose a false perception on the congestion levels of the road network, thus, claiming, e.g., traffic jams or accidents.

We allow adversaries to submit values (arbitrarily) different than the (truthful) ones characterizing the sensed phenomenon. Let V_{a_i} be the adversary-free system output for a spatial unit i, and V_{r_i} the system output in the presence of adversaries. The objective of the adversaries is to create a V_{r_i} that deviates from V_{a_i}. In extremis, the adversaries will try to maximize the distortion:

$$max\,\{|V_{r_i} - V_{a_i}|\}$$

Adversaries can act *individually* or *collectively*. Here, we focus on the latter, i.e., *coordinated attacks*, as they can have far graver impact on the system. We assume adversaries following the same strategy, i.e., submitting reports drawn from the same set of values, or even optimally selecting where to inject faulty reports: they *spoof* their (device) location and submit reports for regions they are not physically present.

Our aim is the design of a system which, in the presence of such adversaries, can safeguard the accuracy of the RS. Simply put, minimize the distortion imposed by adversaries:

$$min\,\{|V_{r_i} - V_{a_i}|\}$$

Adversaries may have a strong motive to manipulate the incentive provision mechanism. For instance, leveraging their (for privacy protection) anonymity, they could try to obtain inordinate, to their contributions, rewards (more details on specific types of misbehavior can be found in Sec. 5.1). It is therefore imperative for the remuneration mechanism to be able to detect and isolate such malicious behavior. With respect to user privacy, our goal is to ensure that rewards cannot be linked to a particular user even if she has accepted multiple rewards for the data she has provided over time.

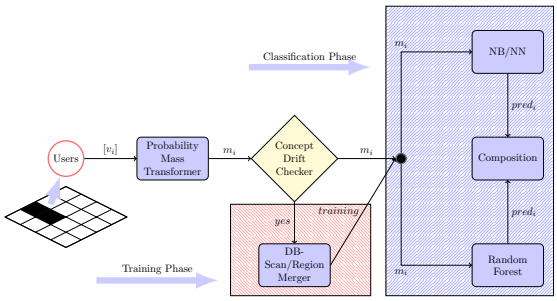


Fig. 1: System Overview

Finally, we also consider *honest-but-curious* infrastructure entities; they execute the protocols correctly but are curious to extract user personal information by, e.g., trying to profile them and reveal their identities by (possibly) colluding with other entities in the system.

3 Data Verification Framework

In this section, we start by looking how to examine the *quality of information* that each participant provides. This allows the system not only to increase the quality of service offered back to the MCS users but also to fairly compensate them (Sec. 4): benign users should be gratified based on the quantity and, more importantly, the quality of their contributions.

The RS preprocesses incoming user reports and then acts in three phases (Fig. 1): a) bootstrapping, b) region merging and training and c) classification. During the bootstrapping, user reports are classified as *inlying* or *outlying* essentially corresponding to non-faulty and faulty ones. As the system has no a-priori knowledge of what makes reports inliers or outliers, this phase explores their innate structure. Reports are classified not as raw data but as evidence: each incoming report is processed; i.e., it is transformed into a mass function (Sec. 3.1), and a feature vector, one for each report, is created and classified (Sec. 3.2). Bootstrapping is run separately for each spatial unit (recall: measurements of honest users follow the same distribution within a spatial unit).

Subsequently, the system examines the spatial characteristics of the sensed phenomenon and merges neighboring spatial units into larger regions. To do this, our system derives an empirical distribution from all the data of all inlying reports within a spatial until. Then, it examines the statistical similarity of the empirical distributions of neighboring spatial units and merges them if they are deemed similar. This way *region merging* (Sec. 3.3) creates larger homogeneous (with respect to the phenomenon) geographical regions. For each of these regions, *training* is performed: it is similar to the bootstrapping phase except that it is performed for the reports of all the spatial units that now comprise a region.

At the third phase, the output of the training and region merging phase (i.e., user reports labeled as inliers and outliers) is fed to an *ensemble* of classifiers (Sec. 3.4). This ensemble, a supervised learning mechanism, leverages the previously acquired training data in order to classify subsequent user reports. A different ensemble is created for each of the regions that was extracted during the region merging phase. Then, each new incoming report is assessed by the ensemble of classifiers; the individual decisions of the classifiers are then combined into a majority-based decision, which classifies the report as inlying or outlying.

The statistical properties of the sensed phenomena can change over time in unforeseen ways. Such events, known as *concept drifts* [32], deteriorate the performance of any classification model. Our system leverages a triggering mechanism that detects and adapts to such changes (Sec. 3.5).

3.1 Handling Evidence (pre-processing)

The RS is an agent that *reasons* on the actual value of the sensed phenomenon. To do this, it relies on multiple sources of evidence (i.e., user reports). For instance, the RS has to decide whether the congestion level of a street is "high", based on the velocities that participating users report, or whether the temperature of an area is within the interval $10\,^{\circ}\mathrm{C} - 11\,^{\circ}\mathrm{C}$, based on temperature measurements.

This is a *decision making* problem. The canonical approach for such problems is *Bayesian Inference* (BI): computing the posterior probability of a hypothesis given the available supporting evidence and its prior probability. Nonetheless, defining prior probabilities is an obstacle. For example, it is hard to estimate the prior probability of hypotheses stating that the road is congested or that the temperature (of an area) is within some interval. Instead, we have to handle *uncertainty*: reports stating that the temperature is within an interval α does not preclude another interval β (as is the case for BI). We leverage the *Dempster-Shafer Theory* (DST) [33] that allows reasoning about uncertainty.

3.1.1 Use of Dempster-Shafer Theory

Θ is the exhaustive set of hypotheses about the actual value of the sensed phenomenon; the *frame of discernment*. Since the phenomenon can only have a single actual value, it follows that all hypotheses of Θ are *mutually exclusive*. A *mass function* m is a probability assignment from the power set of Θ (i.e., 2^{Θ}) to $[0, 1]$ so that:

$$m(\emptyset) = 0 \tag{1}$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \tag{2}$$

where $m(A)$ is the basic probability number for A and it is a measure of the belief committed exactly to this hypothesis. The belief of all possible subsets of A, $Bel(A) : 2^{\Theta} \rightarrow [0, 1]$, is the sum of all masses of all subsets that support A:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{3}$$

Simply put, $Bel(A)$ is a measure of the strength of the evidence in favor of A and it corresponds to a lower bound on the probability that A is true. The upper probability bound is given by the plausibility function, $Pl(A) : 2^{\Theta} \rightarrow [0, 1]$:

$$Pl(A) = \sum_{B \bigcap A \neq \emptyset} m(B) \tag{4}$$

Plausibility is the amount of evidence not contradicting hypothesis A. For $Bel(A)$ and $Pl(A)$, two properties hold:

$$Bel(A) \leq Pl(A)$$
$$Pl(A) = 1 - Bel(\bar{A})$$

Two independent sets of probability mass assignments, m_1, m_2, can be combined (to a joint mass) with Dempster's rule of combination:

$$m_{1,2}(A) = (m_1 \oplus m_2)(A) = \frac{1}{1 - K} \tag{5}$$

where $K = \sum_{B \bigcap C = \emptyset} m_1(B) m_2(C)$ and $m_{1,2}(\emptyset) = 0$

This rule derives the shared belief between the two masses and can be extended to combine multiple masses. Two masses are considered conflicting to the extent they support incompatible hypotheses. An indication of the conflict of the masses is given by K. A quantification can be extracted by the weight of conflict, Con, metric:

$$Con(bel_1, bel_2) = log(\frac{1}{1 - K}) \tag{6}$$

In case of no conflict between bel_1 and bel_2, $Con(bel_1, bel_2) = 0$. If these two beliefs have nothing in common, $Con(bel_1, bel_2) = \infty$.

There are different ways to measure the uncertainty of a mass function. Here, we use the classical entropy measure:

$$LCon(m) = -\sum_{A \in \Theta} m(A) log_2(m(A)) \tag{7}$$

$LCon$ is a measure of the inconsistency within a mass function. Intuitively, the larger the number of mutually disagreeing hypotheses, the larger U becomes.

Assume two devices, c_1 and c_2, participating in a task measuring a phenomenon that takes the values a, b, c and d (either numerical or nominal). As described in Sec. 2, each participating user submits a stream of measurements of the sensed phenomenon. The transformation of these streams to probability masses is as follows: the system generates a probability density function (p.d.f) by constructing a *normalized histogram* [34] from each user-submitted stream. This p.d.f will be used to assign masses to the different hypotheses for the phenomenon. Note that better accuracy requires smaller discretization intervals (i.e., small histogram bin size). For instance, for temperature monitoring tasks we assume 1° intervals. This, however, has an impact on the system performance because the frame of discernment grows (more hypotheses to consider). To compensate for this, we can exclude *improbable* hypotheses; e.g., we do not consider temperatures outside the operational limits of modern smart-phones (e.g., $0 - 32^{\circ}$)[1]. Moreover, histogram bins do not need to have the same granularity. One bin could include temperature measurements $< 0^{\circ}$ (i.e., an unlikely hypothesis during summer months in cities of the northern hemisphere); another temperatures $> 30^{\circ}$. Intermediate bins, corresponding to more probable hypotheses, can have $.5^{\circ}$ granularity (e.g., $0.5, 1, 1.5, ..., 30^{\circ}$).

Assume that the mass of device c_1 is the vector $m_{c_1} = [ab : 0.6, bc : 0.3, a : 0.1, ad : 0.0]$. This states that device c_1 assigns a mass of 0.6 to the hypothesis that the value of the phenomenon is a or b, a mass of 0.3 that the correct hypothesis is either b or c and a mass of 0.1 that a holds. Similarly, the mass of device c_2 is $m_{c_2} = [ab : 0.5, bc : 0.4, b : 0.05, a : 0.05]$. The combination rule (5) yields $m = m_{c_1} \oplus m_{c_2} = [b : 0.46, ab : 0.32, cb : 0.12, a : 0.09]$. Using (4) we get the plausibility that the value of the phenomenon is

1. https://support.apple.com/en-us/HT201678

Algorithm 1 Pseudocode for DBSCAN [35]

```
procedure DBSCAN(D, ε, MinPoints)
    Cluster ← empty
    for each unvisited feature f of D do
        f ← visited
        Neighbors ← QUERYREGION(f, ε)
        if size(Neighbors) < MinPoints then
            f ← outlier
        else
            EXPAND(f, Neighbors, Cluster, ε, MinPoints)

procedure QUERYREGION(f, ε)
    return ε−neighborhood of f

procedure EXPAND(f, Neighbors, Cluster, ε, MinPoints)
    f ← member(Cluster)
    for each n in Neighbors do
        if n visited then
            n ← visited
            Neighbors' ← QUERYREGION(n, ε)
            if size(Neighbors') ≥ MinPoints then
                Neighbors += Neighbors'
        if n ∉ any cluster then
            n ← memberof(Cluster)
```

a, $Pl_m(a) = 0.41$. Finally, with (6) we get that the conflict between m_{c_1} and m_{c_2} is 0.06.[2]

3.2 Bootstrapping Phase

By collecting user reports and transforming them into probability masses, the system enters the bootstrapping phase for each spatial unit. The goal is to give the system an initial understanding of the sensed phenomenon (within each spatial unit) and to remove deviating, or potentially polluting, reports. To do this, the system trains itself to identify structure in user measurements. More specifically, for each report (transformed into a probability mass), it computes *a*) the hypothesis, H_{max}, with the maximum belief, *b*) the belief, $Bel(H_{max})$, of this hypothesis, and *c*) the local conflict of the probability mass. These are included in a 3-dimensional feature vector v_{r_α} (one for each report r_α):

$$v_{r_\alpha} = [H_{max}, Bel(H_{max}), LCon(m_c)]$$

where m_c denotes the probability mass derived from the user report (Sec. 3.1.1). The system waits until a sufficient number of reports has been collected for a spatial unit[3] and then initiates the DBSCAN density-based topological clustering algorithm [35] (Alg. 1). The algorithm input is the set, D, of all feature vectors, the maximum distance, ϵ, that two feature vectors must have to be considered "close", and the $MinPoints$ number. A vector (i.e., point) is considered central if there are $MinPoints$ other vectors close to it. To calculate the distance between two feature vectors v_{r_α} and v_{r_β}, we use the Canberra distance metric [36]:

$$d(v_{r_\alpha}, v_{r_\beta}) = \sum_{i=1}^{3} \frac{|v_{r_\alpha}(i) - v_{r_\beta}(i)|}{|v_{r_\alpha}(i)| + |v_{r_\beta}(i)|}$$

If $d(v_{r_\alpha}, v_{r_\beta}) \leq \epsilon$ we say that v_{r_β} is in the ϵ-neighborhood of v_{r_α}. By setting $MinPoints$ to be low, the algorithm will create many clusters. Nevertheless, since honest reports within a spatial unit follow the same distribution (Sec. 2.1), honest feature vectors will be close to each other. As a result, we can set $MinPoints$ to be rather large (for example, a feature vector can be considered a core point if it has in its neighborhood 45% of all vectors). This expected high density of feature vectors implies that the distance between them will be relatively small. Hence, we can set ϵ to be small (e.g., to be equal to the 30^{th} percentile of all the pairwise vector distances). To estimate the proper values for these parameters, we employ the heuristic described in the original DBSCAN paper [35]. This heuristic leverages the $k - distance$ of the dataset to compute the ϵ and $MinPoints$ of the thinnest cluster created by the algorithm.[4]

2. The numbers used in this example were rounded for easy presentation.

3. This can be identified heuristically depending on the phenomenon; based on our evaluations, 20 reports suffice.

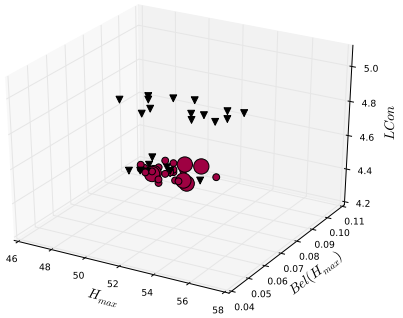


Fig. 2: Clustering with DBSCAN (circles denote inliers).

The output of the algorithm is a partitioning of all feature vectors in D into inliers and outliers. Fig. 2 illustrates such a partitioning: most of the inlying reports suggest $H_{max} = 50$, whereas outlying reports suggest H_{max} to be around 55. Inlying and outlying reports are easily separable when considering the $LCon$ feature. Compared to inlying reports, outlying ones have higher conflict because they support (i.e., they assign masses to) more hypotheses.

DBSCAN examines the topological density of the feature vectors and relies on *honest* majorities to (correctly) identify inliers and outliers. Simply put, it learns what the majority of users suggests for each spatial unit.

As we discuss in Sec. 7, such honest majorities can be marginal; however, it is not certain that they will exist at all spatial units at all time. It might be the case that a spatial unit can be targeted by adversaries in an attempt to locally change the system's perception about the monitored phenomenon. This implies that the data/evidence deemed as inlying is of "malicious" provenance; the majority of users (collectively) exhibit deviant behavior and submit faulty reports from a set of values different than the truthful ones (Sec. 2.2). This will result to a system understanding of the phenomenon (for this spatial unit) that deviates from the correct, adversary-free output. Nonetheless, even in this extreme case, our system manages to limit the impact of such adversarial behavior since the *polluted* spatial unit will be isolated from further affecting the adjacent spatial units (Sec. 3.3).

3.3 Region Merging and Training Phases

In this phase, the system essentially learns the topological variation of the sensed phenomenon. The previous phase labeled user reports (for each spatial unit) as inliers and outliers. Leveraging the inlying reports, the system then merges neighboring spatial units within which user measurements and, thus, in all likelihood, the underlying sensed phenomenon follows (almost) the same distribution.

4. Additional details are presented in [35]

Algorithm 2 Pseudocode for the Region Growing Algorithm

```
procedure REGIONGROWING(Seed s)
    s ← visited
    region ← ∅
    region.add(s)
    active_set ← s
    while active_set not ∅ do
        c = dequeue_point(active_set)
        for n in neighbors_of(c) do
            if n(¬visited) and KSTEST(c, n) : True then
                active_set.add(n)
                region.add(n)
                visited ← n
    return region
```

To compare the similarity between the inlying reports of two neighboring spatial units, we perform a two-sample Kolmogorov-Smirnov (K-S) test. The system first constructs two empirical distributions, from the inlying reports of each spatial unit, and verifies the null hypothesis: i.e., reports are drawn from the same distribution. The K-S statistic is:

$$D_{c_i,c_j} = \sup_x |F_{c_i}(x) - F_{c_j}(x)|$$

where F_{c_i} and F_{c_j} denote the empirical distributions. Based on D_{c_i,c_j}, the system accepts the null hypothesis (and merges the two spatial units) at a significance level of 5%.

Alg. 2 gives the details of the region merging. The algorithm is initiated with a spatial unit as a starting point. It then traverses all neighboring spatial units and examines whether the null hypothesis holds. Its output is a region of units within which the sensed phenomenon follows the same (or almost the same) distribution. Executing the algorithm multiple times, for different starting units (not yet belonging to any region), yields a unique segmentation of the area of interest for the sensing task. If the monitored phenomenon exhibits extreme spatial diversity no (larger) regions will be formed and, thus, each spatial unit will be considered as a separate region. The same holds in the case of adversary-controlled spatial units.

Fig. 3 shows an execution instance of this algorithm for an emulated traffic sensing campaign where participating users contribute their velocities to the RS (Sec. 6). On the left side of the figure, we see a part of the road network of the city of Stockholm. In this context, each road link corresponds to a spatial unit (Sec. 2.1). The right side of Fig. 3 depicts the output of the region merging algorithm, where individual spatial units have been merged into 4 larger regions based on the homogeneity of the reported velocities. The color density of the road links is as an indication of the average speed of each road segment (darker colors indicate smaller velocities).

Once the region merging phase concludes, the system enters the training phase for each formed region. It is identical to the bootstrapping phase except that the clustering algorithm runs over reports originating from all the merged, into a region, spatial units. Again, the output is a labeling of all user reports (of a region) into inliers and outliers.

Our system can operate on data measurements pertaining to any phenomenon. Recall that our system leverages $Hmax, Bel(Hmax), LCon(mc)$ (Sec. 3.3). These features are not specific to any type of distribution. Moreover, this phase employs empirical distributions relevant to *any* type of distribution.

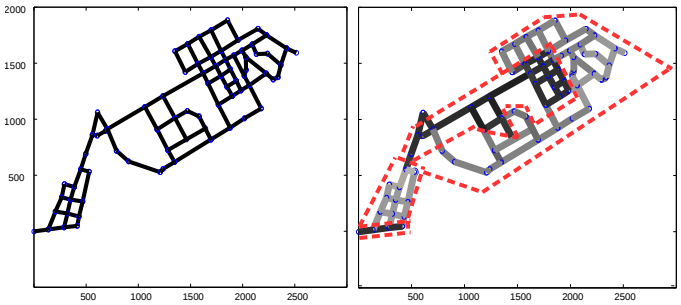


Fig. 3: Illustration of the output of the Region Merging algorithm for a traffic monitoring MCS campaign.

3.4 Classification Phase

Clustering is not efficient for data streams because it requires the execution of the clustering algorithm for each incoming user report. This is why, as described previously, we employ clustering for a small number of initial reports (Sec. 3.2), sufficient to craft an understanding of the sensed phenomenon in each spatial unit. With this knowledge, the system then trains (supervises) an ensemble of classifiers comprising (*i*) a random forest, (*ii*) a naive Bayes (NB) classifier and (*iii*) a nearest neighbor (NN) classifier. The goal of this ensemble is to assess subsequent incoming reports in each region (i.e., their characterization as inliers and outliers).

A random forest [37] is a collection of decision trees, each trained over a different bootstrap sample. Each decision tree is a classification model created during the exploration of the training data. More specifically, the interior nodes of the tree correspond to feature values of the input data. Recall that our system considers three features: $H_{max}, Bel(H_{max})$ and $LCon(m_c)$. For instance, according to the example of Fig. 2, an interior node could be $H_{max} > 52$. Nodes can have other nodes as children, thus, creating decision paths (e.g., $H_{max} > 52 \; and \; Bel(H_{max}) < 0.1$). Tree leaves describe the decision (i.e., classification) of all training data described by the path from the root to the leaf. For example, training data for which $H_{max} > 52$ and $Bel(H_{max}) < 0.1$ are *inlying*. Once a new report arrives, each decision tree estimates its class by examining all the possible paths the report could follow. The estimations of all decision trees are combined into the decision of the random forest.[5]

The classification rule for the NB classifier is:

$$\hat{y} = \underset{y}{argmax} P(y) \prod_{i=1}^{3} P(x_i|y)$$

where $\hat{y}$ is the decision (i.e., inlying or outlying) and x_i is a feature of the feature vector (i.e., $H_{max}, Bel(H_{max})$, $LCon(m_c)$). Prior probabilities are estimated by examining the fraction of inlying/outlying reports for the training set. The NN classifier implements a simple k-nearest neighbors voting scheme (our system operates with $k = 10$). All classifiers are trained with the output of the previous phase: labeling of inlying and outlying reports for each region.

Classifiers make their decisions (on inlying or outlying reports) which are, in turn, combined in a majority voting scheme to form the system's final decision. If deemed inlying, then a report is taken into consideration for *updating* the system's perception of the phenomenon (in a region). More precisely, the system generates a mass function that is the

5. Due to space limitations, we omit the details of this process and we refer the interested readers to the relevant work [37].

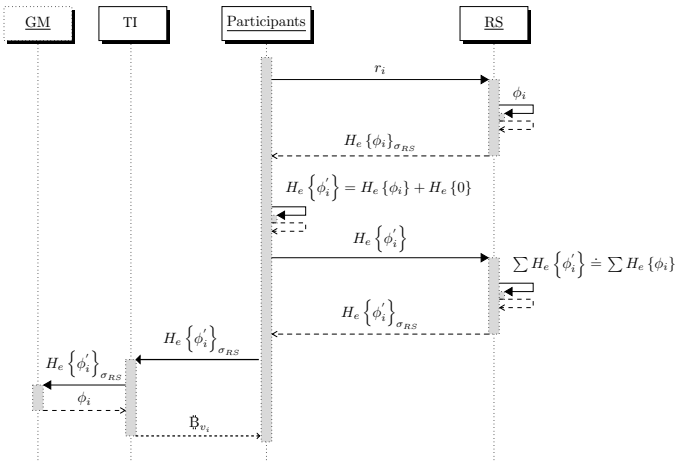


Fig. 4: Remuneration Protocol

combination (according to Dempster's rule of combination) of all probability masses of all inlying reports in a region. Finally, for each assessed report r_i the classification ensemble computes a trustworthiness score sc_{r_i}: the probability that the report is trustworthy.

3.5 Concept Drift Detection Module

This module is responsible for detecting changes in the statistical properties of the sensed phenomenon. It continuously monitors the disagreement between the probability mass, the system has for each region and each new incoming report. This is done by computing the Con metric (Eq. 6). More specifically, if the conflict between the already computed probability mass and the incoming user reports exceeds a predefined threshold, then an alert is triggered. Based on the type of concept drift, different actions can be taken by the system. We consider the following cases:

Local Concept Drifts: They concern only one, or a small number of regions. In this case, the system retrains (i.e., it enters the Training Phase) the classifiers of these regions (in question) with the new data it received from the users.

Global Concept Drifts: They affect the whole area of interest for a task. As a result, the system has to be bootstrapped; the area of interest is broken down to the original spatial units and the training and region merging phases are executed. The system differentiates between these cases of concept drift by leveraging the location information in the user reports.

In general, there is no particular threshold (i.e., weight of conflict) for the concept drift, but rather an entire range of threshold values depending on how sensitive the model is to changes of the underlying phenomenon. Lower thresholds imply faster detection but, at the same time, may result to (possibly unnecessary) retraining of the system. This is especially so for *incremental* concept drifts slowly evolving [38] over time. On the other hand, higher thresholds imply slower adaptation to changes. Identifying the proper thresholds strongly depends on the underlying phenomena and is orthogonal to this investigation.

3.6 Querying the System

Any authenticated entity can query the system sending:

$$q = \{H_{max} \,||\, loc \,||\, \sigma_{PrvKey} \,||\, Ps\}$$

Again, σ_{PrvKey} is a digital signature (of the querier) produced with some private key. The corresponding public key is certified by the credential management system and included in the certificate Ps. When queried, the system identifies the region relevant to the query based on loc. Then, it responds with the region's $[H_{max}, Bel(H_{max}), Pl(H_{max}), LCon(m_s)]$. m_s denotes the mass the system assigns to the different hypotheses derived from the combination of all inlying reports for that region; from the moment of the latest concept drift till the issued query.

Users can issue fine-grained queries as follows:

$$q = \{Bel/Pl \,||\, H \,||\, loc \,||\, \sigma_{PrvKey} \,||\, Ps\}$$

will return the belief and the plausibility that the system assigns to hypothesis H (for the region containing loc). Queriers measuring the same phenomenon (as the task) can assess the degree of agreement (or contradiction) between their measurements and the system's belief, by issuing:

$$q = \{Con \,||\, [v_1, v_2, v_3, ..., v_n] \,||\, loc \,||\, \sigma_{PrvKey} \,||\, Ps\}$$

The RS will respond with the Con (Eq. (6)) value between its mass function and the mass function derived by the measurements that the user provided (i.e., $[v_1, v_2, v_3, ..., v_n]$).

Users can also receive the whole mass function of the system for a region with the query:

$$q = \{m \,||\, loc \,||\, \sigma_{PrvKey} \,||\, Ps\}$$

Queriers can then perform a *pignistic transformation* [39] of m; a transformation of the system's belief function to a probability function that can yield optimal decisions [40].

4 User Remuneration Protocol

Upon completion of the sensing campaign the RS must decide on the reward that user-contributed reports should receive. Although our protocol is not restricted to any specific remuneration method (e.g., micro-payments, service quotas) in what follows we consider *Bitcoin payments*: a rewarding mechanism that can offer immediate value (i.e., monetary) to contributing users [41].

An overview of the protocol is presented in Fig. 4. For each report, r_i, the RS computes a value ϕ_i; the importance of the report (Sec. 4.1). This value is encrypted homomorphically under the public key P (Sec. 2). The homomorphically encrypted ϕ_i values are signed by the RS and encrypted with the corresponding public key of the pseudonym Ps (Sec. 2) included in the report. These encrypted values are then published on-line on a Public Bulletin Board (PBB) instantiated as a web-server. Participating users retrieve from the PBB an Extensible Markup Language (XML) document containing all the signed and encrypted ϕ_i and open the ones belonging to them; i.e., the ones for which they possess the corresponding private keys. Subsequently, the mobile clients re-randomize $H_e\{\phi_i\}$ by adding a unique homomorphic encryption of 0; the probabilistic nature of the employed homomorphic encryption scheme [26] allows the GM to provide each client with a unique homomorphic encryption of 0. More specifically, for each $H_e\{\phi_i\}$ the mobile clients compute $H_e\left\{\phi_i'\right\} = H_e\{\phi_i\} + H_e\{0\}$ and submit it back to the RS. This randomization step protects user privacy against *honest-but-curious* infrastructure entities as discussed in Sec. 5.3.

Once all clients have submitted the randomized $H_e\left\{\phi_i'\right\}$, the RS and the GM examine possible client misbehavior; i.e.,

if they have randomized $H_e\{\phi_i\}$ by adding values other than $H_e\{0\}$. If such a misbehavior is detected, then a resolution protocol is initiated (Sec. 5.1.1). In any other case, the RS signs and publishes all $H_e\left\{\phi_i^{'}\right\}$ on the PBB and informs the participating users to proceed and claim their rewards from the TI. Clients retrieve the signed, randomized, and homomorphically encrypted values and present the ones belonging to them (along with their bitcoin address) to the TI. The TI then computes the homomorphic sum of all the $H_e\left\{\phi_i^{'}\right\}$ which will be decrypted by the GM. At this point, dishonest clients might present some $H_e\left\{\phi_i^{'}\right\}$ (not belonging to them) as their own. Our system is capable of thwarting such misbehavior (Sec. 5.1.2). Finally, in case no misbehavior is detected, the TI credits each user address with the corresponding bitcoin amount.

Communications between system entities are performed over secure TLS channels. Each system entity possesses digital certificates which are installed on the mobile clients of participating users. A discussion on the performance and efficiency of the protocol is presented in Sec. 7.

4.1 Evaluating User Contributions

To assess the value (i.e., importance) of each user report the RS works as follows: let $\Pi = [r_1, r_2, ..., r_N]$ be the set of all N reports the RS has received during the task. We define as $S(\Pi, r_i)$ the set of reports, in Π, which arrived no later than the report r_i. Each user contributed report, r_i, corresponds to vector $[reg_{r_i}, sc_{r_i}]$ where reg_{r_i} is the region (Sec. 3.3) that the location (loc) of the report specifies. The value sc_{r_i} is the report trustworthiness score (Sec. 3.4). For each set $S(\Pi, r_i)$ the RS constructs the vector $R_{S(\Pi,r_i)} = [R^1_{S(\Pi,r_i)}, R^2_{S(\Pi,r_i)}, ..., R^K_{S(\Pi,r_i)}]$ where K is the number of regions. For instance, the sum of scores of all inlying reports for region j can be computed as $R^j_{S(\Pi,r_i)} = \sum_{S_j} sc_{r_i}$ where $S_j : \{r_i | r_i \in S(\Pi, r_i)$ and, $reg_{r_i} = j$ and, $sc_{r_i} \geq 0.5\}$. The value of the report r_i for a region x is computed as follows:

$$\phi(r_i) = \begin{cases} 0 \text{ if } sc_{r_i} < 0.5 \\ log(R^x_{S(\pi,r_i)}) - log(R^x_{S(\pi,r_i-1)}) \quad \text{otherwise} \end{cases} \tag{8}$$

Intuitively, this remuneration mechanism favors reports that are trustworthy and were submitted for regions for which the system has not received many user contributions.

5 Security & Privacy Analysis

In this section, we assess the security and privacy protection offered by the user remuneration protocol. We begin with a discussion on how our system can detect and deter *misbehaving* users (possessing valid cryptographic credentials) and proceed with a formal analysis of the protocol in the presence of *external*, Dolev-Yao [42] adversaries. Finally, we analyze how user-privacy is affected in the presence of *honest-but-curious* system entities.

5.1 Detecting and Deterring Misbehavior

5.1.1 Type I Misbehavior - Increasing $H_e\{\phi_i\}$

Malicious users might try to increase the $H_e\{\phi_i\}$ values they receive from the RS: instead of randomizing it -adding a homomorphic value of 0 (Sec. 4)- they might attempt to increase it by adding a positive value. This way, they manage to increase the score their report received. The RS, in coordination with the GM, can detect such misbehavior: More specifically, although the RS cannot map any $H_e\{\phi_i\}$ to its randomized value $H_e\left\{\phi_i^{'}\right\}$, it can collaborate with the GM to detect whether the sum of all $H_e\{\phi_i\}$ values is equal to the sum of the $H_e\left\{\phi_i^{'}\right\}$. Towards that, the RS computes $c = \sum H_e\left\{\phi_i^{'}\right\} - \sum H_e\{\phi_i\}$ and sends it to the GM for decryption (recall that the GM is the only entity possessing the private key of the homomorphic encryption scheme). If $c \neq 0$, then a misbehavior has occurred. In this case, the RS sends to the GM the set of all randomized $H_e\left\{\phi_i^{'}\right\}$ values it received from the clients. Subsequently, the GM initiates a Publish/Subscribe channel in which all participating clients are registered as subscribers. The GM publishes to this channel one $H_e\left\{\phi_i^{'}\right\}$ at a time. Upon reception, the client that produced this randomized value must reveal to the GM the corresponding, signed by the RS and, thus, not forgeable, $H_e\{\phi_i\}$. The GM checks if $H_e\left\{\phi_i^{'}\right\} = H_e\{\phi_i\}$. If this is not the case, a pseudonym revocation protocol [19] is initiated for the pseudonym associated with $H_e\{\phi_i\}$. The process continues until all invalid $H_e\left\{\phi_i^{'}\right\}$ are detected and the pseudonyms corresponding to malicious users are revoked.

Algorithm 3 Modeling Homomorphic Encryption in ProVerif

```
1. type h_pkey.
2. type h_skey.
3. fun hpk(h_skey): h_pkey.
4. fun h_encrypt(bitstring, h_pkey): bitstring.
5. fun h_randomize(bitstring, bitstring): bitstring.
6. reduc forall
   x: bitstring,
   r: bitstring,
   y: h_skey;
   h_decrypt(h_randomize(
    h_encrypt(x,hpk(y)),
    h_encrypt(r,hpk(y)), y
    )
  ) = x.
```

5.1.2 Type II Misbehavior - Claiming/Decreasing the contributions of other participants

A second type of possible misbehavior is when malicious clients try to claim rewards for $H_e\left\{\phi_i^{'}\right\}$ values not belonging to them. This type of misbehavior is easily detectable: a conflict will occur when the legitimate owner also claims the $H_e\left\{\phi_i^{'}\right\}$. Resolving such a conflict is straight-forward; the GM will again request the corresponding, signed by the RS, $H_e\{\phi_i\}$ value. Then it will initiate a Proof-of-Possession protocol for the private key corresponding to the pseudonym Ps that authenticated the report r_i (which the value $H_e\{\phi_i\}$ was assigned to). A pseudonym revocation protocol [19] can again ensure malicious client(s) are evicted.

Finally, malicious users cannot manipulate the homomorphic encrypted ϕ values of other users: these are encrypted with the public key corresponding to the pseudonym Ps (Sec. 2) attached to each report.

5.2 Secrecy Analysis for Dolev-Yao adversaries

In this section we formalize the security of the remuneration protocol by using ProVerif: an automated protocol

Datum	Entity	Secrecy	Strong Secrecy/ Unlinakbility
Dev. id (id)	GM	✓	✓
Subm. report.	RS	✓	✓
Device pseud.	RS	✓	✓
$H_e\{\phi_i\}$	RS	✓	✓
$H_e\left\{\phi_i^{'}\right\}$	RS, TI	✓	✓

TABLE 1: Secrecy Analysis for Dolev-Yao Adversaries

verifier [43] that enables the modeling of the protocol in π-Calculus [43]. This allows for an increased confidence in the analysis results.

In ProVerif, entities (infrastructure components and users) are described as processes. Protocols are modeled as a parallel composition of multiple copies of these processes. ProVerif assumes sets of *names* and *variables* along with a finite *signature*, Σ, comprising all the function symbols accompanied by their *arity*. The basic cryptographic primitives are modeled as symbolic operations over bit-strings representing messages encoded with the use of *constructors* and *destructors*. Constructors generate messages whereas destructors retrieve parts of the messages they operate on.

Alg. 3 illustrates the modeling of homomorphic encryption in ProVerif. More specifically, we first define two key types corresponding to the public and private keys of the employed homomorphic encryption (lines, 1 and 2); the mapping between these two keys is done in line 3. In line 4 we define the homomorphic encryption function: given a bitstring and a public key, it produces a homomorphically encrypted bitstring. The randomization function is modeled as follows: given any two homomorphically encrypted bitstrings we produce a third one. Finally, line 6 models the destructor, i.e., the decryption function: for a bitstring, output of the $randomize$ function, and the corresponding private key it produces the original bitstring.

Adversaries in ProVerif follow the Dolev-Yao model [42]: they can eavesdrop, modify and forge messages according to the cryptographic keys they possess. To protect communications, every emulated MCS entity in the analysis maintains its own private keys/credentials. In ProVerif, the attacker's knowledge on a piece of information i, is queried with the use of the predicate $attacker(i)$. This initiates a resolution algorithm whose input is a set of Horn clauses that describe the protocol. If i can be obtained by the attacker, the algorithm outputs $true$ (along with a counter-example) or $false$ otherwise. ProVerif can verify *strong-secrecy* properties implying the adversary cannot infer changes over secret values. To examine strong-secrecy for datum i, the predicate *noninterf* is used.

Table 1 summarizes our findings: our system ensures the secrecy of all the critical pieces of information, thus, guaranteeing the secrecy of sensitive information in the presence of external, Dolev-Yao adversaries.

5.3 Honest-but-Curious System Entities

We begin with a discussion on the knowledge of each system entity and proceed with an analysis of the privacy implications resulting from different combinations of *information-sharing* honest-but-curious system entities.

5.3.1 Knowledge of System Entities

The RS receives user reports and assigns them a score (Sec. 3.4). Moreover, each user contributed report contains location information also known to the RS. Nonetheless, the unlinkability achieved due to the use of pseudonyms prevents the RS from knowing which reports were contributed by the same user and, thus, from reconstructing the user whereabouts. Of course, inference attacks leveraging filtering techniques [19] such as Kalman Filters [44], to link reports based on mobility predictions, are still feasible; such attacks are beyond the scope of this work.

The GM decrypts the homomorphic sum of all the randomized ϕ; i.e., it learns $\sum\{\phi_i\}$ for each user. In case of a detected misbehavior, the GM also links (some) $H_e\{\phi_i\}$ to the corresponding randomized $H_e\left\{\phi_i^{'}\right\}$. The GM could also misbehave and abuse the misbehavior detection protocol (Sec.5.1.1) to reconstruct the whereabouts of users. Nonetheless, such a case is beyond the adversarial model considered in this work (Sec. 2.2): MCS system entities, besides users/clients, are honest-but-curious but not malicious.

The TI receives all the $H_e\left\{\phi_i^{'}\right\}$ values belonging to the same user: it knows how many values each user submits. Nonetheless, these pieces of information contain no location information. Moreover, since this entity does not have the private key of the homomorphic encryption scheme it cannot decrypt them. The TI also learns from the GM the $\sum\{\phi_i\}$.

5.3.2 Colluding RS - TI

None of these two entities can decrypt the homomorphically encrypted ϕ values. Nevertheless, given their knowledge of all ϕ_i values the RS assigned and the number (and sum) of the ϕ values users submitted to the TI, these two honest-but-curious entities can try to infer which reports belong to each user in an attempt to reconstruct all users' whereabouts. More specifically, they can try to solve the following problem:

SUM_TO_VALUES Problem: Let $\Phi = \{\phi_1, \phi_2, ..., \phi_N\}$ be the values the RS has assigned to all N user reports. Let $S = \{s_1, s_2, ..., s_K\}$ be the set of all decrypted, by the GM, sums of the homomorphically encrypted ϕ values the TI has received from all K users. The TI also knows the number of ϕ values (i.e., the summands) for each s_i: the set $L = \{l_1, l_2, ..., l_K\}$. Based on the sets $\Phi, S,$ and L, the two colluding entities can try to infer the set $U = \{u_1, u_2, ..., u_K\}$, where u_i is the set of ϕ values corresponding to user i. Simply put, the set U is a *partition* of Φ: all elements in U are disjoint (a ϕ value is assigned to only one user report), thus, it holds that $\bigcup_{u\in U} = \Phi$.

Theorem 1. *The SUM_TO_VALUES Problem is in NP-Hard.*

Proof. We will reduce the SUM_TO_VALUES problem to the MAXIMUM_WEIGHTED_CLIQUE.

We construct a multipartite, edge-weighted graph $G = (V, E)$ with K independent sets: each independent set i corresponds to all possible, p_i, allocations of l_i values of the set Φ summing to s_i. For instance, an allocation, i.e., a vertex v, could indicate that the values $\{\phi_x, \phi_y\}$ belong to user u_1. Each vertex v has a weight which is the probability that the allocation described by the vertex is correct. For example, let us assume that user u_1 contributed two reports to the RS. Moreover, let the following be possible allocations of two ϕ values for u_1 summing to s_1: $u_{1,1} = \{\phi_x, \phi_y\}$ and $u_{1,2} = \left\{\phi_x^{'}, \phi_y^{'}\right\}$ and let $r_x, r_y, r_{x'}$ and $r_{y'}$ be the corresponding reports that these values were assigned to.

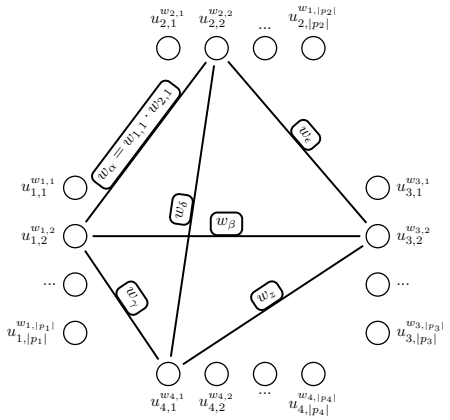


Fig. 5: SUM_TO_VALUES Graph with a 4-Clique

One way of deriving vertex weights is by leveraging the euclidean distance of the respective reports. For instance, if the euclidean distance of the locations of r_x, r_y is less than the one of $r_{x'}, r_{y'}$ and the time difference of these report pairs is approximately the same, then the colluding RS and TI can conclude that $P(u_{1,1} = correct) > P(u_{1,2} = correct)$; i.e., $w_{1,1} > w_{1,2}$. Vertex weights essentially capture the additional knowledge, or inference, the two colluding entities can perform over the collected data. Such inferences can be either simplistic (as the one described above) or the result of more sophisticated filtering techniques: Kalman Filters [44] or multi-hypothesis testing [45].

In G, two vertices v_x, v_y are connected by an edge $e_{x,y}$ if the allocations they describe are disjoint; i.e., they do not allocate the same ϕ value to two different users. For example, no edge connects a vertex $v_1 = \{\phi_x, \phi_y\}$ with a vertex $v_2 = \{\phi_x, \phi_z\}$ since ϕ_x is a common element. Furthermore, no edge connects two vertices containing values assigned to reports signed by the same pseudonym: each pseudonym is bound to a single user. Fig. 5 illustrates such a graph. The weight w of an edge captures the *overall* probability that the allocations described by the two connected vertices are true. This is computed as the product of the respective weights of the two connected vertices.

Obviously, all K-cliques of G are maximum and correspond to partitions of Φ (recall that the graph is K-partite). Based on G, to find the most probable (i.e., the correct U) partition of Φ, the colluding RS and TI will have to find the MAXIMUM_WEIGHTED_CLIQUE of G. This optimization problem is known to be NP-Hard [46]. □

Overall, when the honest-but-curious RS and TI entities collude, they have to devise some effective (and probably costly) method to compute the vertex weights. Assuming that this is the case, they still have to solve a computationally hard problem with a complexity that increases exponentially as more users join the sensing task.

5.3.3 Colluding RS - GM

This collusion does not have the strong repercussions as the previous case of colluding RS and TI entities. The RS and GM do not know how many ϕ values each user submitted: recall, the TI sums all the homomorphically randomized and signed ϕ values and sends the sum to the GM. As a result, in order to build a SUM_TO_VALUES graph they will have to enumerate, for each user i, all possible combinations of ϕ values summing to s_i. This problem is the counting version of the SUBSET-SUM problem, which belongs to the #P complexity class: it is *at least* as hard as the NP-Complete SUBSET-SUM problem [47].

5.3.4 Colluding GM - TI

In case the GM and TI entities collude, they can infer no additional information about the users besides the one they already possess. More specifically, they neither know the set Φ (this is known only to the RS) nor the corresponding locations of the user reports contributed to the RS.

5.3.5 Colluding RS - GM - TI

In the extreme case where all three system entities collude, then they can completely breach user privacy; i.e., infer all the reports belonging to the same user and reconstruct their whereabouts.

6 Experimental Setup

Our data verification comprises machine learning mechanisms and heuristics and, thus, getting theoretical bounds on its quality is intractable. Instead, we provide strong empirical evidence on the performance of the system leveraging both real-world and synthetic datasets.

Datasets - The real-world dataset relates to MCS-based environmental monitoring applications, whereas the synthetic dataset relates directly to a traffic monitoring sensing campaign. In both cases, we inject faulty reports originating from adversaries, as detailed below, and we evaluate the system's ability to accurately yield a truthful and undistorted view of the underlying phenomenon.

We use the Strata Clara (SC) dataset, from the *Data Sensing Lab* [48], as a reference point in the domain of environmental monitoring applications [4, 49]. It contains raw measurements of different physical phenomena (i.e., *humidity*, *sound* and *temperature*), from 40 sensors deployed at the Strata Clara convention center in 2013. The underlying (normal) distributions of the monitored phenomena are: $(\mu, \sigma) = \{(31, 5)|(3, 2)|(21, 1.3)\}$, respectively.

The *synthetic dataset* is an emulation of a traffic monitoring MCS task [5]: drivers' smart-phones report their location and velocity to the RS. We consider 250 users and simulate urban road links (and traffic conditions) by generating "actual" location traces for each vehicle/mobile with the SUMO [50] traffic simulator. To produce "realistic" measurements, a percentage of the location updates was degraded by introducing a random error, for example, due to weak GPS signal.

Adversarial Behavior - To emulate adversaries we inject faulty reports drawn from distributions different than those of the adversary-free datasets (i.e., the ones corresponding to the underlying phenomena and containing reports only from honest users (devices)). We instantiate coordinated adversaries by having them inject data in the same manner. We consider the following three cases:

- *"Uniform" Adversaries* report values drawn from a uniform distribution (i.e., they assign an equal mass to all hypotheses).
- *"Normal" Adversaries* report values drawn from a normal distribution.
- *"N-value" Adversaries* select N hypotheses (Sec. 3.1) and randomly distribute probability masses to them.

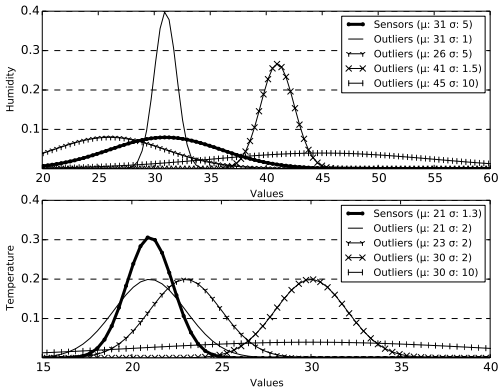
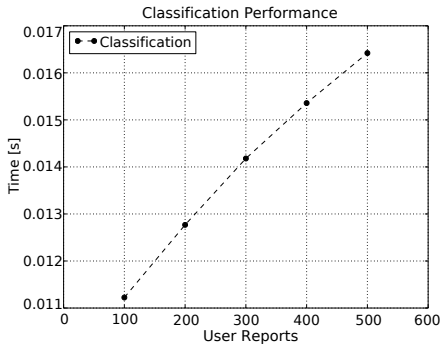


Fig. 6: Adversarial data distributions and their similarity with the original SC humidity and temperature sensor datasets (depicted with thick lines).



(a)

Fig. 7: Performance analysis of the classification ensemble.

The adversarial strategy determines the distortion adversaries try to impose on the system. Among the above, "normal" adversaries may cause significant distortion (by increasing the distance between the μ of their distribution and the mean of the distribution that honest samples follow). On the other hand, adversaries may try to increase the uncertainty of the system, choosing a normal distribution with large σ or a uniform distribution (thus, causing maximum uncertainty). They could also try to increase the system's certainty about the true value of the phenomenon (e.g., by selecting a normal distribution with μ equal to the mean of the honest distribution but with significantly smaller σ). The system should react even in this case; its output must reflect the innate uncertainty of the sensed phenomena.

For "normal" adversaries the employed (μ, σ) determine the similarity (i.e., the overlap) between the honest and adversarial distributions. This is as an indication of the detection difficulty: larger overlaps render the detection of malicious contributions hard. Nevertheless, even for highly similar distributions, our system manages to correctly identify both the honest and malicious samples (Sec. 5).

Fig. 6 presents examples of adversarial strategies (Sec. 6) for the humidity and temperature measurements datasets. Recall that the underlying (normal) distributions for these phenomena are: $(\mu, \sigma) = \{(31, 5)|(21, 1.3)\}$, respectively. For humidity (upper part of Fig. 6), adversaries using the $(\mu = 41, \sigma = 1.5)$ distribution aim for a large deviation from the actual value of the phenomenon. Adversaries using the $(\mu = 31, \sigma = 1)$ try to increase the system's certainty with respect to the actual value of the phenomenon.

We do not consider malicious users acting independently with different strategies: in that case, their effect on the system would be significantly smaller compared to collaborative attackers; *the more reports (users) support the same hypotheses, the more probable it is for the system to believe them.*

To assess the impact of pollution attacks, we examine two cases: *local attacks*, targeting specific regions, and *global attacks* that aim to distort the system output for as many (if not all) regions as possible. For local attacks, we examine the trade-off between the detectability of adversarial reports (depending on the distribution chosen by the adversaries) and the distortion they inflict on the system's output. Adversarial report distributions that significantly differ from the actual one (i.e., based on honest users' reports) can more effectively distort the system's output. But, at the same time, such reports can be detected and sifted easier.

For global attacks, we assume adversaries do not simply decide on a common distribution, but they also jointly agree on the optimal allocation of their faulty reports across the different regions, in order to maximally affect the system [51, 52]. Simply put, they try to gain the majority in as many regions as possible. This is possibly irrespective of the physical placement of the adversaries: they can forge the location of their reports. To model this enhanced adversarial coordination, we assign a popularity value, p_i, to each region: the number of user queries expected to be issued for region i. We also assign c_i, the number of honest users, expected to be within the region i, and we assume that c_i is proportional to p_i; a popular region (e.g., roads around the city center) is expected to have more users. The problem of optimally allocating adversarial reports to each region, based on the knowledge of p_i and c_i, is formulated as:

$$\text{Maximise:} \quad \sum_{i=1}^{N} x \cdot p_i$$

$$\text{subject to:} \quad \sum_{i=1}^{N} x \cdot c_i \leq M,$$

$$x \in \{0, 1\}$$

M is the number of malicious reports (or, equivalently, users, as we assume a sybli-proof security scheme and that the RS accepts reports at the same rate from all devices, adversarial or not) and N is the number of regions. While RS can estimate v_i and c_i relatively accurately, based on the large volume of data it has, this is harder for the adversaries. Exact knowledge of v_i and c_i is unrealistic, thus, we assume that adversaries have inaccurate estimates of those values. We assume that they solve this optimization problem centrally. This way, we emulate their ability to coordinate and decide on the allocation of faulty reports to different regions.

7 Results and Analysis

First, we evaluate the efficiency of our classification ensemble since MCS applications can generate massive amounts of data. We, then, analyze its accuracy in the presence of adversaries. Our focus is on the system's ability to identify and filter out faulty reports; i.e., the labeling of user reports as *inlying* and *outlying*. We use five performance metrics [53]: (*i*) *precision*, (*ii*) *recall*, (*iii*) *F-score*, (*iv*) *Matthew's correlation*

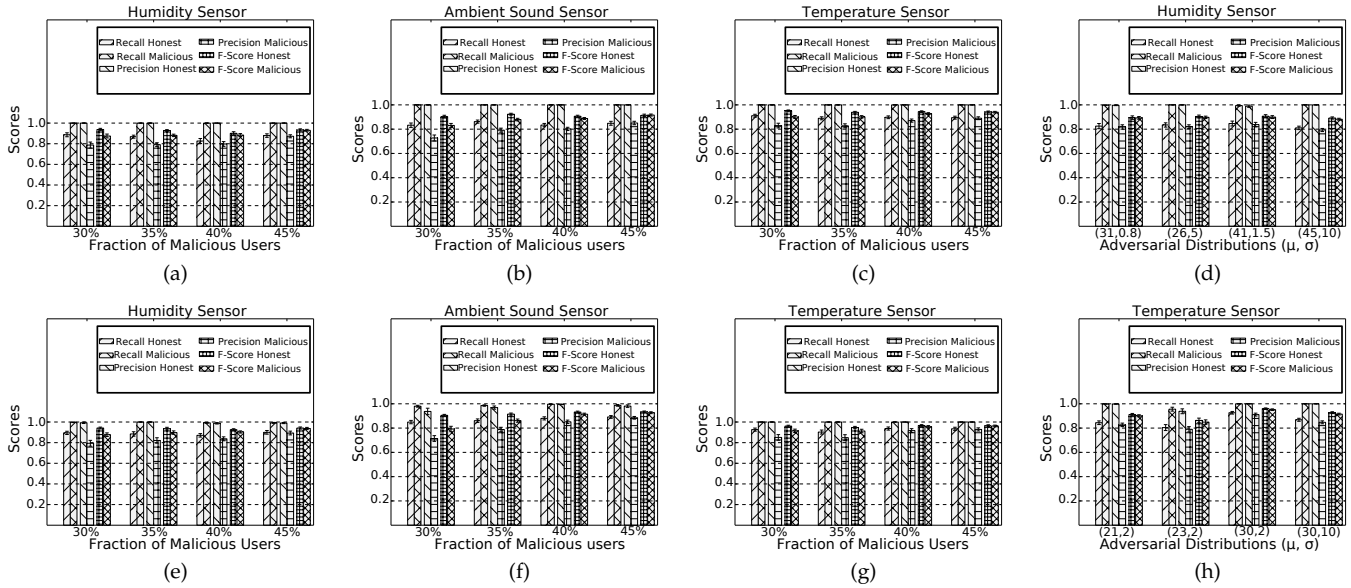


Fig. 8: Precision, Recall and F-score metrics, for *uniform* (a, b, and c), *N-value* (e, f, and g) and *normal* (d, h) adversarial strategies. *SC dataset*.

coefficient (MCC) and (*v*) *Jaccard similarity score.*; they measure a system's classification accuracy and of its ability to correctly identify instances inliers and outliers. Next, we continue with an evaluation of the adversarial impact on the system responses to user queries and we compare our scheme with different robust aggregation functions. For an analysis of the system's adaptiveness to concept drifts, we refer readers to [24]. We conclude this section with a performance evaluation of the user remuneration protocol.

In each experiment, we provide the system with reports originating from both honest and malicious users (Sec. 6). This dataset is partitioned into two sub-sets: a *training* set (TS) and an *evaluation* set (ES). Based on the TS, the bootstrapping (Sec. 3.2) and training (Sec. 3.3) phases take place. Then, the ES is used to assess the performance of the supervised classification part (Sec. 3.4). Due to space limitations, we refrain from evaluating the accuracy of DBSCAN as this is reflected on the performance of the unsupervised classification: better training yields better classification results. For each simulation we perform ten-fold cross validation to avoid *overfitting*. We show results based on both datasets (Sec. 6); due to space limitations, we do not repeat similar figures from both datasets.

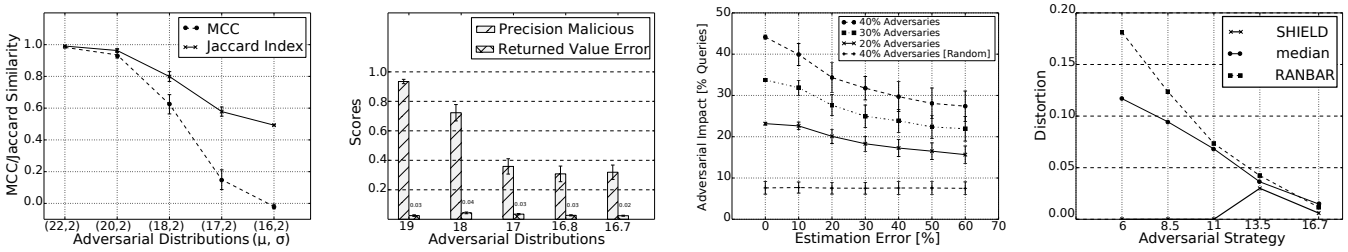
Classification Complexity Analysis and Efficiency. The complexity of the DBSCAN algorithm is $\mathcal{O}(r \cdot logr)$, where r is the number of reports within a spacial unit. Furthermore, the complexity of the region merging algorithm, for an area of interest with n spatial units, is $\mathcal{O}(n^2)$. Each KS test is executed in approximately $0.0005\ sec$. The low complexity of both algorithms serves as an indication of the efficiency of the training phase. This is important for highly dynamic and regularly changing phenomena (entailing many concept drifts) that require retraining of the system.

MCS campaigns might result in large amounts of user contributions. These must be examined by the ensemble of classifiers (Sec. 3.4) in an efficient manner. Fig. 7 (a) shows the performance of the ensemble as a function of the number of user reports per sec: the classification time increases (linearly) with the number of user reports. The classification of 200 user reports requires less than $0.013\ sec$ whereas 500 concurrent reports are classified in less than $0.017\ sec$. For the performance analysis the RS is deployed on a commodity server with an 8-Core, 3.6 GHz CPU.

Classification Accuracy. Fig. 8 shows the precision, recall and F-score metrics for the SC dataset for different adversarial strategies (i.e., uniform, N-value and normal distributions). Bars depict the scores both for honest and malicious reports. The overall correctness of the system remains high, regardless of the number of malicious users (Average F-score ≥ 0.85); in almost all cases, the percentage of correctly classified samples is at least 80%. However, higher accuracy is achieved for larger number of adversaries ($\geq 35\%$). This happens because the number of negative samples increases in the evaluation set and, thus, mis-classifying one such sample has a smaller impact on the overall precision.

Fig. 8 (d), (h) examine the system's accuracy for adversaries following a normal distribution (for the humidity and temperature sensors). We fix the number of malicious users to be 45% and we vary the percentage of overlapping regions (with the honest distribution) by changing the (μ, σ) values of the adversarial distribution. We see that our framework manages to correctly classify both positive and negative samples with high accuracy even for rather similar distributions. For instance, even when the humidity dataset is injected with malicious reports drawn from a normal distribution with high overlap ($\mu = 26, \sigma = 5$ i.e., 70% overlap, Fig. 6), accuracy remains high (F-score ≥ 0.78).

We further examine the impact of overlapping regions (Fig. 9 (a)) for the synthetic datasets generated from the emulated traffic monitoring sensing task (Sec. 6). Honest users report their velocities drawn from a normal distribution with $(\mu = 16, \sigma = 2)$. For each simulation



(a) (b) (c) (d)

Fig. 9: (a) MCC and Jaccard coefficients as a function of the percentage of overlapping regions. (b) Analysis of Local and (c) Global Adversarial Impacts. (d) Comparison of SHIELD with robust aggregation functions. *Transportation dataset.*

run, we used different normal (adversarial) distributions; $\{(\mu = 22, \sigma = 2)|(\mu = 20, \sigma = 2)|(\mu = 18, \sigma = 2)|(\mu = 17, \sigma = 2)|(\mu = 16, \sigma = 2)\}$. Each case corresponds to a different percentage of overlapping regions; 35%, 55%, 80%, 90% and 100% respectively. Again, we set the number of malicious users to be 45%.

We see that the system achieves perfect prediction when the overlap between the distributions is relatively small (i.e., their similarity is less than 35%). For a 50% overlap, the classification accuracy still remains high (MCC, Jaccard) ≥ 0.85. Even for high overlap percentages ($\geq 80\%$), accuracy is close to 60%. Considering that the two distributions are almost identical, adversaries cannot significantly distort the system's output. Finally, for identical distributions, the system exhibits an average random behavior (MCC = 0, Jaccard = 0.5) since malicious and honest reports do not differ at all (same as classifying based on a "coin toss").

In conclusion, both honest and malicious reports are shown to be assessed correctly, irrespectively of the employed adversarial strategy. Honest data following distributions with small standard deviations can be classified better with almost perfect scores (e.g., temperature SC dataset with $(\mu = 21, \sigma = 1.3)$). However, even for high standard deviation values - a measure of the phenomenon's uncertainty - the impact on classification accuracy is negligible ($\leq 10\%$).

Impact of Adversaries. For local attacks, within a region, we measure the detectability of adversaries data through the precision metric. We employ the traffic monitoring dataset and set the fraction of adversaries to 45% of the total number of users (reports) in the region. We assess their impact by examining the distortion they cause; i.e., the difference between the value the system would report for a region in an adversary-free state and the value the system reports in the presence of adversaries. As Fig. 9 (a) shows, adversaries with report values drawn from the $(\mu = 19, \sigma = 2)$ normal distribution are easily detected and, thus, the distortion they inflict is negligible (0.03). As the adversarial distribution moves towards the actual one, precision drops; but so does the distortion of the system output (for the targeted region).

For global attacks, we assume overwhelming adversaries; they comprise almost the majority of system users and coordinate their attacks by deciding an optimal allocation of their reports (by solving the optimization problem presented in Sec. 6). Note that this optimal allocation is computed against a non-optimal allocation of honest users (honest users report data from their current location and thus, they do not coordinate). Figure 9 (b) shows the utility that adversaries

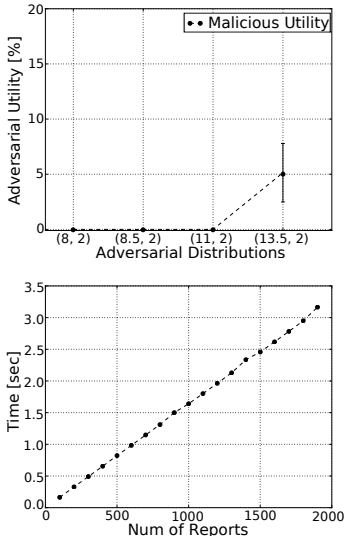


Fig. 10: (a) Adversarial Utility, (b) Performance Analysis of the Remuneration Protocol.

achieve (i.e., the number of queries they manage to arbitrarily pollute) as a function of M (i.e., the adversarial strength) and the estimation error for c_i (Sec. 6). Small estimation errors yield high utility for the adversaries. Nonetheless, as the estimation error grows, this utility significantly reduces.

We also compare our system to robust aggregation schemes proposed for wireless sensor networks. More specifically, we consider the *median* robust aggregation function, discussed in [54] and used in [31], and the *RANBAR* aggregation scheme [55]. We compare them to our system for different adversarial strategies. We set the fraction of adversary-controlled nodes to 25% (because it was shown that median performs well in this case [55]) and we assume adversaries report values from a normal distribution; we vary μ and fix σ to 3. Recall that honest users report (velocity) values from the $(\mu = 16, \sigma = 2)$ distribution. We assess the impact of adversaries by examining the distortion they impose to the system output. Figure 9 (c) shows that our system significantly outperforms both schemes for all examined adversarial strategies. More specifically, for adversarial distributions that significantly differ from the actual one, our system correctly identifies and removes malicious contributions, thus, achieving no distortion.

Remuneration Protocol Analysis & Efficiency. First we examine the (possible) utility achieved by malicious users

as a function of their strategy. More specifically, we employ the transportation dataset and set the fraction of adversarial users to 45%. The adversarial distributions are the same as the ones considered in Fig. 9 (d). Fig. 10 (a) presents our findings: adversaries with report values drawn from the $(\mu = 19, \sigma = 2)$ normal distribution can be easily detected by the RS, and, thus, the utility they receive is 0. As the adversarial distribution moves towards the actual one, detectability decreases and thus, the utility that adversaries receive from the remuneration protocol increases. Nonetheless, this increase is small (i.e., 5% of the total utility) especially when considering the adversarial strength (i.e., 45%). Recall that in this case adversaries are reporting almost truthful values: the distortion they inflict to the system output is negligible (0.03) (Fig. 9 (d)).

Fig. 10 (b) depicts the performance of the remuneration protocol ran on a mainstream workstation. We plot the execution time as a function of the number of user contributions received during a sensing task. We have also considered the time needed for the calculation of the score values of the reports (Sec. 4.1), their homomorphic encryption, randomization and digital signing[6] and, finally, their decryption by the GM (Sec. 4). As the figure shows, the remuneration protocol overhead scales linearly with respect to the number of received user reports. But still, it adds a modest complexity on the data verification framework: for example, it requires less than $3.5\ sec$ for 2000 user reports.

On the client side, homomorphic randomization is a simple addition of elliptic curve points. Moreover, the protocol requires each client to decrypt, with its private key, the cipher-text of $H_e\{\phi_i\}$ encrypted by the RS. A modest Quad-Core phone (we used Sony Z3) performs approximately 65 decryptions/sec. Finally, the size of a signed, randomized and homomorphically encrypted ϕ value is 320 bytes.

8 Conclusions

Technological advances in sensing, microelectronics and their integration in everyday consumer devices laid the groundwork for the rise of participatory sensing. However, its success requires effective protocols that guarantee security and privacy for MCS systems and their users. To meet this challenge, we presented a novel data verification protocol that ensure resilience against strong adversaries that pollute sensing campaigns. At the same time, our system enables the provision of incentives in a privacy-preserving manner; a catalyst for user participation. We formally evaluated the achieved security and privacy properties and provided a strong empirical evidence of its accuracy and efficiency.

References

[1] B. Guo et al. "From Participatory Sensing to Mobile Crowd Sensing". In: *CoRR* abs/1401.3090 (2014).

[2] G. Chatzimilioudis et al. "Crowdsourcing with Smartphones". In: *Internet Comp, IEEE* 16.5 (2012).

[3] M. V. Kaenel, P. Sommer, and R. Wattenhofer. "Ikarus: Large-scale Participatory Sensing at High Altitudes". In: *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*. Phoenix, USA, 2011.

6. The RS signs the randomized homomorphically encrypted values with 2048-bit RSA keys.

[4] D. Mendez et al. "P-Sense: A Participatory Sensing system for air pollution monitoring & control". In: *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Seattle, 2011.

[5] S. Gisdakis et al. "Secure and Privacy-Preserving Smartphone-Based Traffic Information Systems". In: *IEEE Transactions on Intelligent Transportation Systems* (2015), pp. 1428–1438.

[6] A. Thiagarajan et al. "VTrack: Accurate, Energy-aware Road Traffic Delay Estimation Using Mobile Phones". In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. Berkeley, USA, 2009.

[7] S. Gisdakis et al. "SEROSA: SERvice oriented security architecture for Vehicular Communications". In: *IEEE Vehicular Netw Conf*. Boston, USA, 2013, pp. 111–118.

[8] T. Giannetsos, T. Dimitriou, and N. R. Prasad. "People-centric sensing in assistive healthcare: Privacy challenges and directions". In: *Security and Communications Network* 4.11 (Nov. 2011), pp. 1295–1307.

[9] N. Lane et al. "BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing". In: *5th International ICST Conference on Pervasive Computing Technologies for Healthcare*. Dublin, Apr. 2012.

[10] J. Ballesteros et al. "Safe cities. A participatory sensing approach". In: *IEEE Conf on Local Computer Netw*. 2012.

[11] R. Gimenez et al. "Moving Advanced Safety to the Cloud: Some Outcomes of SafeCity Project." In: *Future Security*. Vol. 318. Communications in Computer and Information Science. Springer, 2012, pp. 85–88.

[12] G. Greenwald. *"NSA Prism Program Taps in to User Data of Apple, Google and Others"*. June 2013. URL: http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data.

[13] I. Krontiris, F. Freiling, and T. Dimitriou. "Location privacy in urban sensing networks: research challenges and directions". In: *IEEE Wireless Communications* 17.5 (Oct. 2010), pp. 30–35.

[14] S. Reddy et al. "Examining Micro-payments for Participatory Sensing Data Collections". In: *ACM International Conference on Ubiquitous Computing*. UbiComp. Copenhagen, Denmark, 2010.

[15] M. Shin et al. "AnonySense: A system for anonymous opportunistic sensing." In: *Pervasive and Mobile Computing* 7.1 (2011), pp. 16–30.

[16] E. De Cristofaro and C. Soriente. "Extended Capabilities for a Privacy-Enhanced Participatory Sensing Infrastructure (PEPSI)". In: *IEEE Transactions on Information Forensics and Security* 8.12 (2013), pp. 2021–2033.

[17] T. Das et al. "PRISM: platform for remote sensing using smartphones". In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. San Francisco, USA, 2010.

[18] L. Kazemi and C. Shahabi. "TAPAS: Trustworthy privacy-aware participatory sensing". In: *Knowledge and Information Systems* 37.1 (2013), pp. 105–128.

[19] S. Gisdakis, T. Giannetsos, and P. Papadimitratos. "SPPEAR: Security & Privacy-preserving Architecture for Participatory-sensing Applications". In: *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*. WiSec. Oxford, United Kingdom, 2014.

[20] D. Christin et al. "IncogniSense: An anonymity preserving reputation framework for participatory sensing applications." In: *Pervasive and Mobile Comp.* 9.3 (2013), pp. 353–371.

[21] Xinlei Oscar Wang et al. "ARTSense: Anonymous reputation and trust in participatory sensing." In: *32nd Int. Conference on Computer Communications*. Turin, Italy, 2013.

[22] T. Luo and C. K. Tham. "Fairness and social welfare in incentivizing participatory sensing." In: *IEEE Conf. on*

Sensor, Mesh and Ad Hoc Communications and Networks. Seoul, 2012.

[23] I. Krontiris and A. Albers. "Monetary incentives in participatory sensing using multi-attributive auctions". In: *International Journal on Parallel Emerging Distributed Systems* 27.4 (2012), pp. 317–336.

[24] S. Gisdakis, T. Giannetsos, and P. Papadimitratos. "SHIELD: A Data Verification Framework for Participatory Sensing Systems". In: *ACM Conference on Security & Privacy in Wireless and Mobile Networks*. New York, 2015.

[25] T. Giannetsos, S. Gisdakis, and P. Papadimitratos. "Trustworthy People-Centric Sensing: Privacy, Security and User Incentives Road-map". In: *IEEE 13th Mediterranean Ad Hoc Networking Workshop (Med-hoc-Net)*. Piran, Slovenia, 2014.

[26] J. Benaloh Clarkson. "Dense Probabilistic Encryption". In: *In Proceedings of the Workshop on Selected Areas of Cryptography*. 1994, pp. 120–128.

[27] J. H. Saltzer and M. D. Schroeder. "The protection of information in computer systems". In: *Proceedings of the IEEE* 63.9 (1975), pp. 1278–1308.

[28] L. Anselin and A. Getis. "Spatial statistical analysis and geographic information systems". In: *The Annals of Regional Science* 26.1 (1992).

[29] B. Hull et al. "CarTel: a distributed mobile sensor computing system". In: *International Conference on Embedded networked Sensor Systems*. Boulder, USA, 2006.

[30] K. Farkas et al. "Participatory sensing based real-time public transport information service". In: *IEEE International Conference on Pervasive Computing and Communications Workshops*. Budapest, Hungary, 2014, pp. 141–144.

[31] D. Mendez and M. A. Labrador. "On Sensor Data Verification for Participatory Sensing Systems". In: *Journal of Networks* 8.3 (2013), pp. 576–587.

[32] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy. "Mining Data Streams: A Review". In: *ACM Special Interest Group on Management of Data Record* 34.2 (June 2005).

[33] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[34] S. Garcia et al. "A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 25.4 (2013), pp. 734–750.

[35] M. Ester et al. "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *2nd International Conference on Knowledge Discovery and Data Mining*. Portland, OR, USA, 1996.

[36] G. N. Lance and W. T. Williams. "Mixed-Data Classificatory Programs I - Agglomerative Systems." In: *Australian Computer Journal* 1.1 (1967), pp. 15–20.

[37] L. Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32.

[38] J. Gama et al. "A Survey on Concept Drift Adaptation". In: *ACM Computing Surveys* 46.4 (Mar. 2014).

[39] P. Smets. "Data fusion in the transferable belief model". In: *3rd International Conference on Information Fusion*. Paris, France, 2000, pp. 21–33.

[40] P. Smets. "The Combination of Evidence in the Transferable Belief Model". In: *IEEE Trans on Pattern Analysis and Machine Intelligence* 12.5 (May 1990), pp. 447–458.

[41] D. Wörner and T. von Bomhard. "When Your Sensor Earns Money: Exchanging Data for Cash with Bitcoin". In: *ACM Int. Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. Seattle, Washington, 2014, pp. 295–298.

[42] D. Dolev and A. C. Yao. *On the security of public key protocols*. Tech. rep. Stanford University, 1981.

[43] B. Blanchet. "Automatic proof of strong secrecy for security protocols". In: *IEEE Symposium on Security*. 2004.

[44] R. Emil Kalman. "A New Approach to Linear Filtering & Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[45] B. Wiedersheim et al. "Privacy in inter-vehicular networks: Why simple pseudonym change is not enough". In: *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*. 2010.

[46] R. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. 1972.

[47] L.G. Valiant. "The complexity of computing the permanent". In: *Theoretical Computer Science* (1979), pp. 189 –201.

[48] Data Sensing Lab. *"Strata Santa Clara Dataset"*. Feb. 2013. URL: http://datasensinglab.com/data/.

[49] E. Miluzzo et al. "Tapping into the Vibe of the City Using VibN, a Continuous Sensing Application for Smartphones". In: *ACM Symp. From Digital Footprints to Social and Community Intelligence SCI*. Beijing, China, 2011.

[50] D. Krajzewicz et al. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 5.4 (Dec. 2012), pp. 128–138.

[51] S. Gisdakis and P. Papadimitratos. "On the Optimal Allocation of Adversarial Resources". In: *1st ACM Workshop on Mission-oriented Sensor Networks*. Istanbul, Turkey, 2012.

[52] S. Gisdakis, D. Katselis, and P. Papadimitratos. "Allocating adversarial resources in wireless networks". In: *21st IEEE EU Conf. on Signal Processing*. Marrakech, Morocco, 2013.

[53] R. J. G. B. Campello. "Generalized External Indexes for Comparing Data Partitions with Overlapping Categories". In: *Pattern Recogn. Lett.* 31.9 (July 2010), pp. 966–975.

[54] D. Wagner. "Resilient Aggregation in Sensor Networks". In: *2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*. Washington DC, USA, 2004, pp. 78–87.

[55] L. Buttyán, P. Schaffer, and I. Vajda. "RANBAR: RANSAC-based resilient aggregation in sensor networks". In: *4th ACM Workshop on Security of Ad-Hoc and Sensor Networks*. Alexandria, VA, USA, 2006, pp. 83–90.

Stylianos Gisdakis received his Diploma in Computer Science from Athens University of Economics and Business (AUEB) in 2008. He received his MSc degree in 2011 on Information and Communication Systems Security from the Royal Institute of Technology (KTH), Sweden, where he is currently working towards the Ph.D. degree in Networked Systems Security.

Thanassis Giannetsos earned his Ph.D. degree from University of Aalborg, Denmark in 2012. Currently he is a Senior Researcher at the Networked Systems Security group with KTH Royal Institute of Technology, Stockholm, Sweden. His research interests span from the theoretical foundations of cryptography to the design and implementation of efficient and secure communication protocols.

Panagiotis (Panos) Papadimitratos earned his Ph.D. degree from Cornell University, Ithaca, NY, in 2005. He then held positions at Virginia Tech, EPFL and Politecnico of Torino. Panos is currently an Associate Professor at KTH, where he leads the Networked Systems Security Group. His research agenda includes a gamut of security and privacy problems, with emphasis on wireless networks. Web page: www.ee.kth.se/nss