SHIELD: A Data Verification Framework for Participatory Sensing Systems

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

ABSTRACT

The openness of PS systems renders them vulnerable to malicious users that can *pollute* the measurement collection process, in an attempt to degrade the PS system data and, overall, its usefulness. Mitigating such adversarial behavior is hard. Cryptographic protection, authentication, authorization, and access control can help but they do not fully address the problem. Reports from faulty insiders (participants with credentials) can target the process intelligently, forcing the PS system to deviate from the actual sensed phenomenon. Filtering out those faulty reports is challenging, with practically no prior knowledge on the participants' trustworthiness, dynamically changing phenomena, and possibly large numbers of compromised devices. This paper proposes SHIELD, a novel data verification framework for PS systems that can complement any security architecture. SHIELD handles available, contradicting evidence, classifies efficiently incoming reports, and effectively separates and rejects those that are faulty. As a result, the deemed correct data can accurately represent the sensed phenomena, even when 45% of the reports are faulty, intelligently selected by coordinated adversaries and targeted optimally across the system's coverage area.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection; E.3 [**Data Encryption**]: Public Key Cryptosystems

General Terms

Experimentation, Performance, Security

Keywords

Participatory Sensing; Security; Privacy

1. INTRODUCTION AND BACKGROUND

Participatory Sensing (PS) [1] leverages the proliferation of modern sensing-capable devices enabling (wide-scale) information collection practically from *anywhere*, at *anytime*, about *anything*. PS has incubated a wide gamut of applications aiming to improve the welfare of individuals and the general public: environmental monitoring [2, 3], urban sensing [4, 5], intelligent transportation [6, 7, 8, 9], assistive health-care [10, 11] and public safety [12, 13].

PS users are *information prosumers* [14]: they *produce* and *consume* information. As producers, they deliver content to the system, contributing sensed data with their smartphones, smart-vehicles, wearable sensor vests and armbands. Such information is sensitive: it reveals their location, daily routines and social relations and it can be used to profile them. Sensible users valuing their privacy might opt-out or even oppose systems that harm their privacy [15, 16]. Therefore, strong privacy protection, combined with incentive mechanisms, can facilitate mass user participation [14].

At the same time, to *attract* and *retain* information consumers, PS systems must create added value by providing accurate information and intelligence. Unlike the deployment of dedicated sensors, PS systems can have much broader spatial coverage. The broader the participation, the better the results, in principle. But this openness is a double-edge sword: any of the participants can be adversarial and *pollute* the collected data, seeking to manipulate (or even dictate) the PS system output. Faulty, distorted information can lead to wrong decisions, possibly rendering PS systems useless. Security mechanisms can mitigate such misbehavior to a certain extent: authentication, authorization and access control can prevent *outsiders* (i.e., external entities, not part of the sensing campaign) from polluting the sensing task. This has led to numerous proposals aiming at securing PS data collection [17, 18, 19, 20, 21, 22].

However, what happens when *registered PS users (insiders) attack* the system? Insiders, that is, compromised devices equipped with the system credentials and cryptographic keys, can still (easily) pollute PS data. Worse even, as PS gets widely adopted and mobile devices are being infected, in large numbers, by malware [23], one cannot expect that such internal adversaries will be a small minority in the system. *This implies that security cannot alone guarantee the collection of truthful and trustworthy data.*

The eviction of misbehaving users is possible [22], but it necessitates fine-grained and node-specific identification of offending behavior. This can be time-consuming and hard against intelligent and numerous adversaries. Even harder it is to rely on reputation [24, 25]: users would need to have their reputation built up based on the trustworthiness of their data. 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*

WiSec'15, June 22 - 26, 2015, New York, NY, USA.
 ISBN 978-1-4503-3623-9/15/06 ...$15.00.
DOI: http://dx.doi.org/10.1145/2766498.2766503.

trustworthiness of their source. Designing such mechanisms is not easy: it requires fusing (contradictory) data, originating from untrustworthy sources describing dynamic and uncertain phenomena evolving over space and time.

Statistical anomaly detection could mitigate pollution attacks: adversary-free statistical models of the sensed phenomena, e.g., estimated based on past sensing campaigns, can be empirically compared to incoming user reports. Nonetheless, such models are hard to obtain for every monitored phenomenon (e.g., temperature or traffic congestion) in every possible geographical area. Furthermore, even if such models existed, they would not capture (frequent) changes of the underlying phenomena or extraordinary events. For example, a model describing traffic congestion levels is not valid in the presence of accidents or road maintenance activities. We need to assess user-contributed information *without prior statistical descriptions or models of the data to be collected*. The absence of such prior knowledge has been considered in [26, 27]. Nonetheless, these works are limited to scenarios where the goal of the system is to solely infer the presence or absence of (boolean) conditions of interest (e.g., damage, pollution, litter).

Other works focus on mitigating data manipulation in the domain of Wireless Sensor Networks (WSNs), notably seeking to render data aggregation robust. Resilient aggregation functions [28, 29] thwart data pollution by removing some of the faulty data without relying on prior data knowledge. The challenge is to have a scheme that *effectively removes most, if not all, of the faulty data*. An alternative approach, also investigated in the WSN context, is to leverage node-to-node communication to detect and filter out faulty data [30, 31, 32]. However, this approach is not applicable to the open, volatile, and voluntary PS context. Existing PS applications require user-PS system communication alone and any additional user-to-user communication can be a burden (e.g., data transmission cost, battery, complexity). What we need, for easy adoption, is to have user mobile devices implementing the *bare minimum of functionality on the user (client) side*, placing the load for safeguarding the data collection on the server side.

Contributions: We meet these challenges with SHIELD; a novel data verification framework that combines evidence handling (Dempster-Shafer Theory) and data-mining techniques to identify and filter out erroneous (malicious) user contributions. More specifically, SHIELD (*i*) is generic and applicable to any type of PS task, (*ii*) it guarantees no (or small) distortion of the PS system output even in the presence of strong, colluding adversaries, (*iii*) it adapts to spatio-temporal changes of the underlying phenomena, (*iv*) it enables users to perform fine-grained queries, and (*v*) it provides information that can be used for decision making. We provide a prototype implementation of SHIELD and we extensively evaluate its performance under various scenarios, employing both real and synthetic datasets. We also show how it outperforms various existing resilient aggregation and outlier detection schemes. Overall, our framework can complement any security architecture for PS, closing the existing gap and protecting against powerful internal adversaries.

In the rest of the paper, we first describe the system and adversarial models (Sec. 2). We then provide an overview of SHIELD, followed by a detailed description of its core components (Sec. 3). Sec. 4 outlines the experimental setup used to evaluate SHIELD, with results presented in Sec. 5, before we conclude (Sec. 6).

2. SYSTEM AND ADVERSARY MODEL

We consider a generic Participatory Sensing (PS) [14] system consisting of:

Users: Set of individuals using 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). Mobile devices collect and report sensory data to the PS infrastructure across any available network (e.g., 3/4G, WiFi). Users (devices) also query the results of sensing tasks.

Campaign Administrators: Organizations, public authorities or individuals [33], initiating data collection campaigns: they recruit users and distribute descriptions of sensing tasks to them.

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 require a *Sybil-proof* infrastructure, which guarantees that no registered user can obtain multiple identities and credentials valid simultaneously; [19, 22] provide these features and guarantees.

Reporting Service (RS): The RS exposes the interfaces that enable registered users to submit data to and query the results of a sensing task. SHIELD operates on the RS, to assess and remove invalid, faulty data.

The *area of interest* of a sensing task is the locality within which users contribute data. It is defined explicitly (i.e., by means of coordinates forming polygons on maps) or implicitly (through annotated geographic areas, e.g., New York city). In either case, the area of interest is divided into *spatial units*. Spatial units are homogeneous, with respect to the sensed phenomenon, areas [34]. More specifically, the value of the sensed phenomenon within a spatial unit might exhibit temporal but not significant spatial variations. The definition of spatial units largely depends on the PS application and, hence, on the underlying monitored phenomenon. For instance, in the context of traffic information systems (e.g., [6, 7]), road links (i.e., road segments between two junctions) serve as spatial units. Similarly, for public transport PS applications (e.g., [35]) spatial units can correspond to individual bus, metro and train lines. Finally, pollution areas around points of interest (e.g., factories) can serve as spatial units [3] for environmental monitoring tasks.

Each user submits to the RS a stream of measurements on the sensed phenomenon over a time interval, t, specified in the PS campaign [14]. These data are submitted in successive *reports*, each with n measurements, v_i where $i \in \{1, 2, ..., n\}$, corresponding to a device location, *loc*:

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

σ_{PrvKey} is a digital signature with some private key, with the corresponding public key included in the certificate C. User-RS communication is over an end-to-end secure channel. We do not dwell further on the identity management and cryptographic protection specifics; we assume these are addressed, e.g., by the most recent scheme [22] in the literature.

Adversary model: Malicious users (clients) can participate in sensing campaigns: having obtained valid credentials, they can submit authenticated yet faulty reports to the RS. We do not refer only to human operators with malevolent intentions, but more generally to compromised devices (clients),

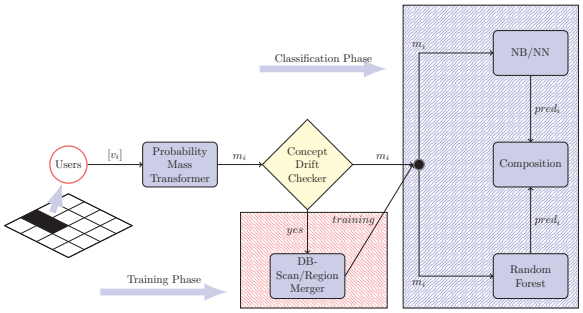


Figure 1: System Overview

e.g., imbued with a rogue version of the PS application. Each such compromised client (essentially, the corresponding identity, certificate, and private key) can *pollute* the data collection process. This can result in the *distortion* of the PS system's perception of the sensed phenomenon, and thus degrade the usefulness of the campaign (e.g., causing wrong answers to user queries). Consider, for example, traffic monitoring campaigns, with users submitting their velocity to the RS: adversaries could submit low velocities to impose a false perception of the congestion levels of the road network (e.g., traffic jams or accidents).

We allow adversaries to submit values (arbitrarily) different than the ones of the honest users (i.e., the values 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 impose a V_{r_i} that deviates from V_{a_i}. In extremis, the adversaries can seek 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 follow the same strategy, i.e., submit reports drawn from the same set of values, or even optimally select how (where) to inject faulty reports. This is possible because adversaries can *spoof* (forge) their (device) location and submit reports for regions they are not physically present but they want to affect. We elaborate the attack instantiation in Sec. 4.

SHIELD objective: In the presence of such adversaries, we seek to safeguard the *accuracy* of the RS perception of the sensed phenomenon. In other words, minimize the distortion imposed by adversaries:

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

Note that SHIELD is agnostic to the cause of faulty reports; these can so be due to benign impairments of the PS clients (devices operated by honest users). In what follows, we do not distinguish deliberately and unintentionally faulty reports.

3. SHIELD FRAMEWORK

3.1 High-Level Overview

In a nutshell, 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 phase, user reports are classified as *inlying* or *outlying*, also termed *inliers* and *outliers*, 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 is an exploration of their innate structure (Sec. 3.3). 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. 3.2) and a feature vector, one for each report, is created and classified. Bootstrapping is run separately for each spatial unit (Sec. 2) (recall: measurements of honest users follow the same distribution within a spatial unit).

At the second phase, SHIELD explores the spatial characteristics of the sensed phenomenon and merges spatial units into larger regions. To do this, SHIELD derives an empirical distribution for all data from all inlying reports within a spatial until. Then, it examines the statistical similarity of empirical distributions for neighboring spatial units and merges them if they are deemed similar. This way *region merging* (Sec. 3.4) creates larger and homogeneous (with respect to the phenomenon) geographical regions. For each of these regions, *training* is performed: this process 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 given into an *ensemble* of classifiers (Sec. 3.5). This ensemble is a supervised learning mechanism that 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 may change over time in unforeseen ways. Such events, known as *concept drifts* [36], can deteriorate the performance of any classification model. SHIELD leverages a triggering mechanism that detects and adapts to such changes (Sec. 3.6).

3.2 Handling Evidence

The SHIELD-ed RS is an agent that *reasons* on the actual value of the sensed phenomenon, relying on multiple sources of evidence (i.e., reports). For example, RS has to decide whether the congestion level of a street is "high", based on the velocities that participating users report, or that the temperature of an area is within the interval $10\,^{\circ}\text{C} - 11\,^{\circ}\text{C}$, based on temperature measurements. This is essentially a *decision making* and a *sensor fusion problem*. The canonical approach for such problems is *Bayesian Inference* (BI), which computes the posterior probability of a hypothesis given the available supporting evidence and its prior probability. Nonetheless, the definition of prior probabilities is an obstacle for using BI in the context of PS. For example, it is hard to define the prior probability of a hypothesis that the road is congested or that the temperature (of an area) is within some interval. Instead, we need to handle *uncertainty*: having a user report stating that temperature is within an interval α does not preclude temperature from lying in another interval β (as is the case for BI). SHIELD leverages *Dempster-Shafer Theory* (DST) [37] that allows reasoning about uncertainty; we provide basic notions of DST next.

3.2.1 Use of Dempster-Shafer Theory

Let Θ be 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:

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

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

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 the 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 sensing task of a phenomenon that takes the abstract values a, b, c and d (either numerical or nominal). As it was described in Sec. 2, each participating user submits a stream of measurements of the sensed phenomenon. The transformation of these streams to probability masses depends on the phenomenon itself. First, if the system wants to infer the value of the same phenomenon that users monitor, it generates a probability density function (p.d.f) by constructing a *normalized histogram* [39] 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, in case of temperature monitoring tasks we assume 1°C intervals. This, however, has an impact on the system performance as 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., $0 - 32$°C)[1]. Moreover, histogram bins do not need to have the same granularity. One bin could include temperature measurements < 0°C (i.e., an unlikely hypothesis during summer months in cities of the northern hemisphere); another temperatures > 30°C. Intermediate bins, corresponding to more probable hypotheses, can have .5°C granularity (e.g., $0.5, 1, 1.5, ..., 30$°C).

Algorithm 1 Pseudocode for DBSCAN [38]

```
1: procedure DBSCAN(D, ε, MinPoints)
2:     Cluster ← empty
3:     for each unvisited feature f of D do
4:         f ← visited
5:         Neighbors ← QUERYREGION(f, ε)
6:         if size(Neighbors) < MinPoints then
7:             f ← outlier
8:         else
9:             EXPAND(f, Neighbors, Cluster, ε, MinPoints)
10:
11: procedure QUERYREGION(f, ε)
12:     return ε−neighborhood of f
13:
14: procedure EXPAND(f, Neighbors, Cluster, ε, MinPoints)
15:     f ← member(Cluster)
16:     for each n in Neighbors do
17:         if n  visited then
18:             n ← visited
19:             Neighbors′ ← QUERYREGION(n, ε)
20:             if size(Neighbors′) ≥ MinPoints then
21:                 Neighbors + = Neighbors′
22:         if n ∉ any cluster then
23:             n ← memberof(Cluster)
```

SHIELD can also infer the value of a phenomenon P_α, based on measurements of a phenomenon P_β. For example, users monitor air-temperature but their measurements are used to infer soil-temperature.[2] An air-temperature measurement, $v_{\alpha,i}$, can correspond to multiple soil-temperature values: e.g., $v_{\beta,x}, v_{\beta,y}, v_{\beta,z}$. In this case, the system treats each $v_{\alpha,i}$ as supporting evidence for the $v_{\beta,x}, v_{\beta,y}, v_{\beta,z}$ hypotheses. We term such sensing tasks as *Indirect Inference (IInf)*.[3]

Let us assume that the mass of c_1 is the vector $m_{c_1} = [ab : 0.6, bc : 0.3, a : 0.1, ad : 0.0]$. This states that 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, let the mass of c_2, be $m_{c_2} = [ab : 0.5, bc : 0.4, b : 0.05, a : 0.05]$. Using the combination rule (5), we get that the combined mass is $m = m_{c_1} \oplus m_{c_2} = [b : 0.46, ab : 0.32, cb : 0.12, a : 0.09]$. We can compute the plausibility that the value of the phenomenon is a. Using (4) we get that $Pl_m(a) = 0.41$. Finally, (6) yields that the conflict between m_{c_1} and m_{c_2} is 0.06.[4]

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

[2] Possibly based on an inference model that leverages the empirical relationships between the two phenomena.

[3] Eq. (7) becomes $LCon(m) = \sum_{A \in \Theta} m(A) log_2(|A|) - \sum_{A \in \Theta} m(A) log_2(m(A))$ [40].

[4] The numbers used in this example have been rounded for easy readership.

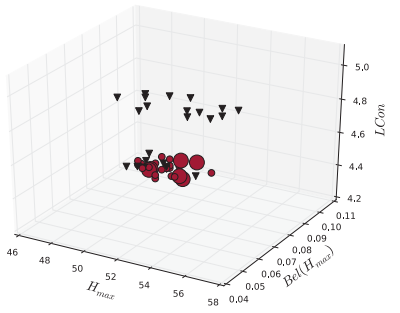


Figure 2: Clustering with DBSCAN (circles denote inliers).

3.3 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. 3.2.1). The system waits until a sufficient number of reports has been collected for a spatial unit[5] and then initiates the DBSCAN density-based topological clustering algorithm [38] (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 [41]:

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

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), 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 [38]. This heuristic leverages the $k - distance$ of the dataset to compute the ϵ and $MinPoints$ of the thinnest cluster created by the algorithm.[6].

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.

Since DBSCAN examines the topological density of the feature vectors, it relies on *honest* majorities to (correctly) identify inliers and outliers. Simply put, it learns what the majority of users suggest for each spatial unit. Nevertheless, as we discuss in Sec. 5, such honest majorities can be rather marginal.

[5]This can be identified heuristically; based on our evaluations, 20 reports suffice.

[6]Additional details are presented in [38]

Algorithm 2 Pseudocode for the Region Growing Algorithm

```
1: procedure REGIONGROWING(Seed s)
2:     s ← visited
3:     region ← ∅
4:     region.add(s)
5:     active_set ← s
6:     while active_set not ∅ do
7:         c = dequeue_point(active_set)
8:         for n in neighbors_of(c) do
9:             if n(¬visited) and KSTEST(c, n) : True then
10:                active_set.add(n)
11:                region.add(n)
12:            visited ← n
       return region
```

3.4 Region Merging and Training Phases

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. 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 of the sensing task. The worst case scenario is when the monitored phenomenon exhibits extreme spatial diversity; then, no (larger) regions will be formed and, thus, each spatial unit will be considered as a separate region.

For illustration, Fig. 3 shows an execution instance of this algorithm for an emulated traffic sensing campaign with participating users contribute their velocities to the RS (Sec. 4). 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). 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 (annotated with red color) based on the homogeneity of the reported velocities. The color density of the road links serves as an indication of the average speed users experience on each road segment (darker colors indicate smaller velocities).

Once the region merging phase concludes, the training takes place 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.

SHIELD can operate for any phenomenon irrespective of the underlying distribution. 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 able to describe *any* type of distribution. This, in fact, makes SHIELD generic and relevant to any physical phenomenon.

3.5 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. 3.3), 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 [42] 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 together resulting into the final estimation of the random forest.[7]

The classification rule for SHIELD's 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 and outlying reports for the training set. The NN classifier implements a simple k-nearest neighbors voting scheme (in our case k is set to 10). All classifiers are trained with the output of the previous phase which is the labeling of inlying and outlying reports for each region.

[7]Due to space limitations we omit the details of this process and we refer the interested readers to the relevant work [42].

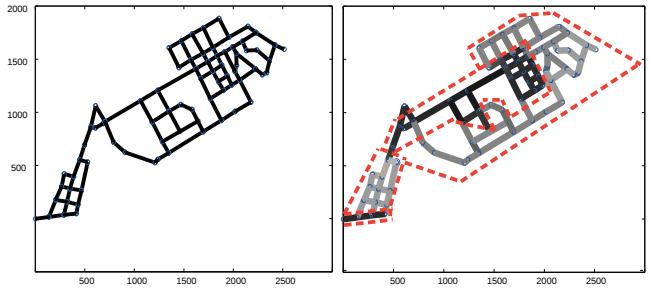


Figure 3: Illustration of the output of the Region Merging algorithm for a traffic monitoring PS campaign.

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 combination (according to Dempster's rule of combination) of all probability masses of all inlying reports in a region.

3.6 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) described in Sec. 3.2. 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 occur locally and concern only one, or a small number of regions. In this case, the system retrains (i.e., it enters the Training Phase described in Sec. 3.4) the classifiers of these regions (in question) with the new data it received from the users.

Global Concept Drifts: They occur globally and 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.

3.7 Querying SHIELD

Any authenticated entity can query the system sending:

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

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 C. 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:

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

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 sensing task) can assess the degree of agreement (or contradiction) between their measurements and the system's belief, by issuing:

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

SHIELD 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:

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

Queriers can then perform a *pignistic transformation* [43] of m; a transformation of the system's belief function to a probability function that can yield optimal decisions [44].

4. EXPERIMENTAL SETUP

Since SHIELD is a composition of machine learning mechanisms and heuristics, getting theoretical bounds on the quality of the system is intractable. Instead, we provide strong empirical evidence on the performance of the system, in various scenarios, leveraging both real-world and synthetic datasets.

Datasets - The real-world dataset relates to PS-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 ability of SHIELD to accurately classify incoming reports and yield a truthful and undistorted view of the underlying phenomenon.

As *real-world measurements*, we use Strata Clara (SC) dataset, from the *Data Sensing Lab* [45], as a reference point in the domain of environmental monitoring applications [2, 5]. It contains raw measurements for 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 based on simulations of urban road links, emulating a traffic monitoring PS task [9]: 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 [46] 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 - We emulate adversaries by injecting 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 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. 3.2) and randomly distribute probability masses to them.

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; it is important to reflect the innate uncertainty of the sensed phenomena in the system's output.

For "normal" adversaries the employed (μ, σ) determine the similarity (i.e., the overlap) between the honest and adversarial distributions and it can be computed by simple numerical integration. This overlap serves as an indication of the detection difficulty: the bigger the overlap is, the harder it becomes to detect and sift malicious data. Nevertheless, even in the case of highly similar distributions, SHIELD manages to correctly identify both the honest and malicious samples (Sec. 5).

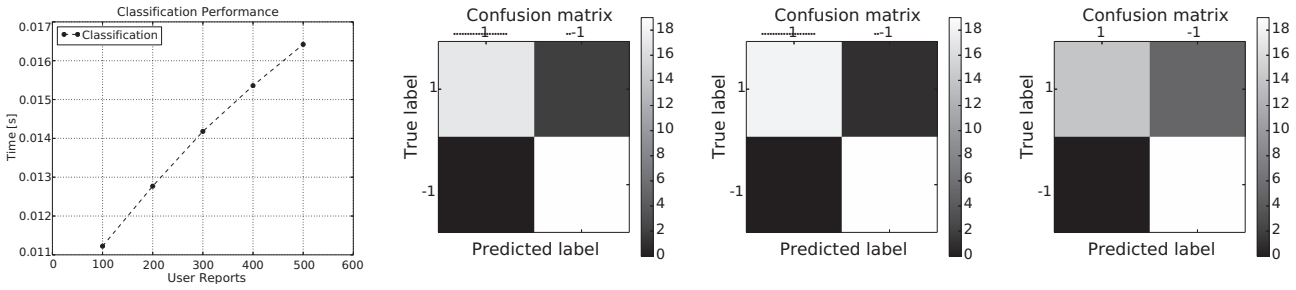
Examples of such strategies can be found in Appendix A. We do not consider malicious users acting independently with different strategies: in that case, their effect on the system would be significantly smaller compared to collaborative attackers; *the more reports (users) support the same hypotheses, 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 harm (distortion) they inflict on the system. 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 may be more easily characterized as faulty (outlying).

For global attacks, we assume adversaries do not simply cooperate (i.e., decide on a common distribution) but they also jointly decide on the optimal allocation of their faulty reports across the different regions, in order to maximally affect the system [47, 48]. 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:

$$\begin{aligned} \text{Maximise:} \quad & \textstyle\sum_{i=1}^{N} x \cdot p_i \\ \text{subject to:} \quad & \textstyle\sum_{i=1}^{N} x \cdot c_i \leq M, \\ & x \in \{0, 1\} \end{aligned}$$

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



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

Figure 4: Performance analysis of the classification algorithm (a). Confusion matrices for humidity ($\mu = 31, \sigma = 5$), sound ($\mu = 3, \sigma = 2$) and temperature ($\mu = 21, \sigma = 1.3$) sensors (b, c, d).

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.

5. RESULTS AND ANALYSIS

First, we assess the efficiency of SHIELD because PS applications can generate massive amounts of data. We then analyze its accuracy in the presence of adversaries. Our focus is on SHIELD'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 [49]: (*i*) *precision*, (*ii*) *recall*, (*iii*) *F-score*, (*iv*) *Matthew's correlation coefficient* (MCC) and (*v*) *Jaccard similarity score* (Appendix B). These indices measure different aspects of a system's classification accuracy and of its ability to correctly identify instances of a certain class (i.e., either inliers or 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. We conclude with an assessment of our framework's reactiveness to concept drifts.

In each experiment, SHIELD is provided with reports originating from both honest and malicious users (Sec. 4). This dataset is partitioned into two sub-sets: a *training* set (TS) and an *evaluation* set (ES). Based on the TS, the bootstrapping (Sec. 3.3) and training (Sec. 3.4) phases take place. Then, the ES is used to assess the performance of the supervised classification part (Sec. 3.5). We refrain from assessing the accuracy of DBSCAN as it is reflected on the performance of the unsupervised classification: better training yields better classification results. For each simulation we perform ten-fold cross validation [50] to avoid *overfitting*. We show results based on both datasets (Sec. 4); due to space limitations, we do not repeat similar figures from both datasets. The results show that SHIELD is equally effective for both datasets.

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 SHIELD's efficiency during the training phase. This is important for highly dynamic and regularly changing phenomena (entailing many concept drifts) that require retraining of the system.

As sensing tasks proceed, large amounts of data will be contributed by users. These data must be examined by the ensemble of classifiers (Sec. 3.5) in an efficient manner. Fig. 4 (a) depicts 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. For instance, the classification of 200 user reports requires less than 0.013 *sec* whereas 500 concurrent reports are classified in less than 0.017 *sec*. To obtain these measurements we deployed SHIELD on a commodity server with an 8-Core, 3.6 GHz CPU. System components were implemented with the *Scikit-learn* library.[8]

Classification Accuracy. We begin with the SC dataset to evaluate the accuracy of SHIELD. We are interested in its ability to correctly assess (i.e., classify) the samples submitted by both honest and malicious users (i.e., positive and negative labels, respectively). We perform separate tests for each sensed phenomenon (i.e., humidity, sound and temperature).

Classification accuracy can be represented in a structure termed *confusion matrix* or *contingency table*. Each column of the matrix shows the instances in a predicted class (1 for positive, i.e., *inlying* and -1 for negative, i.e., *outlying*, reports), while each row shows the instances in an actual class. A confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in $class_i$ but labeled, by the ensemble of classifiers, to be in $class_j$. Essentially, C shows the true positives (TPs), false positives (FPs), true negatives (TNs) and false negatives (FNs).

The confusion matrices for the humidity, sound and temperature sensors are shown in Fig. 4 (b), (c) and (d). Here, we assume that 45% of the users are malicious, representing an equal percentage of faulty data injected into the dataset. The distributions, from which the malicious samples were drawn, are: $\{(\mu = 31, \sigma = 6)|(\mu = 3, \sigma = 1)|(\mu = 21, \sigma = 3)\}$, respectively. The diagonal elements (in the matrices) show the number of correct classifications made for each class, and the off-diagonal elements indicate the errors.

SHIELD is almost 100% successful in correctly classifying both the inlying (i.e., *positive*) and outlying (i.e., *negative*) reports. For instance, in the case of humidity readings, 18 out of 19 samples were classified positive correctly and all true negatives were identified without error. Overall, the true negative classification rate remains high in all cases and only

[8] http://scikit-learn.org

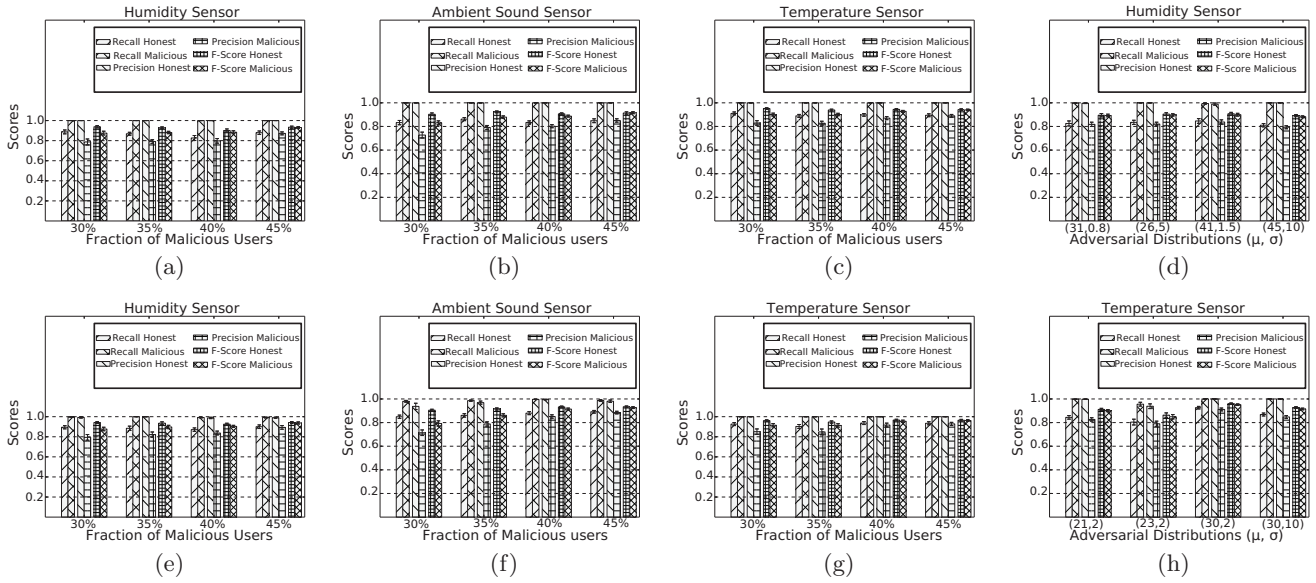


Figure 5: Precision, Recall and F-score metrics, for *uniform* (a, b, and c), *N-value* (e, f, and g) and *normal* (d, h) adversarial strategies. *SC dataset.*

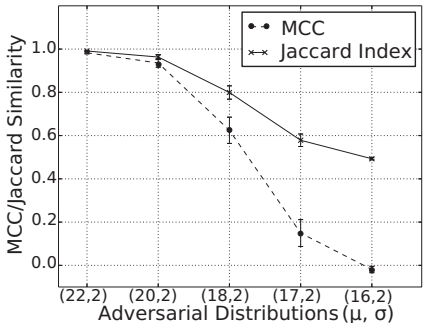


Figure 6: MCC and Jaccard coefficients as a function of the percentage of overlapping regions. *Traffic dataset.*

the false positive rate shows a variation, depending on the percentage of overlapping regions between the honest and malicious distributions (Sec. 4). Nevertheless, the observed variation is small even when the underlying distributions are rather similar (e.g., in the case of temperature).

Fig. 5 shows the precision, recall and F-score metrics when different adversarial strategies are employed (i.e., uniform, N-value and normal distributions). The bars depict the score values for both honest and malicious reports, respectively. As we can see, the overall correctness of SHIELD 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, it might seem interesting that higher accuracy is achieved for larger number of adversaries ($\geq 35\%$). In fact, this is because the number of negative samples increases in the evaluation set and, thus, mis-classifying one such sample will have a smaller impact on the overall precision.

Fig. 5 (d) and (h) take a closer look at SHIELD's accuracy when adversaries follow a normal distribution (for the humidity and temperature sensors, respectively). Here, 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), SHIELD's accuracy remains high (F-score ≥ 0.78).

We further examine the impact of overlapping regions (Fig. 6) for the synthetic datasets generated from the emulated traffic monitoring sensing task (Sec. 4). Honest users report their velocities drawn from a normal distribution with $(\mu = 16, \sigma = 2)$. For each simulation 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 SHIELD achieves perfect prediction when the overlap between the distributions is relatively small (i.e., their similarity is less than 35%). When it is around 50%, the classification accuracy still remains high (MCC, Jaccard) ≥ 0.85. But even for high overlap percentages ($\geq 80\%$), the SHIELD 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, SHIELD 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

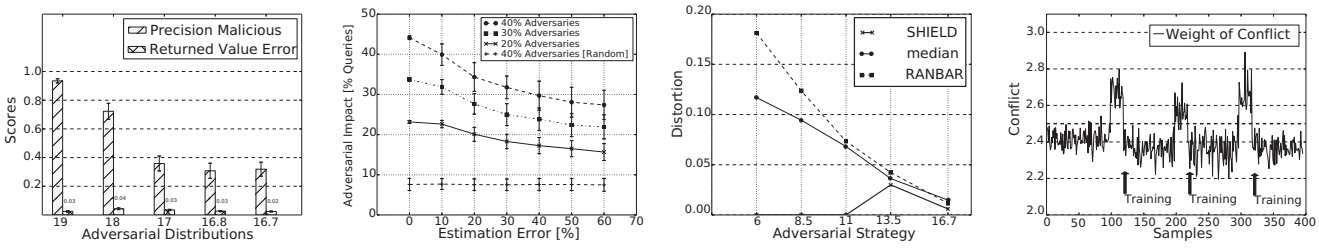


Figure 7: Analysis of Local (a) and Global (b) Adversarial Impacts. Comparison of SHIELD with robust aggregation functions (c) and Concept Drift detection (d). *Transportation dataset.*

measure of the phenomenon's uncertainty - the impact on SHIELD's classification accuracy is negligible ($\leq 10\%$).

Impact of Adversaries. Considering local attacks, within a region, we measure the detectability of adversarial data through the precision metric. Again, the fraction of adversaries (or, equivalently, adversarial reports) is set to 45% of the total number of users (reports) in the region. Moreover, 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. 7 (a) shows, adversaries with report values drawn from the $(\mu = 19, \sigma = 2)$ normal distribution can be easily detected by SHIELD, and, thus, the distortion of the system output is negligible (0.03). As the adversarial distribution moves towards (resembles to) the actual one, detectability decreases (i.e., 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. 4). 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 7 (b) shows the utility that adversaries 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. 4). Small estimation errors yield high utility for the adversaries. Nonetheless, as the estimation error grows, this utility significantly reduces.

We also compare SHIELD to robust aggregation schemes proposed for wireless sensor networks. More specifically, we consider the *median* robust aggregation function, discussed in [28] and used in [3], and the *RANBAR* aggregation scheme [29]. We compare them to SHIELD 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 [29]) 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 7 (c) shows that SHIELD significantly outperforms both schemes for all examined adversarial strategies. More specifically, for adversarial distributions that significantly differ from the actual one, SHIELD correctly identifies and removes malicious contributions, thus, achieving no distortion.

Concept Drift Adaptation. Fig. 7 (d) shows SHIELD's concept drift detection process. We demonstrate the triggering mechanism based on the observed weight of conflict between the probability mass the system assigns to each hypothesis (for a region) and the masses of incoming reports (Sec. 3.6). For this experiment, we modeled the underlying data as synthetic streams that change over time, in an unexpected and unpredictable (for the system) manner. The conflict threshold value was set to 2.5 (Eq. 6). Moreover, concept drifts were simulated every 100 samples.

Our model succeeds in maintaining an up-to-date understanding of the underlying phenomenon. Once it detects that the incoming reports do not conform to its current view, the previous prediction model is deemed obsolete and it retrains the employed classifiers. After retraining itself, subsequent classifications start again to converge to the actual values. Furthermore, all incoming reports that led to this drift detection (and were, initially, falsely classified as outlying), are reconsidered with the retrained prediction model.

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 [51] 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.

6. CONCLUSIONS

We presented SHIELD, a novel PS data verification framework, for resilience against strong adversaries that pollute sensing campaigns, efficient adaptation to the underlying phenomena spatio-temporal changes, and fine-grained user querying. Our prototype implementation and extensive evaluation under various realistic scenarios demonstrate its resilience and practicality, with SHIELD outperforming existing aggregation and outlier detection schemes. Towards future work, the nearly agnostic SHIELD can break, in conjunction with a security architecture, the circular dependency of data and user trustworthiness: simply put, SHIELD-deemed outlying reports can be linked to the device (credentials) and allow the security architecture to adjust the device trustworthiness (reputation), and possibly evict it.

References

[1] J. Burke et al. "Participatory sensing". In: *Workshop on World-Sensor-Web: Mobile Device Centric Sensor Networks and Applications*. Boulder, USA, 2006.

[2] D. Mendez et al. "P-Sense: A Participatory Sensing system for air pollution monitoring & control". In: *IEEE Conf. on Pervasive Computing-Communications*. Seattle, USA, 2011.

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

[4] L. Deng and L. P. Cox. "LiveCompare: Grocery Bargain Hunting Through Participatory Sensing". In: *10th ACM Workshop on Mobile Computing Systems and Applications*. Santa Cruz, California, 2009.

[5] E. Miluzzo et al. "Tapping into the Vibe of the City Using VibN, a Continuous Sensing Application for Smartphones". In: *1st ACM Int. Symp. From Digital Footprints to Social and Community Intelligence*. Beijing, China, 2011.

[6] B. Hull et al. "CarTel: a distributed mobile sensor computing system". In: *4th International Conference on Embedded Networked Sensor Systems*. Boulder, USA, 2006.

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

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

[9] S. Gisdakis et al. "Secure and Privacy-Preserving Smartphone-Based Traffic Information Systems". In: *IEEE Trans. on Intelligent Transportation Systems* PP.99 (2014), pp. 1–11.

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

[11] N. Lane et al. "BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing". In: *5th ICST/IEEE Int. Conf. on Pervasive Computing Technologies for Healthcare*. Dublin, 2012.

[12] J. Ballesteros et al. "Safe cities. A participatory sensing approach". In: *37th IEEE Conf. on Local Computer Networks*. Florida, USA, 2012, pp. 626–634.

[13] Roberto 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.

[14] T. Giannetsos, S. Gisdakis, and P. Papadimitratos. "Trustworthy People-Centric Sensing: Privacy, security and user incentives road-map". In: *13th Mediterranean Ad-Hoc Networking Workshop*. Piran, Slovenia, 2014.

[15] E. Mills. *"Google sued over Android data location collection"*. 2011. URL: `http://news.cnet.com/8301-27080_3-20058493-245.html`.

[16] J. Lowensohn. *"Apple sued over location tracking in iOS"*. 2011. URL: `http://news.cnet.com/8301-27076_3-20057245-248.html`.

[17] I. Boutsis and V. Kalogeraki. "Privacy Preservation for Participatory Sensing Data". In: *IEEE Conf. on Pervasive Computing & Comm.* California, USA, 2013, pp. 103–113.

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

[19] 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.

[20] T. Das et al. "PRISM: platform for remote sensing using smartphones". In: *8th Int. Conf. on Mobile Systems, Applications & Services*. S. Francisco, USA, 2010, pp. 63–76.

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

[22] S. Gisdakis, T. Giannetsos, and P. Papadimitratos. "SPPEAR: Security & Privacy-preserving Architecture for Participatory sensing Applications". In: *7th ACM Conf. on Security and Privacy in Wireless & Mobile Networks*. Oxford, UK, 2014, pp. 39–50.

[23] Y. Zhou and X. Jiang. "Dissecting Android Malware: Characterization and Evolution". In: *33rd IEEE Symp. on Security and Privacy*. S. Francisco, USA, 2012, pp. 95–109.

[24] 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.

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

[26] Dong Wang et al. "Recursive Fact-Finding: A Streaming Approach to Truth Estimation in Crowdsourcing Applications". In: *33rd IEEE Int. Conf. on Distributed Computing Systems*. Philadelphia, USA, 2013, pp. 530–539.

[27] Dong Wang, Lance Kaplan, and Tarek F. Abdelzaher. "Maximum Likelihood Analysis of Conflicting Observations in Social Sensing". In: *ACM Transactions on Sensor Networking* 10.2 (Jan. 2014), 30:1–30:27.

[28] David 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.

[29] 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.

[30] P. Haghani et al. "Efficient and Robust Secure Aggregation for Sensor Networks". In: *Proceedings of the Third IEEE ICNP Workshop on Secure Network Protocols (NPSec)*. Beijing, China, 2007, pp. 1–6.

[31] Bo Sheng et al. "Outlier Detection in Sensor Networks". In: *8th ACM Int. Symposium on Mobile Ad Hoc Networking and Computing*. Montreal, Quebec, Canada, 2007.

[32] F. Liu, X. Cheng, and D. Chen. "Insider attacker detection in wireless sensor networks". In: *26th IEEE Int. Conf. on Computer Communications*. Anchorage, Alaska, USA, 2007.

[33] D. Christin et al. "A survey on privacy in mobile participatory sensing applications". In: *Journal of Systems and Software* 84.11 (2011), pp. 1928 –1946.

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

[35] 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.

[36] 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).

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

[38] 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.

[39] Salvador 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.

[40] N. R. Pal, J. C. Bezdek, and R. Hemasinha. "Uncertainty measures for evidential reasoning II: A new measure of total uncertainty". In: *Int. Journal of Approximate Reasoning* 8.1 (1993), pp. 1 –16.

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

[42] Leo Breiman. "Random Forests". In: *Int. Journal on Machine Learning* 45.1 (Oct. 2001), pp. 5–32.

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

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

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

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

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

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

[49] 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.

[50] R. Kohavi. "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection". In: *14^{th} Joint Conf. on Artificial Intelligence*. Montreal, Canada, 1995.

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

[52] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[53] A. N. Albatineh et al. "Correcting Jaccard and Other Similarity Indices for Chance Agreement in Cluster Analysis". In: *Adv. Data Anal. Classif.* 5.3 (Oct. 2011), pp. 179–200.

APPENDIX

A. ADVERSARIAL DISTRIBUTIONS

Fig. 8 presents examples of the adversarial strategies (Sec. 4) for the humidity and temperature measurements of the SC dataset. Recall that the underlying (normal) distributions for these phenomena are: $(\mu, \sigma) = \{(31, 5)|(21, 1.3)\}$, respectively. For the case of the humidity (upper part of Fig. 8), 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.

B. EVALUATION METRICS

In binary classification problems, classifiers label points as *positive* or *negative*. In our case, positive (p) refers to the inlier class (i.e, honest reports), whereas negative (n) to the

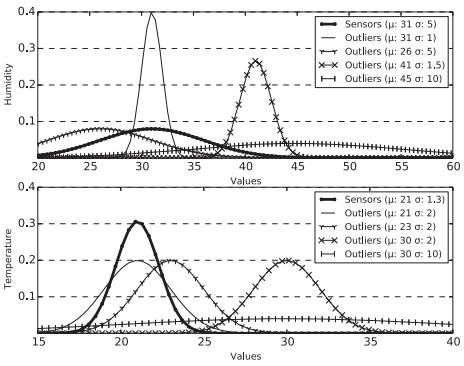


Figure 8: Adversarial data distributions and their similarity with the original SC humidity and temperature sensor datasets (depicted with thick lines).

outlier (and possibly malicious) class. $D = \{x\}_{i=1}^{m}$ is a dataset consisting of m reports divided into a training (TS) and an evaluation (ES) set. TS is given to the classifier for training, whereas ES is used to assess the its accuracy. Let $y_i = \{n, p\}$ be the *true* label of each report contained in the ES. Based on these true labels, ES is partitioned into two classes $\{T_n, T_p\}$. The classifier decides independently on the classification of ES and produces a partitioning, $C = \{C_n, C_p\}$, where each partition comprises the reports sharing the same classification label, $\hat{y}_i = \{n, p\}$. The accuracy of such a classification is assessed by examining the similarity of each partition, C_i, with the corresponding ground-truth, T_i, and measuring the degree of agreement between the actual and classified labels.

Precision and *recall* metrics [52] measure the correlation between TPs, TNs, FPs and FNs. Intuitively, precision examines the ability of a classifier not to label as positive (or negative) a sample that is negative (or positive), whereas recall measures the accuracy of a classification model in selecting all instances of a certain class. They are commonly expressed through the counts of TPs, TNs, FPs and FNs:

$$(Prec, Recall, F) = \begin{cases} Prec = \frac{TP}{TP+FP} \\ Recall = \frac{TP}{TP+FN} \\ F = 2 \cdot \frac{Prec \cdot Recall}{Prec + Recall} \end{cases}$$

where the F-measure is the harmonic mean of the precision and recall values for each class. Overall, higher values for these metrics indicate better classification.

Another important aspect that needs to be examined is the *similarity* between the classification and the ground-truth; the degree to which the two sets of actual and classified labels are related. In this context, the *Matthew's Correlation Coefficient* (MCC) reflects the correlation between these two sets and the *Jaccard* coefficient [53] measures the similarity as the intersection divided by the union of the sample sets. The formal definitions of these two metrics are,

$$(M, J) = \begin{cases} MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \\ Jaccard = \frac{|y_i \bigcap \hat{y}_i|}{|y_i \bigcup \hat{y}_i|} \end{cases}$$

MCC is a correlation coefficient ranging between $[-1 : 1]$. A value of 1 indicates perfect prediction, 0 an average random prediction (same as deciding based on a "coin toss") and -1 an inverse prediction. The Jaccard coefficient ranges between $[0 : 1]$. It is equal to the highest value when $\hat{y}_i = y_i$ and to the lowest when $\hat{y}_i$ and y_i are disjoint.