Detecting Malicious Data Injections in Event Detection Wireless Sensor Networks

Vittorio P. Illiano and Emil C. Lupu

***Abstract*—Wireless Sensor Networks (WSNs) are vulnerable and can be maliciously compromised, either physically or remotely, with potentially devastating effects. When sensor networks are used to detect the occurrence of events such as fires, intruders or heart-attacks, malicious data can be injected to create fake events and, thus, trigger an undesired response, or to mask the occurrence of actual events. We propose a novel algorithm to identify malicious data injections and build measurement estimates that are resistant to several compromised sensors even when they collude in the attack. We also propose a methodology to apply this algorithm in different application contexts and evaluate its results on three different datasets drawn from distinct WSN deployments. This leads us to identify different trade-offs in the design of such algorithms and how they are influenced by the application context.**

***Index Terms*—Security management, Ad-Hoc and sensor networks, Mining and statistical methods**

I. Introduction

WIRELESS SENSOR NETWORKS are a low-cost, flexible, and easy to deploy solution to monitor physical phenomena, but they are also particularly vulnerable to malicious attacks since deployments are often unattended; the sensors are physically accessible; the sensor hardware has limited computational resources; and the wireless interface is difficult to secure.

WSNs are often used to detect events occurring in the physical space across different applications such as military surveillance [1], health [2], and environment (e.g., volcano) monitoring [3]. Although these applications have different tasks, they all collect sensor measurements and interpret them to identify *events*, i.e. particular conditions of interest followed by a remedial response. Such response may have significant consequences and cost. Therefore, the measurements leading to the event detection, become a critical resource to secure.

When the measurements are somehow replaced or modified by an attacker, we deal with *malicious data injections*. The attacker may make use of the injected data to *elicit* an event response, such as evacuation in the case of fire, when no event has occurred, or *mask* the occurrence of a true event, such as the trigger for an intrusion alarm. Different means for obtaining control over the measurements are possible. Many of the studies in the literature address physical and network layer threats (e.g., [4]–[9]) by protecting the integrity of the measurements during their transmission (e.g., with a cryptographic hash function). However attacks may compromise the measurements even before they are transmitted. For example, an attacker may tamper with a sensor in the field and load a software that reports false measurements. Another possibility is that the attacker manipulates the environment by using for instance a lighter to trigger a fire alarm. In this paper we consider directly the scenario where an attacker gains full control of one or more sensors and can run arbitrary malware on them to fabricate new measurements and *report* them in place of the *observed* ones. Our task consists of detecting the incongruities between the observed and the reported measurements. Although, in an environment manipulation scenario there is no incongruity between observed and reported measurements, a reported measurement still differs from what would be reported in the absence of attacks. In both cases the non compromised measurement is an unknown variable, so it needs to be characterised through observable properties, which in turn are not compromised.

In the presence of malicious data injections, there are few observable properties that can help detection. One of them is the loss of integrity of the sensor e.g., detect that it is running malicious software. For such a scenario, *software attestation* techniques have been proposed [10]–[13], but require further evaluation in concrete deployments i.e., outside the controlled environment of a research lab. Note, however, that injections through environment manipulation cannot be detected through attestation since the software is still genuine.

Beyond the integrity of the software, the measurements reported by the sensors are themselves another observable affected by malicious data injections. The approach pursued in this paper is based on measurements analysis and its applicability relies on the assumption that the measurements are correlated under genuine circumstances, while compromised measurements disrupt such correlations. The presence of correlations indicates that the measurements contain redundant information. Such redundancy is necessary, since without it, there are no terms of comparison to detect anomalies other than broad constraints such as the admitted measurements range. Correlation instead adds more constraints to the measurements, and allows less freedom for an attacker. Besides correlation, detection of inconsistencies in the measurements requires the presence of a non void subset of genuine sensors in the network. This is typically the case since compromising a sensor generally entails a cost for the attacker, as well as a risk of being detected. However, if the subset of genuine sensors is too small with respect to the measurements correlation, the attack may be undetectable or, even if detectable, there may be insufficient information to determine which sensors are compromised and which are genuine.

To detect malicious data injections, we propose an algorithm that characterises the relationships between the sensors' reported values arising from the *spatial correlations* present in the physical phenomenon. Even though correlation-based analyses may easily spot a single malicious measurement, the problem becomes more difficult in the presence of multiple malicious measurements, originating from colluding sensors. We define *collusion* as the process of injecting malicious measurements through multiple sensors in a coordinated fashion. We do not make assumptions about how colluding sensors manifest but we highlight that collusion may enable eliciting/masking an event whilst remaining undetected and/or lead to genuine sensors being considered as compromised. The potential effects of collusion increase with the number of sensors involved, until the attacker has enough power to conduct any attack undetected. To ensure the detection is resistant to collusion, we introduce novel ways of aggregating measurements that are aimed at discarding malicious contributions under attack and minimise the false positives under genuine circumstances as well. We show that even though our approach is based on simple and solid techniques (e.g., linear regression, weighted median, Pearson correlation) it can detect sophisticated collusion attacks that have not been considered before. Indeed, we consider an attacker that selects the sensors to compromise and injects real measurements, which are chosen to maximise the correlation among compromised sensors and minimally disrupt correlations with genuine sensors. Furthermore, the low computational complexity of such techniques makes the overall approach suitable for extensive analysis of online data.

We also propose a novel more general methodology to apply our algorithm in different application settings. In particular, we show how the algorithm parameters can be derived from tests and knowledge that are pertinent to the deployment and application of the WSN, such as the event detection criterion used. Indeed, many prior studies (e.g., [14]–[17]), propose algorithms that are evaluated in a single deployment or application setting, or even on a single simulated dataset. Would the same algorithm work in a different setting? For many reasons, this is not likely. In contrast, we show our work to be applicable across three different application domains: health-care monitoring, monitoring of volcanic activity, and home fire alarms, each with different challenges. We build realistic attacks that undermine the core objective of each application and minimise the chance of detection.

The remainder of this paper is organised as follows: We review the related work in Section II and define the scope of our work accordingly. We characterise the challenges originating from collusion attacks and introduce a detection algorithm in Section III. In Sections IV, V, and VI we describe the three phases of our algorithm: *Estimation*, *similarity check* and *characterisation*, each time identifying the aspects that need to be customised to the application setting. Section VII discusses the computational complexity of our approach. We show how our methodology is tailored to three different application settings in Section VIII. Finally, our conclusions and directions for future work are presented in Section IX.

II. Related Work

Our proposed solution improves upon state-of-the-art approaches and provides:

1) A method to run the detection on broad neighbourhoods where the measurements of two neighbours can significantly differ (but are still correlated).
2) An extension of voting-based and trust-based frameworks to *estimation*-based frameworks.
3) A general methodology to flexibly tailor the technique to WSN applications that detect different kinds of events.

Contribution 1 consists of extending the assumption made in [14]–[20] where the value of the sensed phenomenon is required to be the same within a neighbourhood and measurements differ only because of noise. This assumption allows to easily estimate the ground truth and label the measurements as outlying through e.g., one-class quarter sphere support vector machines (SVM) [20]. However, this assumption is generally valid only for very small neighbourhoods, where collusion attacks can be successful by compromising all the sensors. To resist collusion, it is necessary to broaden neighbourhoods and cope with measurements that are significantly different. But in the absence of a common ground truth, what should a sensor's reported measurement be compared against? Previous work, such as [21]–[23], have proposed to detect inconsistencies in the correlation within a neighbourhood by extracting a unique overall consistency metric, to which every neighbour contributes. This, however, allows colluding sensors to compensate for each other and reduce the overall inconsistency, whilst still disrupting the reported values [24].

We observe that each sensor can exploit correlations to produce an *estimate* for the measurements of other sensors. The estimates can then be aggregated with a collusion-resistant operator that produces a final reliable estimate to be compared with the reported measurement. An approach similarly based on aggregation of individual sensors' information is majority voting [15], [17], [25], [26], where each sensor votes for a neighbour's maliciousness and the votes are aggregated by majority. Similarly, trust-management frameworks aggregate individual beliefs about a sensor's behaviour [18], [19], [27], [28]. A sensor's behaviour is mapped to a trust value by all its neighbours, and then the sensor's trustworthiness is obtained e.g., by averaging the trust values [28]. However, the main drawback of these techniques is that they introduce an additional variable - the vote, or trust value - about which an attacker can lie with or without lying about the measurements at the same time. Detecting such attacks incurs additional computation and communication costs. In contrast, our contribution 2 extends voting-based and trust-based frameworks by *aggregating measurements estimates rather than votes or trust values*. Such choice does not introduce additional variables, since the estimates are directly calculated from the raw measurements.

We show the limitations that estimation-based frameworks overcome by analysing possible voting scenarios among the sensors in Fig.1. Consider at first nodes A, B, and C to be compromised. In this case A is free to inject arbitrary malicious data if B and C collude to not report on it and act genuinely to avoid reports from D, E, F. If estimates were available, we would notice that the measurements of B and C are consistent with those of D, E, F (the reason why they don't report), but are not consistent with those of A. Alternatively,

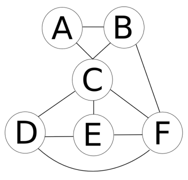


Fig. 1: Example WSN topology. Nodes represent sensors and edges indicate a neighbour relationship.

consider nodes D, E, F to be compromised. Here nodes D and E can inject any kind of measurements, although C may report on them. Indeed, node F can avoid reporting on them and report on C instead. Then, with a simple majority voting approach [26], node C would appear as the compromised node. Through the estimates instead, we can detect that there is no valid reason for sensor F to endorse D and E and report on C, uncovering the collusion attempt. A majority voting approach will always fail when more than 50% of sensors are compromised. However, such upper bound considers the best case scenario, where all the genuine nodes report correct votes without uncertainty. Our estimation-based framework instead, considers the *degree of uncertainty*, which becomes the only bounding factor. Indeed, our experiments in Sect. VIII show tolerance against up to 88% compromised nodes.

To the best of our knowledge, no previous papers develop a general methodology showing how the algorithms can be systematically tailored to different deployments and different applications. Our contribution 3 deals with this aspect. We also describe sophisticated collusion attacks that are application-agnostic and therefore can be used as a generic testing criterion for assessing the robustness of different algorithms.

III. Overall approach

The *estimation*-based framework, which iteratively extracts and aggregates measurements estimates, is at the core of our detection mechanism. Estimates are iteratively computed on new measurements and a *similarity check* compares them as shown in Fig. 2. When the similarity check fails, we run a *characterisation* step – an extensive analysis that identifies the likely compromised sensors. The models used to build the estimates are learnt during an *initialisation*, which serves to customise our techniques to the specific WSN deployment and whose output is the *estimation models*.

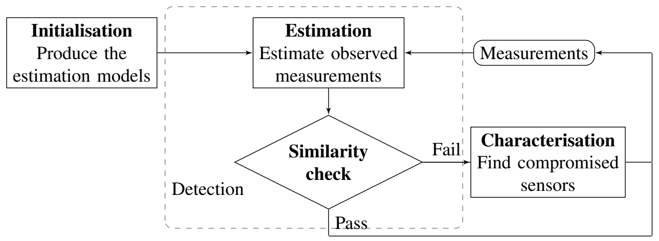
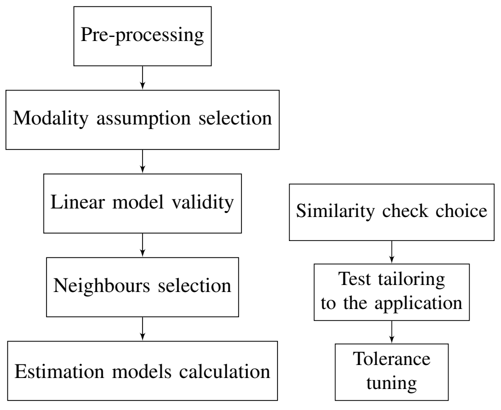


Fig. 2: Outline of the algorithm

During the *initialisation*, the network is assumed to be free of compromise. The steps leading to the estimation models are shown in Fig. 3a: *a*) we first pre-process the data to eliminate faulty readings, alleviate noise and transform the data to extract the parameters of interest, *b*) we check if the estimation models change significantly in the presence of events, and if so we create a separate set of estimation models for each *modality* of the physical phenomenon, *c*) we analyse the data to test if the correlation detected allows to build a linear model to perform the estimation, *d*) the validity of a linear model is defined for each pair of sensors and allows to identify the neighbourhood of a sensor as the set of sensors with which there is a strong linear relationship, *e*) finally the parameters that fit the estimation models are calculated.

The *similarity check* also needs to be customised to the sensor deployment during the initialisation, according to the steps shown in Fig. 3b. To assess if the estimates are similar enough to the the reported measurements, a similarity metric is required. Since the final goal of our detection scheme is securing the event detection, the similarity metric should check the integrity of observable properties that characterise the event. Such properties can be derived from the *event detection criterion*, i.e. the algorithm that is run on the measurements to detect the presence of events. We define two main properties that characterise most event detection criteria: the *magnitude* and the *shape* of the measurements signal, based on which, we build respective tests. Customisation involves: *a*) choosing a similarity check based on such tests, *b*) tailoring the test to the application according to the event detection criterion, *c*) tuning a *dissimilarity tolerance* parameter to reach an objective false positive rate (FPR).



(a) Estimation models (b) Similarity check

Fig. 3: Methodology scheme

The tasks pertinent to each step can be assigned to different devices. However, in estimation-based frameworks, as in voting- or trust-based frameworks, the final detection decision should be taken by an entity that is not compromised, typically the base station. The decision is taken based on the values (votes, trust, measurements) received from the sensors. In some cases, the deciding entity cannot perform such aggregation alone e.g., when measurements are aggregated while being collected in the network. In such cases the base station can delegate the detection tasks to the aggregating nodes but must also, additionally, verify the validity of the aggregation. Techniques to detect injections in aggregated reports have been proposed in [19], [29], [30] among others and are complementary to the work we present here, which focuses on the analysis of the raw sensor data. In the next section we describe in detail the steps in Fig. 2, their rationale and customisation.

IV. Estimation

We model the sensing problem in WSNs considering a generic sensors deployment S, with $|S| = N$. We refer to the measurement *observed* by sensor i ($i \in S$) at time t as $O_i(t)$, while the measurement *reported* to the base station (or data sink), is denoted by $S_i(t)$. We calculate $\hat{O}_i(t)$, an estimate of the observed measurement, from the measurements given by i's neighbours $S_{j \in N(i)}(t)$. For the moment we will consider all the other sensors as *neighbours* but will review the selection of neighbours in Sect. IV-A2. This aggregated estimate allows us to detect if the sensor has reported a measurement that differs significantly from it. Note that compromised neighbours could collude, as mentioned in Sect. II, to bias the estimate and make it more consistent with the reported measurement. To avoid this problem, we obtain separate (pairwise) estimates from each neighbour. In a second step, the estimates are aggregated by an operator that is resistant to compromised estimates. We refer to the first part of this approach as *pairwise estimation*, and to the second part as *estimate aggregation*.

A. Pairwise Estimation

The measurements of two sensors are related, and in particular spatially correlated, because the sensed physical phenomena affect and propagate across the environment in which the sensors are placed. Ideally, the relationship could be characterised in a mathematically precise way, given by the laws of the physical phenomenon and its propagation. In reality, environmental changes, noise, interactions with other phenomena etc., constrain us to work with approximations and more specifically inferred correlations that can be established with a certain margin of error.

Though we are exploiting spatial correlations, in event detection WSNs, we also need to account for a temporal parameter, which is the propagation delay of the event. By modelling the stream of measurements from sensors i and j as the random variables $O_i(l)$ and $O_j(m)$ respectively, we define the inter-sensor delay δ_{ji} as the time it takes for the event to propagate from i's to j's position. We estimate this quantity as the value that maximises the *Pearson correlation coefficient* and use it to align the measurements in time. We assume that δ_{ji} is small enough to perform the alignment within the timeliness requirements of the event detection. Note that the calculation of the inter-sensor delay absorbs any synchronisation errors, so the method does not require clock synchronisation. δ_{ji} can change with the value of the measurements, so ideally, would need to be estimated for each new measurement. This choice is cumbersome (calculation of the Pearson correlation is expensive), and an attacker can also try to subvert this calculation. Thus, we calculate it only once for each new estimation model, by optimising the average Pearson correlation coefficient. From now on, we use the notations O_i and O_j to refer to two samples of the random variables aligned by the calculated inter-sensor delay, i.e. $O_i(l)$ and $O_j(l + \delta_{ji})$.

At this point, we calculate $\hat{O}_{ij}$: an estimate of O_i based on O_j. To characterise this quantity, we assume that O_i can be approximated with a linear combination of O_j plus a residual noise random variable ϵ_{ij}. We assume that noise is normally distributed, after removing obvious outliers; when this condition cannot be satisfied, robust regression should be preferred [35]. A linear relationship makes our estimation lightweight both in time and space complexity (see Sect. VII). Its validity can also be tested statistically, and the neighbourhood of a sensor can be defined as the set of sensors for which this relationship holds. When this linear relationship holds less, the estimate is weighted to have a smaller influence on the final result (as shown in Sect. IV-B). We refer to the equation that approximates $\hat{O}_{ij}$ with a function of O_j as the *estimation model* described below.

Each estimation model can be extracted through *linear regression*, which calculates the coefficients a_{ij} and b_{ij} in:

$$\hat{O}_{ij} = a_{ij} O_j + b_{ij} \tag{1}$$

Thus, an estimation model is defined by the pair (a_{ij}, b_{ij}) calculated as shown in Algorithm 1. In the absence of an a-priori characterisation of random variables O_i and O_j, we use their sample mean, sample variance and sample covariance, respectively, estimated on real data.

Algorithm 1 Estimation models calculation

```
Input: $O_i$ $i \in S$
Output: $(a_{ij}, b_{ij}) \forall i \neq j$
1: {Initialisation: align the measurements with the inter-sensor delay}
2: for all $i \in S$ do
3:     for all $j \in N(i)$ do {$N(i)$ indicates $i$'s neighbours}
4:         $a_{ij} = \frac{cov(O_i, O_j)}{var(O_i)}$
5:         $b_{ij} = E[O_i] - a_{ij} E[O_j]$
6:         store $(a_{ij}, b_{ij})$
7:     end for
8: end for
```

Regression schemes for anomaly detection in WSN have been previously proposed in [23], [31], [32]. However, [31] and [32] apply regression in the temporal rather than the spatial domain i.e., they estimate correlations between measurements of a single sensor. In [23] the spatial domain is also considered with multiple linear regression, however this technique is sensitive to outliers, especially when they are correlated because of collusion. In contrast we focus on spatial correlations because an attacker can always introduce measurement deviations slow-enough to make them appear consistent. In the spatial domain the presence of genuine sensors makes such deviations detectable.

1) The Validity of the Estimation Models: Estimating the models between every pair of sensors at run-time is computationally expensive. Moreover, the measurements used to learn the models must be genuine, i.e., from non-compromised sensors. The most secure and less expensive option is to learn the models one-off with many historical samples. This choice also allows to use more powerful devices and to extensively analyse the data set to ascertain its integrity.

However, the estimation models are valid only when the linear relationships remain stable through time. This may not be true, especially when the measurements' accuracy is subject to degradation[1]. If degradation is due to faults and a fault-detection module is present, the faulty sensors can be excluded. Other changes in the models' accuracy can be handled by online model updates, that add information extracted from newly collected measurements. Design of an update criterion is beyond the scope of this paper, but we emphasise that such updates require good confidence in the integrity of the new data, otherwise an attacker can "poison" the model e.g., to make anomalous data become normal.

Another important factor to consider for the validity of the estimation models is that an event may induce different relationships between sensors than those present in non-event conditions. This is confirmed by the data sets we have examined. We must therefore distinguish between different *modalities* in which the network operates, each corresponding to a set of estimation models. We consider the following three *modality assumptions*:

1) *Unique modality*. The same relationships between sensors hold in event or non-event conditions.
2) *One modality in event conditions, a different one at rest*. This occurs when events induce a different spatial pattern, but all events have similar patterns.
3) *Infinite or unknown number of modalities*. In the most general case.

In this paper we cater for assumption 1) and also for 2) since it is a straightforward extension of 1). Two different sets of estimation models are computed: one in event conditions, the other in non-event conditions. The estimation models learnt in event condition apply when the detection criterion is satisfied while those learnt in restful conditions are used when no event is detected. Considering different modalities complicates detection, so different modalities should be used only when different behaviours in the relationships between sensors can be identified (e.g., through a scatter plot).

A higher number of modalities can similarly be catered for. However, we leave the more complex assumption 3) for future work. In this case the estimation models could be fitted to the measurements rather than learnt in advance. But the measurements may themselves be compromised.

2) The Accuracy of the Estimation Models: The accuracy of the estimates is influenced by several factors including the accuracy of the linear relationships, noise, faults, and the informative content of the measurements. Noise is a common source of inaccuracy, especially for sensors in harsh environments. Generally, noise signals can be removed or reduced with standard filters. Another common cause for wrong estimates is the presence of faults. Our algorithm alone cannot discern if the wrong estimate was intentional (malicious) or unintentional (faulty), but both will be detected to enable a proper reaction. However we point out that our solution is not designed to detect sensor faults, and a dedicated fault-detection module should run in parallel for this task.

[1]or when sensors are mobile but we focus here on infrastructure sensing.

To increase accuracy, the analysis must be applied to the data containing the information of interest, which may not be the raw measurements but information derived from them. For example, the heart voltage measured by electrocardiography is less informative than its frequency. Another example, regarding the use of infrasound sensors for monitoring volcanic eruptions, is shown in Section VIII-C. In such cases pre-processing the raw signals improves the accuracy significantly.

The accuracy of the estimation models also depends on the presence of a linear relationship between the sensors measurement. Previous work (e.g., [14]–[17], [19]) has considered a stronger assumption that does not allow spatial variations between the sensors, i.e. the measurements are assumed to be so close that their expected value is the same. We have instead given broader applicability to our technique by assuming a linear relationship between the measurements. For some WSN applications such as target tracking, linear relationships can be seen only at very close points, and if the WSN deployment is not dense enough there may be no pair of sensors for which such relationship holds. Therefore, the validity of the linearity assumption needs to be checked beforehand, and we do so with the *squared Pearson correlation*, which is a *goodness of fit* metric for linear regression. In some cases, our approach can also be applied when linear relationships cannot be found by applying a non-linear transformation to achieve linearity [33]. After the transformation, an increasing Pearson correlation coefficient indicates whether the transformation is convenient.

The goodness of fit allows us also to measure the goodness of the neighbours contribution and weigh it accordingly. Thus, even when considering all the other sensors as neighbours, uncorrelated neighbours will not degrade the result. However, the complexity of the detection algorithm increases with the neighbourhood size (see Section VII). Then, a network-wide neighbourhood is realistic only in small scale deployments while in large deployments, the neighbourhoods should be restricted. Our principle for selecting the best neighbourhoods is selecting the first sensors in a list sorted by descending goodness of fit: this choice is more robust than distance based criteria since, e.g., in the presence of obstacles, two sensors may be very close but show poor correlation.

Restricting the neighbourhoods does not ensure that all the neighbours are equally correlated, hence the goodness of fit should still be used to weigh the neighbours' contributions. We therefore introduce the *prior weight* (w_{ij}^{-}), which is the goodness of fit normalised across a neighbourhood and captures the relative a-priori confidence in the pairwise estimation model between a sensor and one of its neighbours. By weighting the neighbours appropriately, the accuracy increases with the neighbourhood size.

B. Estimate Aggregation

For every new measurement collected by a sensor, multiple pairwise estimates are calculated through the estimation models. At this point we aggregate them into a final estimate $\hat{O}_i$ that approximates O_i and allows us to detect the presence of malicious data injections. To achieve this, $\hat{O}_i$ must aggregate estimates in a way that is both accurate and minimally corrupted by malicious estimates. In particular, the second requirement demands us to not trust the relationships between different estimates. Indeed, different estimates for the same measurements share some mutual information, or in other words the information brought in by an estimate is reduced by knowledge of other estimates. Nevertheless, such property holds only in the absence of malicious interference. With respect to malicious data injections instead, even two estimates that are expected to be perfectly correlated bring in independent information, since we assume independent probabilities of compromise for different nodes. For this reason, our weighting scheme does not consider inter-estimate correlation.

Two candidates to aggregate pairwise estimates are *weighted mean* and the *weighted median*: both take as input a set of estimates and their prior weights and return an aggregated value. The *weighted mean* can achieve a smaller error than those of the single estimates. However, it is highly sensitive to compromise, since the final result is proportional to the input values: even one compromised (outlier) estimate can introduce an arbitrary deviation in the result. In contrast, the *weighted median* [34] is more resistant to compromise. It first sorts the values ascendingly, then arranges the weights with the same order, transforms them into substrings with a length proportional to the weight and picks the element at the half-length of the resulting string. Its drawback is that by picking one among all estimates, the error cannot be reduced further.

Since there is a trade-off between accuracy and compromise resistance, we propose to combine the two operators with the following heuristic: first, the weighted median operator is applied; then the weighted mean is calculated with new weights, the *posterior weights* (w^+_{ij}), obtained as the prior weights times a function which penalises values distant from the result of the first step. Such function is the complementary cumulative distribution function of the estimation error, where the latter is calculated as the difference between the pairwise estimates and the result of the weighted median.

$$\begin{aligned} p_{ij}(\hat{O}', \hat{O}_{ij}) &= P(|\epsilon_{ij}| > |\hat{O}'_i - \hat{O}_{ij}|) \\ &= 1 - erf(\tfrac{|\hat{O}'_i - \hat{O}_{ij}|}{\sqrt{2} std(\epsilon_{ij})}) \end{aligned} \tag{2}$$

Where erf is the error function and $std(\epsilon_{ij})$ is the residual standard deviation [35], calculated together with the respective estimation model. The overall procedure is detailed in Algorithm 2 below, where $\boldsymbol{\hat{O}_{iN(i)}}$ are the estimates for i's observed measurement from its neighbours and $\boldsymbol{w^-_{iN(i)}}$ are their respective prior weights.

Our novel algorithm gives a collusion-resistant and accurate aggregation. It differs from robust aggregators, such as the Orthogonalized Gnanadesikan-Kettenring (OGK) operator [15], since it takes also the prior weights in input to cater for the general case where the values are not equally pertinent to the aggregate. As a consequence, the accuracy of our aggregate always increases when adding more values if the prior weights are correct, while the accuracy of OGK decreases with the introduction of less accurate values. Moreover, our operator exploits the residual standard deviation $std(\epsilon_{ij})$ to quantify how much a weight should be penalised; OGK instead penalises all values equally, without considering their specific uncertainty. The penalty applied by OGK depends on the data variability measured with the median absolute deviation. This is disadvantageous for collusion resistance, as an attacker may seek to increase the estimates' variability to be penalised less.

Algorithm 2 Calculation of the aggregated estimation

```
Input: $\boldsymbol{w^-_{iN(i)}}$, $\boldsymbol{\hat{O}_{iN(i)}}$
Output: $\hat{O}_i$
1: $\hat{O}'_i$ = weightedMedian($\boldsymbol{w^-_{iN(i)}}, \boldsymbol{\hat{O}_{iN(i)}}$)
2: for all $j \in N(i)$ do {Calculate the posterior weights}
3:     $w^+_{ij} = w^-_{ij} \cdot p_{ij}(\hat{O}', \hat{O}_{ij})$
4:     $\boldsymbol{w^+_{iN(i)}}$.append($w^+_{ij}$)
5: end for
6: $\boldsymbol{w^+_{iN(i)}} = \frac{\boldsymbol{w^+_{iN(i)}}}{\sum_{j \in N(i)} w^+_{ij}}$
7: $\hat{O}_i$ = weightedMean($\boldsymbol{w^+_{iN(i)}}, \boldsymbol{\hat{O}_{iN(i)}}$)
8: return $\hat{O}_i$
```

To control the aggregation result, an attacker must ensure that the weighted median is one of the compromised estimates and thus that the sum of the weights of compromised estimates is > 0.5. This condition enables non-detectable injections into a single sensor but is not sufficient to keep the attack undetected. The attacker also needs to control the estimations for the other compromised sensors. The total number of sensors needed to keep all compromised sensors undetected depends on the strength of the pairwise correlations. Instead, the number of sensor needed to mask or elicit an event depends on the event detection criterion. Our empirical evaluations show that although a few sensors are generally required to subvert the event detection, a substantial additional number of sensors is required to avoid detection.

V. Similarity Check

From the estimate aggregation step, each reported measurement S_i has an estimate $\hat{O}_i$ of the observed value. To detect data injections in S_i, we compare the two using a *similarity metric* that must be consistent with the event detection criterion. So, two signals that are similar according to the metric must also have similar effects on the event detection and vice-versa. Here we propose two tests that capture the characteristics of most event detection criteria. The *magnitude test* verifies that reported measurements are close in magnitude to their estimates. The *shape test* verifies that the estimate and reported signal have a similar shape. The choice of the most appropriate test, or a combination of the two should be made at design time based on the event detection criterion.

A. Similarity Test 1: Magnitude

In some WSNs, events are triggered when measurements are higher or lower than a reference value. For example, fire alarms trigger when the temperature is above a threshold. An attacker must therefore inject measurements, which differ in magnitude with the observed ones. In such cases we use $M_i =$

$(\hat{O}_i - S_i)$ – the difference between the reported measurement and its estimate – to build a *magnitude test*, which checks that the difference is small enough.

We assumed that the regression *residual*, i.e. the error between a value and its estimate, is zero-mean and normally distributed. Even if $\hat{O}_i$ is the result of the aggregation described in Section IV-B, the error $\epsilon_i = (\hat{O}_i - O_i)$ can still be assumed to be normally distributed. Indeed, our aggregate is a weighted mean of pairwise estimates, so it equals the true value plus the weighted mean of the pairwise residuals as shown below, where ϵ_{ij} denotes the residual in the regression of sensor i's measurement based on sensor j's.

$$\begin{aligned}\hat{O}_i = \textstyle\sum_{j\in N(i)} w^+_{ij}\hat{O}_{ij} = \sum_{j\in N(i)} w^+_{ij}(O_i + \epsilon_{ij}) \\ = O_i + \textstyle\sum_{j\in N(i)} w^+_{ij}\epsilon_{ij}\end{aligned} \tag{3}$$

Assuming that neighbours have independent residuals (e.g., because of independent noise), ϵ_i is a linear combination of independent normally distributed samples, and is thus normally distributed too [36]. Its mean is still zero, and its variance is:

$$var(\epsilon_i) = \sum_{j\in N(i)} {w^+_{ij}}^2 var(\epsilon_{ij}) \tag{4}$$

This equation has an important characteristic: the variance of the estimate is a combination of the variances given by each neighbour. Therefore, if a sensor joins or leaves the network, it is sufficient that all its new/old neighbours recompute the variance instead of learning a new one. Since $\frac{\epsilon_i}{std(\epsilon_i)}$ follows the standard normal distribution, also $\widetilde{M}_i = \frac{M_i}{std(\epsilon_i)}$ does when the measurements are genuine. We refer to $\widetilde{M}_i$ as the *magnitude deviation*.

Related studies (e.g., [14], [37]) have defined the normal samples with a confidence interval, characterised by the condition $|\widetilde{M}_i| < \theta$. The threshold θ determines the trade-off between false positives and false negatives, and has been usually set to $\theta = 3$: in this case only 0.3% samples are expected to be beyond the interval. Though 0.3% may seem a small percentage, it needs to be compared with the measurements' sampling period. For instance, with a sampling period of 1 second, each genuine sensor will generate a false positive about every 5.5 minutes. This may trigger the shutdown of the sensor, reconfiguration and/or restart, which can be so expensive that the cost of detection may be prohibitive.

Increasing the threshold reduces false positives, but decreases the detection rate. However, in event detection WSNs the false positives can be partly reduced without losing the detection rate by elaborating magnitude deviations in the same way as the event detection criterion elaborates the measurements. Specifically, if events are detected by applying a function to the measurements in a W_E-long time window, we can apply the same function to the magnitude deviations in the same time window. Consecutive magnitude deviations are unlikely to cause genuine anomalies with a long duration, unless there is a permanent fault that the fault-detection module should detect. Anomalies due to compromise, instead, have a longer duration as the attacker aims to subvert the event detection result. The final step consists of comparing the elaborated magnitude deviation to the *threshold* T_M. In our experiments, we ran the algorithm on genuine historical data with different values of T_M and selected the lowest value of T_M that achieves a reasonable false-alarms frequency (calculated as sampling frequency over FPR) for that application.

B. Similarity Test 2: Shape

Some event detection algorithms trigger based on changes in the time evolution of measurements such as changes in trend or of frequency. These are characteristics of the shape of the signal rather than its magnitude.

A metric that measures similarity in the shapes of two signals is the Pearson correlation coefficient. Since our purpose is to check the shape of the measurements used for event detection, we calculate this coefficient within a moving time window of size W_E: the event detection time window. Calculating Pearson correlation for all sensor pairs in a neighbourhood would have a computational complexity of $O(N_N^2 W_E)$, with N_N being the neighbourhood size. In contrast, we evaluate the Pearson correlation coefficient of a sensor's measurements with its estimates, achieving a complexity of $O(N_N^2 + W_E N_N)$. Indeed, we compute the coefficient $R_{S_i,\hat{O}_i}$, between W_E consecutive values of S_i and $\hat{O}_i$, and compare it against the distribution of $R_{O_i,\hat{O}_i}$. Specifically, if the coefficient is below the median[2], we check if at least $100 - C_R\%$ samples are expected to be so low by testing $\widetilde{R}_{O_i,\hat{O}_i} = \frac{R_{O_i,\hat{O}_i} - MED(R_{O_i,\hat{O}_i})}{D_{R_i}} > 1$, where D_{R_i} is the C_R-th percentile of $R_{O_i,\hat{O}_i}$.

To eliminate the need for the distribution of $R_{O_i,\hat{O}_i}$, the quantities $MED(R_{O_i,\hat{O}_i})$ and D_{R_i} are approximated with $\widehat{MED(R_{O_i,\hat{O}_i})}$ and $\hat{D}_{R_i}$ respectively. These are calculated with a heuristic described in Algorithm 3 for a generic sensor i. We find the best neighbour j^*, for which the median Pearson correlation coefficient is maximum. Then we approximate $MED(R_{O_i,\hat{O}_i})$ with its median and D_{R_i} with its respective distance to the C_R-th percentile.

Algorithm 3 Characterisation of the distribution of R_{S_i,O_i}

```
Input: $R_{ij:j\in N(i)}(r)$, $C_R$
Output: $\widehat{MED(R_{O_i,\hat{O}_i})}$, $\hat{D}_{R_i}$
 1: for all $j \in N(i)$ do
 2:     $MED_{R_{ij}} = MED(R_{ij}(r))$
 3:     $\boldsymbol{MED_{R_{ij}}}$.append($MED_{R_{ij}}$)
 4:     $r^{LOW} = \{r : r < MED_{R_{ij}}\}$
 5:     $D_{R_{ij}} =$ percentile($MED_{R_{ij}} - R_{ij}(r^{LOW}), C_R$)
 6:     $\boldsymbol{D_{R_{ij}}}$.append($D_{R_{ij}}$)
 7: end for
 8: $j^* = \text{argmax}_{j\in N(i)}(\boldsymbol{MED_{R_{ij}}})$
 9: $\widehat{MED(R_{O_i,\hat{O}_i})} = \boldsymbol{MED_{R_{ij}}}[j^*]$
10: $\hat{D}_{R_i} = \boldsymbol{D_{R_{ij}}}[j^*]$
11: return $(\widehat{MED(R_{O_i,\hat{O}_i})}, \hat{D}_{R_i})$
```

[2]We characterise the samples below the median since the injected measurements are supposed to have a low correlation with the real values.

In the absence of the distributions $R_{ij \in N(i)}(r)$, we estimate $MED_{R_{ij}}$ and $D_{R_{ij}}$ on historical data. Note that we could estimate $MED(R_{O_i,\hat{O}_i})$ and D_{R_i} empirically, by running the whole algorithm on an ad-hoc dataset. However, the latter must be disjoint from the data used to learn the estimation models, otherwise the results could be biased by overfitting [38]. Moreover, the parameters would only be valid for the configuration of sensors in the ad-hoc dataset. Instead, the heuristic removes the need for an ad-hoc dataset and automatically adapts when sensors join or leave the network.

For genuine sensors $S_i = O_i$, then $\widetilde{R}_{S_i,\hat{O}_i} \leq 1$ for $C_R\%$ genuine samples. We thus define $\widetilde{R}_{S_i,\hat{O}_i}$ as the *shape deviation* and calculate C_R as the lowest value that achieves a reasonable false-alarms frequency. The false positives due to short term anomalies can be reduced in a similar way to that used in the magnitude test i.e., by computing the median of W_{Sm} consecutive correlation coefficients calculated on overlapping time windows. W_{Sm} should never exceed W_E, otherwise the information from disjoint time windows would be merged.

VI. Characterisation

When the similarity check fails for a sensor, the sensor may have been compromised by malicious data. However, in some cases the similarity check could also fail on genuine sensors, because the wrong modality was chosen (e.g., a non-event modality rather than an event modality) or because the estimation was disturbed by compromised sensors.

The latter occurs when several nearby sensors collude in providing malicious estimates. However, to bias the estimates for genuine sensors by a certain quantity and increase their deviation, compromised nodes typically need to inject measurements that have even larger deviations (if they do not need this, colluding sensors have probably enough influence over the measurements to remain undetected). Therefore, our characterisation step consists in removing the sensors with the highest deviation, one by one, and recomputing the similarity check on the remaining sensors in the neighbourhood. Each time we remove a sensor, which we presume compromised, the genuine sensors gain in consistency with their estimate whereas colluding sensors lose the benefits of the removed sensor's estimates. The procedure stops when all the remaining sensors pass the similarity check. The overall characterisation algorithm is shown in Algorithm 4, where SCheck is the similarity check and D_i is the generic deviation (coming from the magnitude/shape tests) calculated for the similarity check.

Another factor to consider when the similarity check fails is the *modality assumption* (Sect. IV-A1). When different modalities are used in event conditions and non-event conditions, there is some uncertainty about which modality to use because malicious data may have compromised the event detection output. In this case, the wrong estimation model may be used and genuine sensors may fail it. Our solution is to run Algorithm 4 in both modalities when the similarity check fails and then choose the modality in which the smallest compromised set is returned. It is reasonable to choose the correct modality based on a majority approach, as the attack costs increase with the number of measurements that need to be controlled. Note that this is different from event detection with majority voting, since the measurements are not required to trigger in majority, but to reflect event propagation and show graceful transitions in their measurements.

Algorithm 4 Characterisation algorithm

```
Input: $D_i \ \forall i \in S$
Output: compromisedSet
 1: compromisedSet={}
 2: residualSet=$S$
 3: while OR$_{i \in \text{residualSet}}$(SCheck($D_i$)$fails$) do
 4:     $s^* = argmax_{i \in residualSet} D_i$
 5:     compromisedSet.append($s^*$)
 6:     residualSet.remove($s^*$)
 7:     for all $j \in S : s^* \in N(j)$ do
 8:         $N(j) = N(j) \setminus s^*$
 9:         recompute $D_j$
10:     end for
11: end while
12: return compromisedSet
```

VII. Complexity analysis

Our approach relies on computations on raw measurements, which are carried out by a trustworthy node. When the base station collects all the measurements, it is a good candidate for this role as it has a global view, which is useful to deal with collusion of compromised nodes [25]: we refer to this scenario as *centralised*. This is also the case when the measurements are collected through intermediate *aggregators* but the aggregation is lossless. When the intermediate aggregators perform a lossy aggregation, the base station can instead delegate the detection task to the aggregators rather than forcing delivery of all raw data. In this case, which we refer to as *distributed*, the base station must also assess the integrity of the processing at the aggregators. The centralised and distributed solutions have different computational and communication overheads, which we analyse below.

For the WSN nodes that do not act as aggregators, the estimation-based framework adds no overhead, because no additional software is run on the sensor nodes to manage votes or trust values like in [19]. For the base station and aggregators, the most computationally expensive operation of our approach is the calculation of the estimation models. When this operation is done one-off, powerful devices may be used offline, but when this is not possible, for instance because there is not enough historical data, the models need to be estimated in real time. In this case using external devices may be infeasible, and an efficient calculation is required to estimate the models with the sensor nodes. We deal with this problem with incremental regression, which updates the model through D new samples with a complexity $O(D)$ [39]. The overall complexity in the distributed case is thus $O(DN_AN_N)$ for each aggregator, where N_A is the number of sensors managed by the aggregator and N_N is the average number of neighbours used for the estimations. Similarly, in the centralised approach the complexity is $O(DNN_N)$. This complexity is generally low for a base station, but may be high for an aggregator: in this case random sampling and neighbourhood size reduction reduce D and N_N respectively. Note that new sensors joining the network require the computation of further regression problems, $2N_N$ on average, so the total overhead for a new node joining is $O(DN_N)$.

Besides the calculation of the the estimation models, the estimations calculation and the similarity check have a computational overhead. Each aggregated estimation requires N_N multiplications and sums, while the similarity check has a complexity $O(W_E)$ (for the shape test it is the complexity of Pearson correlation and for the magnitude test it is the complexity of the operators commonly used, such as mean, median, etc.). Since these operations need to be repeated for each sensor, the overall time complexity is then $O(N_N^2 + W_E N_N)$ and $O(N_N N + W_E N_N)$ for the centralised and distributed case respectively. Note that the neighbourhood size is the parameter to reduce in large WSNs, e.g., if neighbourhoods are structured into a tree, the complexity can be reduced to $O(Nlog(N) + W_E N_N)$ and $O(log^2(N) + W_E N_N)$ in the centralised and distributed cases respectively.

We consider now the communication overheads. Since in the centralised solution the data needs to be conveyed to the base station anyway, our framework does not introduce additional overhead as we do not transmit additional packages like in [25]. In the distributed scenario, instead, the base station needs to assess the integrity of the aggregation at each aggregator. For this task, solutions have been proposed with a communication overhead sublinear in N, such as the *aggregate-commit-prove* approach [29]. This technique is based on three steps. First the aggregator collects the measurements aggregates them. Then, it reports the result to the base station together with a commitment value. The base station challenges the aggregator by requiring some of the measurements and additional information to verify the commitment value. Based on such information, the base station proves with a certain probability that the committed measurements are authentic and have not been changed during the challenge. Note that although the aggregation alleviates the communication overhead, it also introduces the overhead of verifying the integrity of the processing done by the aggregators. Additionally, since the detection is only performed within a cluster, an adversary may be able to compromise a significant number, or even all, of the sensors in a cluster. The trade-offs involved would benefit from a more detailed investigation which we plan to address in our future work.

VIII. Experiments

Different WSN applications sense different processes, have different semantics and different deployments. It would be cumbersome to design new data injection detection algorithms for each new set of circumstances. Yet few, if any, algorithms proposed in the literature have been shown to work in different scenarios or identify how they can be tailored to them.

In the previous sections: We have seen that we use estimation models to make each sensor express its expectation about the measurement a neighbour should report (see Sect. IV). We provided algorithms to compare the calculated estimates with the reported measurements (see Sect V) and to detect the compromised sensors in case such comparison indicates the presence of malicious data injections (see Sect. VI). We framed all these operations into a methodology that leads the final deployment into specific WSN applications.

We now evaluate our algorithms and methodology in three very different contexts: *Monitoring health parameters* (Sect. VIII-A), *detecting volcanic eruptions* (Sect. VIII-B) and *home fire alarms* (Sect. VIII-C). In each case, we divide the data into two datasets: A historical dataset, from which the estimation models and other parameters are learnt, and a test set, which represents the online measurements and is used for evaluation.

Ideally, we would need datasets containing both genuine measurements and malicious data injections in order to calculate the False Positive Rate (FPR) and the False Negative Rate (FNR). However, real malicious data injections are not common yet, so we need to simulate the attacks in each case to study our algorithm's behaviour. Since it is reasonable to assume that the test sets used in our experiments are free from malicious injections, the FPR estimates are still reliable. However, FNR estimates are not, so we omit them.

We consider both attacks where non-existent events are elicited and where real events are masked. We recognise that a broad range of attacks can be conceived according to: *i*) the number of sensors compromised, *ii*) whether the compromised sensors can be chosen, *iii*) the amount of time available for the attack, *iv*) whether the neighbouring sensors' measurements can be overheard, and *v*) whether the attack needs to be carried out at a specific moment. A systematic investigation of all data injection attacks has not, to our knowledge, been carried out. Our threat model considers attackers that: *i*) have compromised a subset of sensors (its size depending on the application), *ii*) can target the best sensors (i.e., those that can mislead the event detection algorithm with maximum chance of remaining undetected), *iii*) have infinite time for the attack, *iv*) can overhear the measurements of the other sensors, and *v*) carry out the attack as soon as possible.

We simulate the attacks by injecting measurements describing normal circumstances (i.e. absence of faults and compromise) but that subvert the event detection result, i.e. elicit a non existent event, or mask a real event. In some cases, the attacker may need to inject measurements substantially different from the observed ones, but this will not be easily noticeable because the data describes wrong but still normal circumstances. Moreover, the measurements injected from different compromised sensors collude to be perfectly adherent to the expected correlation, thus estimates from two compromised sensors will always support each other. Note that such sophisticated attacks require a sound and well-planned strategy, and thus are difficult to automate. The purpose of the following experiments is indeed to show that, even with sophisticated collusion strategy, our algorithm is capable of detecting the attack and, under certain conditions, to correctly characterise the compromised sensors. Even though experiments based on naive principles, such as random increases in the measurements magnitude or amplification of noise power, lead to experiments that are easier to automate and to results that are easier to interpret, they do not give any information about the ability to detect attacks in real scenarios.

Since our experiments noticeably differ from those carried out in state-of-the-art papers, we implemented the common majority-voting framework [15], [17], [25], [26] and compared the results with those obtained with our estimation-based framework. To make the comparison as fair as possible, we considered votes given by local applications of the similarity check chosen for our algorithm. Hence, if we use the magnitude test, the votes will be pairwise magnitude tests $\frac{\epsilon_{ij}}{std(\epsilon_{ij})} < T_M$; instead if we use the shape test, the votes will be pairwise shape tests $\frac{R_{S_i,S_j} - MED(R_{O_i,O_j})}{D_{ij}} < T_S$. We set the threshold T_M or T_S to achieve the same FPR achieved by our algorithm and classify as compromised the nodes for which at least 50% votes are reported. When we introduce collusion, we assume that compromised nodes do not report on themselves, while they report on the genuine nodes.

A. PhysioNet Dataset

We start with the *PhysioNet MIMIC II Waveform Database* [40], which contains thousands of recordings collected from bedside patient monitors in adult and neonatal intensive care units. We consider blood pressure (mean, systolic, and diastolic), heart rate, pulse, respiration and oxygen saturation. Note that we are considering sensors measuring different physical phenomena, however our algorithm is still applicable since it abstracts from the physical meaning of the measurements and only requires that there is correlation between them. We use the event detection algorithm described in [41] and tested on the *PhysioNet MIMIC Database*, a predecessor of the one we use. An event is triggered when there are high temporal variations in the measurements of the sensors within a time window, which in our experiments is 40 samples long. Table I summarises the experiment setup.

TABLE I: PhysioNet experiment setup

PhysioNet		
Data	Content	Health-related parameters
	Sampling period	1 minute
	Number of sensors	8
	Historical data size	6282 samples per sens.
	Test set size	3412 samples per sens.
Event detection	Criterion	Described in [41], based on temporal variations
	Time window size	40
Algorithm	Similarity check	Shape test
	Similarity check parameters	$W_{Sm} = 10$, $C_R = 99.7$
	Modality assumption	Unique modality for event presence and absence
	FPR	0.002
	Expected false positive frequency	1 in 8 hours

Fig. 4a shows the original measurements from the dataset and the events detected. To simulate malicious data injections, we assume that an attacker has compromised 3 sensors and injected measurements registered by the same sensors while monitoring healthy patients. If the attacker chose uncorrelated sensors, well correlated neighbours would detect the attack. Instead, we choose 3 correlated sensors and inject correlated values from the same sensors in non-event conditions that also show the best correlation with the remaining sensors: the sensors for mean, systolic, and diastolic blood pressure. As shown in Fig. 4b, no event is detected after the attack, although the patient is subject to a life-threatening condition.

To apply our algorithm, we first need to define the similarity check. Since the event detection triggers on fast increasing or decreasing measurements, regardless of their magnitude, we chose the shape test. The shape deviation for the data after the malicious data injections is shown in Fig. 4c. Note that one of the genuine sensors also fails the shape test in the last part; this is due to the collusion effects explained in Sect. VI. Nevertheless, the characterisation algorithm described in Sect. VI correctly returns the set of compromised sensors, and does not include any genuine sensors.

We have run the experiments with other sets of compromised nodes and verified that the attacker needs to compromise at least one more sensor to become undetected – the one closest to those already compromised (Fig. 4b). When this sensor is also compromised, each compromised sensor has 4 genuine neighbours and 3 compromised ones, but there is not enough information to determine if the estimates of the genuine neighbours disagree because of errors or because the other sensors are malicious. Nevertheless, we conclude that in this scenario an attacker needs to compromise 50% (4 out of 8) sensors to remain undetected.

We have applied majority voting as well and obtained no detection at all when 1 or 2 compromised sensors were injecting malicious measurements. In the case of Fig. 4b, with 3 compromised sensors, it did not identify any malicious sensor but also misclassified 2 genuine sensors as compromised.

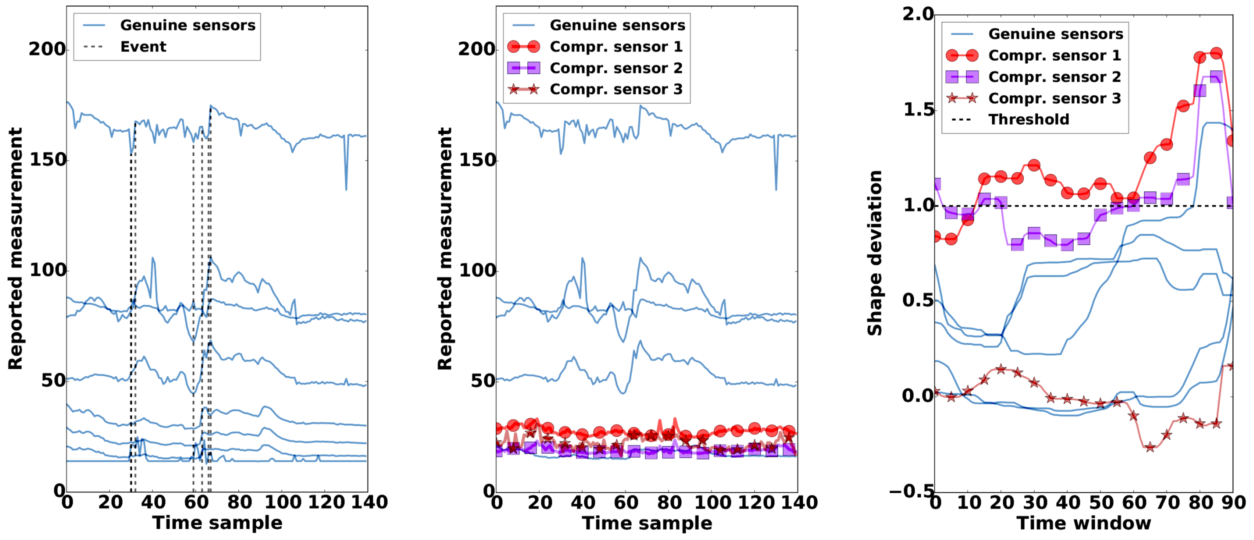
B. Home Fire Alarm Dataset

Our second dataset originates from a WSN conceived for monitoring homes to generate fire alarms. It is available from the NIST website [42] as part of the Home Smoke Alarm Tests project (see NIST Technical Note 1455 [43]). We considered the portion of data collected by three groups of temperature sensors in three adjacent rooms, each group made up of 5 sensors placed on a wall at different distances from the ceiling. Table II summarises the experiment setup.

TABLE II: Home Fire Alarm experiment setup

Home Fire Alarm		
Data	Content	Temperature readings in a room at different distance from the ceiling
	Sampling period	2 seconds
	Number of sensors	15
	Historical data size	14769 samples per sens. at rest, 800 in event mod.
	Test set size	9210 samples per sens. at rest, 312 in event mod.
Event detection	Criterion	One sensor observes 3 consecutive measurements above 50 °C
	Time window size	3
Algorithm	Similarity check	Magnitude test
	Similarity check parameters	T_M = 3
	Modality assumption	One modality for event-presence, one modality for event-absence
	FPR	0.003 in event mod., 0 at rest
	Expected false positive frequency	1 every 3000 fires

In this case, we adopted the second modality assumption (Sec. IV-A1) where different relations between sensors are present in event and non-event conditions. Intuitively, in the absence of a fire, temperatures are broadly uniform but a fire



(a) Alarms with genuine data (b) Alarms with compromised data (c) Shape deviation

Fig. 4: PhysioNet Dataset: masking attack. Alarm conditions are present and the compromised sensors mask them all. The shape test fails on 2 compromised sensors and also on a genuine one, because of collusion.

introduces spatial patterns across the rooms that are significantly different. Linear relationships are therefore derived for both modalities, the event detection algorithm defines which modality applies, and finally the estimates are calculated according to that modality.

Our synthetic attack consists of eliciting a false fire alarm. Here, an event is detected when a fixed temperature is reached, in compliance with the detection algorithm for fixed temperature heat detectors [43]. Such devices generally trigger at temperatures between 47 °C and 58 °C, so we chose 50 °C in our experiment. To make the signals more credible we considered 3 compromised sensors that collude in reporting high temperatures. We simulated the injection by progressively increasing the measurements of 3 sensors as shown in Fig. 5a. This eventually triggers a false fire alarm.

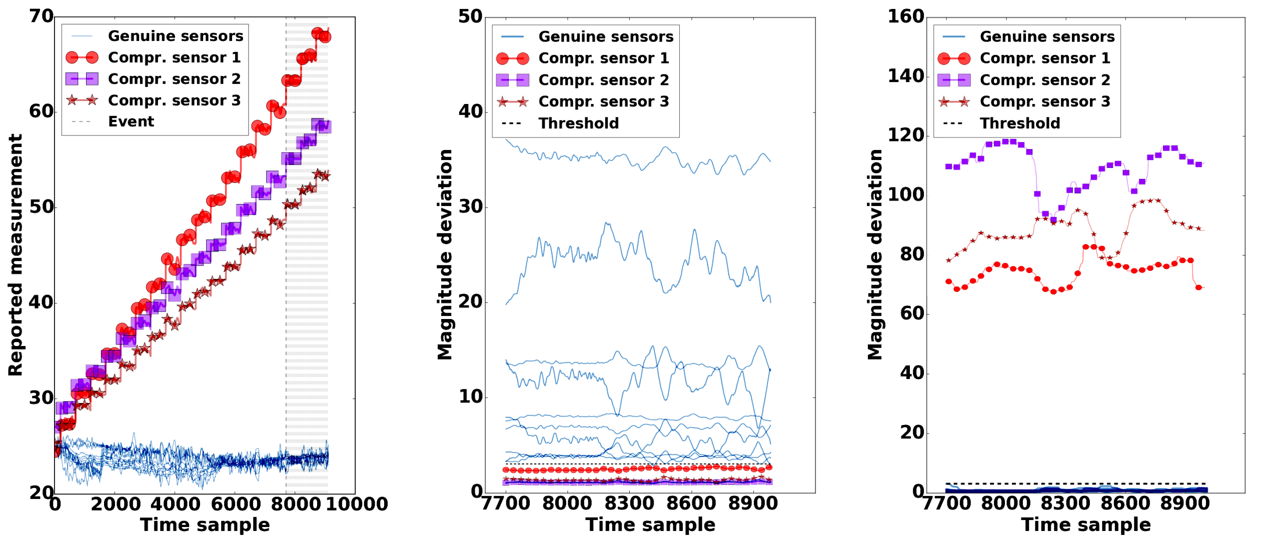
We adopted a similarity check using the magnitude test since the event is detected based on the magnitude of measured values. As explained in Sect. V-A, our magnitude test reflects the event detection criterion, which triggers alarms when the temperature is high. As shown in Fig. 5a, the event is detected around sample 7700, hence, in the first place, the estimation models learnt under event-conditions apply here. Note that the measurements of the genuine sensors are not consistent with the presence of the fabricated event, hence the similarity check fails for most of them as shown in Fig. 5b and the characterisation algorithm identifies 12 nodes (10 of which are genuine). Since the check failed, the characterisation algorithm (see Section VI) needs to decide if the event modality is the correct one and it does so by running the detection again on the same measurements but with the estimation models for the non-event modality. This time, the characterisation algorithm returns the 3 compromised sensors, a smaller set than the 12 nodes identified with the event modality, so the non-event modality is correctly chosen and the compromised sensors are correctly detected. The magnitude deviation under the non-event modality is shown in Fig. 5c.

We have run the experiments with other sets of compromised nodes and verified that the attacker needs to compromise at least 7 sensors to remain undetected. In this scenario, 11 sensors are identified in the non-event modality and 8 sensors (all the genuine ones) in the event modality, therefore the event modality is wrongly chosen and the genuine sensors are classified as malicious. We conclude that in this scenario the attacker needs to compromise at least 47% sensors for a successful undetected attack.

For this experiment majority voting detected correctly the compromised nodes when 1 sensor was compromised. Detection succeeds with 2 compromised nodes but misclassifies 2 genuine sensors. In the case shown in Fig. 5a with 3 compromised sensors, detection fails and 4 genuine nodes are misclassified. Note that the real limit of majority voting here is 13% (2 sensors), which is far below the 50% theoretical limit. Such limit requires correct votes from the genuine nodes, which are guaranteed only with perfect correlation.

C. *Reventador Volcano Dataset*

We finally consider a dataset gathered from a WSN of infrasound sensors deployed at the Reventador volcano by the Harvard Sensor Networks Lab [44]. The network consists of 16 sensor nodes deployed every 200-400 metres along a side of the volcano. The sensors are connected to a remote base station, which waits for the sensors to trigger the presence of an event (which may reflect earthquakes related to eruptions): if at least 30% sensors trigger, the base station collects the lasts 60 seconds of data from all the sensors to analyse the event. The event detection algorithm, given in [44] is based on temporal changes in the measurements average. Here we consider the measurements shown in Fig. 6a, which trigger the



(a) Alarms with compromised data, represented with vertical lines (b) Magnitude deviation in alarm modality (c) Magnitude deviation in non-alarm modality

Fig. 5: Home Fire Alarm Dataset: eliciting attack. The 3 compromised sensors trigger the fire alarm. Many sensors fail the check in the wrong modality. In the correct modality, the compromised sensors can be easily identified.

event detection[3]. Table III summarises the experiment setup.

TABLE III: Reventador experiment setup

Reventador Volcano		
Data	Content	Infrasound readings along a side of the volcano
	Sampling period	0.01 seconds
	Number of sensors	8
	Historical data size	28471 samples per sens.
	Test set size	8398 samples per sens.
Event detection	Criterion	EWMA [44], based on changes in average
	Time window size	6000 samples
Algorithm	Similarity check	Shape test
	Similarity check parameters	$W_{Sm} = 1$, $C_R = 99.7$
	Modality assumption	Unique modality for event presence and absence
	FPR	0
	Expected false positive frequency	0

Before running the experiments, we note that infra-sound data is made up of high-frequency oscillations around zero. Applying our algorithm to the raw data would be inappropriate since the measurements are mostly uncorrelated and uninformative. In infrasound measurements, the valuable information is mostly contained in the trend of the peak values, which can be captured with a pre-processing step that averages the measurements' absolute value in a short time window. From the graphs in 6a we note that peak values are generally consistent for about 400 data samples, so we used a pre-processing time window of 400 samples. The pre-processed measurements are shown in Fig. 6b.

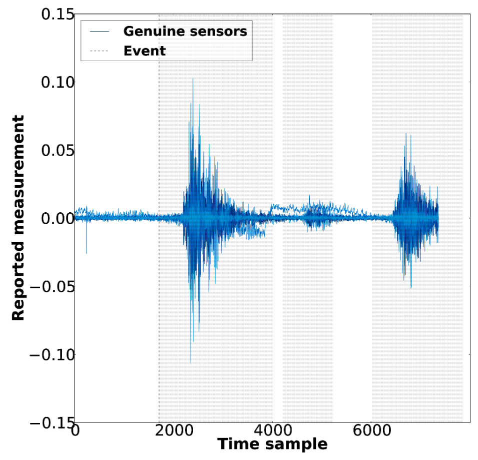
For this experiment, we simulate masking attacks that silence the event detection by injecting measurements taken from the same sensors, but in restful conditions, i.e. when there is no volcanic activity. In the pre-processed data, injected measurements appear as roughly constant data, i.e. without increasing/decreasing trends. According to the event detection algorithm described in [44], an event occurs when 2 or more sensors trigger (30% of 8). We observe that the event is triggered by all the sensors except sensor 4, which is the one with a roughly flat signal in Fig. 6b. So, to mask the event, an attacker needs to compromise at least 6 sensors that do not include sensor 4. As conveyed in Section V, the similarity check should be tailored to the event detection algorithm, that in this case is a exponentially-weighted moving average (EWMA). Since EWMA triggers with temporal variations of the average signal, including sudden but small variations, the shape test is more appropriate than the magnitude test for the similarity check.

We first ran an experiment that simulates a scenario where only sensor 2 is injecting malicious measurements. The results are shown in Table IV. We note that the shape test unequivocally recognises the inconsistency of the measurements from sensor 2. Sensor 4 is also not triggering, however the similarity check does not fail on it since its measurements are consistent with the estimation models, which indirectly captured the characteristic that the sensor may not trigger even if all the other sensors are triggering.

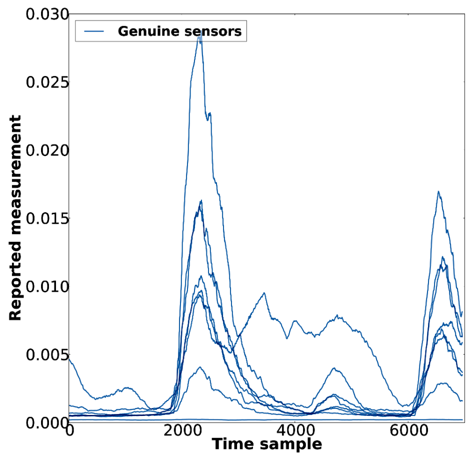
We then progressively increased the number of sensors injecting malicious measurements. In Table IV we report the output from the characterisation algorithm in the order in which sensors are found. Note that whenever the similarity check fails, the genuine nodes that failed the test because of the collusion effect are finally classified as genuine. With the exception of sensor 5, all the compromised sensors are correctly detected by the characterisation algorithm. Sensor 5 is the sensor in Fig. 6b, whose measurements mostly dissociate themselves from the others. The behaviour of this sensor is therefore less predictable, and a considerable deviation is required to make the similarity check fail.

When sensors $\{1,2,3,5,6,7\}$ are all injecting malicious mea-

[3]Note that only 8 sensors are analysed since the remaining 7 sensors were not working during the observed time.



(a) Alarms with genuine data



(b) Pre-processed genuine data

Fig. 6: Reventador Dataset. Alarming conditions are present. The measurements are noticeably more correlated after the transformation.

TABLE IV: Reventador shape deviation. The values that failed the shape test are in boldface

Compromised Sensors	Event masked	S_1 dev.	S_2 dev.	S_3 dev.	S_4 dev.	S_5 dev.	S_6 dev.	S_7 dev.	S_8 dev.	Characterisation output
2	NO	0.03	**25.13**	0.32	-0.04	0.05	-0.04	-0.07	0.03	2
2,6	NO	0.07	**25.27**	0.23	-0.05	0.06	**8.48**	0.03	0.08	2,6
1,2,6	NO	**8.12**	**25.15**	**4.47**	0.01	0.07	**8.66**	**2.52**	**1.40**	2,6,1
5,6,7	NO	-0.01	0.31	0.09	-0.05	0.89	**8.55**	**13.36**	0.08	7,6
5,6,7,8	NO	**2.04**	**6.50**	**7.22**	-0.05	0.89	**8.41**	**13.17**	**6.75**	7,6,8
1,2,3,5,6,7	YES	**4.28**	**11.72**	**15.14**	0.04	0.48	**5.63**	**8.07**	**6.94**	3,2,7,1,6
1,2,3,5,6,7,8	YES	-0.05	-0.07	-0.05	0.40	-0.98	-0.05	-0.14	-0.10	N/A

surements, the masking attack eventually succeeds. However, our algorithm still detects the attack, even though 75% sensors are reporting malicious measurements, as the colluding sensors did not succeed in making the measurements credible. The attacker needs to compromise all the triggering sensors (88% of the total) to carry out an undetected masking attack as shown in the last row of Table IV.

In the same settings, majority voting fails with 1 compromised node (13%). With 2 or 3 compromised nodes (25%, 37%), the detection succeeds but 2 genuine nodes are classified as malicious. With 4 (50%) nodes, it reaches its theoretical limit, so detection clearly fails. Instead, our algorithm detects the attack without false positives when even 75% sensors are compromised and reaches its limit at 88%.

D. Discussion

Our current approach has shown to detect malicious interference also with sophisticated attacks, based on injection of credible measurements. Based on our characterisation algorithm, we are able to detect correctly the set of compromised sensors when the number of genuine sensors is low compared to the expected correlation. Note that, in our approach, the number of compromised sensors that can be tolerated is correlation-dependant. In one of our experiments attacks could be detected whenever fewer than 88% sensors were compromised. Voting-based frameworks instead, cannot tolerate more than 50% compromised sensors and, when our algorithm tolerated less than 50% compromised sensors, majority voting tolerated a substantially lower percentage. The reason behind this result is that the correlation between the sensors used in the experiments is not high enough to guarantee correct votes from all the genuine sensors and votes become inaccurate. Our approach deals with such inaccuracy by merging the contributions from each sensor, weighting the contributions according to their expected accuracy and discarding potentially unreliable contributions.

We have also seen that the attack detection does not always mean correct identification of the compromised nodes, especially when the correlations change dramatically between restful and event conditions. In this case indeed, we can easily detect the presence of a problem but cannot easily infer whether the restful or event-related measurements are correct. Our approach chooses the most likely condition, but we foresee the possibility of rejecting the detection when there is not enough confidence in it. The final decision may be taken otherwise and may require human intervention.

IX. Conclusions and Future Work

In this paper we have focused on detecting malicious data injections in event detection WSNs, in particular when collusion between compromised sensors occurs. We have proposed an algorithm that can be customised and used in different applications, and for different kinds of events.

Dealing with collusion and the occurrence of events makes the problem of detecting malicious data injections significantly more complex because both affect the dynamics of the system and comparisons between measurements. Furthermore, they interact with each other as collusion may leverage deviations in sensed values introduced by the event.

Addressing this challenge has exposed several trade-offs in the design of the algorithm. Firstly, resistance to collusion requires to compare measurements over a broader set of sensors and thus introduces additional complexity and computational cost. This trade-off is particularly visible in the selection of neighbourhoods, which becomes a simple ranking-based choice when using our pairwise estimation models. Another trade-off arises when merging information with potentially malicious sources. While information coming from genuine sensors increases the estimates accuracy, it is important to select only information that appears reliable. Colluding sensors should not be allowed to compensate for each other in the detection metric whilst still injecting malicious data. This requires the use of pairwise comparisons and an aggregation operator that is accurate in the presence of genuine measurements as well as resistant to malicious data.

From applying our methodology in three applications, where requirements and the nature of the events is markedly different, we conclude that the development of a general framework to cope with malicious data injection in event detection WSNs is possible. However, it requires customisation of the parameters based on a collection of historical data and information about the application's goals and requirements. The methodology for customisation, on the other hand, can be provided with a generic but well defined procedure.

Experimental results validated the choice of structuring the detection on top of simple techniques that, without introducing significant overhead in the sensor nodes, achieve high detection rates. These results encourage us to pursue further investigations in this area. In future work, we aim to extend the methodology to cases where events cause unpredictable changes in the spatial patterns. We also aim to investigate WSN applications where more sophisticated regression methods (e.g., polynomial regression or generalised linear models [35], [38]) may be more appropriate.

Acknowledgment

We wish to thank the anonymous reviewers for their valuable comments and suggestions, which have improved this paper. We are grateful to Dr. Igor Muttik from Intel and to the colleagues in our research group, who have contributed to this work through many useful discussions. This work was funded as part of the ICRI on Sustainable Connected Cities.

References

[1] T. He, S. Krishnamurthy, J. A. Stankovic et al., "Energy-Efficient Surveillance System Using Wireless Sensor Networks." *MobiSys* 2004.
[2] C. Otto, A. Milenković et al. "System Architecture of a Wireless Body Area Sensor Network for Ubiquitous Health Monitoring," *J. Mob. Multimed.* 2005.
[3] G. Werner-Allen, K. Lorincz et al. "Deploying a Wireless Sensor Network on an Active Volcano." *Internet Computing* 2006.
[4] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures." *Ad Hoc Networks* 2003.
[5] A. Perrig, J. Stankovic et al. "Security in wireless sensor networks" *Commun.* 2004.
[6] W. Du, J. Deng et al. "A pairwise key predistribution scheme for wireless sensor networks." *Trans. Inf. Syst. Secur.* 2005.
[7] M. Mathews, M. Song et al. "Detecting Compromised Nodes in Wireless Sensor Networks." *SNPD (1)* 2007.
[8] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks." *IPSN* 2008.
[9] M. K. Khan and K. Alghathbar, "Cryptanalysis and Security Improvements of 'Two-Factor User Authentication in Wireless Sensor Networks'," *Sensors* 2010.
[10] A. Seshadri, A. Perrig et al. "SWATT: SoftWare-based ATTestation for Embedded Devices." *Symposium on Security and Privacy* 2004.
[11] T. Park and K. G. Shin, "Soft Tamper-Proofing via Program Integrity Verification in Wireless Sensor Networks." *Trans. Mob. Comput.* 2005.
[12] A. Seshadri, M. Luk et al. "SCUBA: Secure Code Update By Attestation in sensor networks." *Workshop on Wireless Security* 2006.
[13] D. Zhang and D. Liu, "DataGuard: Dynamic data attestation in wireless sensor networks." *DSN* 2010.
[14] S. Tanachaiwiwat and A. Helmy, "Correlation Analysis for Alleviating Effects of Inserted Data in Wireless Sensor Networks." *MobiQuitous* 2005.
[15] F. Liu, X. Cheng et al. "Insider Attacker Detection in Wireless Sensor Networks." *INFOCOM* 2007.
[16] M. Rezvani, A. Ignjatovic et al. "A robust iterative filtering technique for wireless sensor networks in the presence of malicious attacks." *SenSys* 2013.
[17] B. Sun, X. Shan et al. "Anomaly Detection Based Secure In-Network Aggregation for Wireless Sensor Networks." *Systems J.* 2013.
[18] M. Raya, P. Papadimitratos et al. "On Data-Centric Trust Establishment in Ephemeral Ad Hoc Networks." *INFOCOM* 2008.
[19] S. Ganeriwal, L. Balzano et al. "Reputation-based framework for high integrity sensor networks." *TOSN* 2003.
[20] S. Rajasegarar, C. Leckie et al. "Quarter Sphere Based Distributed Anomaly Detection in Wireless Sensor Networks." *ICC* 2007.
[21] V. Chatzigiannakis and S. Papavassiliou, "Diagnosing Anomalies and Identifying Faulty Nodes in Sensor Networks," *Sensors J.* 2007.
[22] S. Rajasegarar, J. C. Bezdek et al. "Elliptical anomalies in wireless sensor networks." *TOSN* 2009.
[23] A. B. Sharma, L. Golubchik et al. "Sensor faults: Detection methods and prevalence in real-world datasets." *TOSN* 2010.
[24] Y. Liu, P. Ning et al. "False data injection attacks against state estimation in electric power grids." *Trans. Inf. Syst. Secur.* 2011.
[25] Q. Zhang, T. Yu et al. "A Framework for Identifying Compromised Nodes in Wireless Sensor Networks." *Trans. Inf. Syst. Secur.* 2008.
[26] C. V. Hinds, "Efficient detection of compromised nodes in a wireless sensor network." *SpringSim* 2009.
[27] W. Zhang, S. K. Das et al. "A Trust Based Framework for Secure Data Aggregation in Wireless Sensor Networks." *SECON* 2006.
[28] F. Bao, I.-R. Chen et al. "Hierarchical Trust Management for Wireless Sensor Networks and its Applications to Trust-Based Routing and Intrusion Detection." *Trans. Netw. and Serv. Manag.* 2012.
[29] B. Przydatek, D. X. Song et al. "SIA: secure information aggregation in sensor networks." *SenSys* 2003.
[30] S. Roy, M. Conti et al. "Secure data aggregation in wireless sensor networks: Filtering out the attacker's impact," *Trans. Inf. Forensics Security* 2014.
[31] Y. Sun, H. Luo et al. "A Trust-Based Framework for Fault-Tolerant Data Aggregation in Wireless Multimedia Sensor Networks." *Trans. Dependable Sec. Comput.* 2012.
[32] Y. Zhang, N. A. S. Hamm et al. "Statistics-based outlier detection for wireless sensor networks." *IJGIS* 2012.
[33] D. T. Larose, *Data mining methods and models.* 2006.
[34] X. Wang, L. Ding et al. "Reputation-Enabled Self-Modification for Target Sensing in Wireless Sensor Networks." *T. Instrumentation and Measurement* 2010.
[35] A. Gelman and J. Hill, *Data Analysis Using Regression and Multilevel/Hierarchical Models* 2006.
[36] W. Bryc, *The normal distribution: characterizations with applications*, ser. Lecture notes in statistics 1995.
[37] S. Papadimitriou, H. Kitagawa et al. "LOCI: Fast Outlier Detection Using the Local Correlation Integral." *ICDE* 2003.
[38] C. Bishop, *Pattern Recognition and Machine Learning* 2006.
[39] A. L. Strehl and M. L. Littman, "Online linear regression and its application to model-based reinforcement learning," in *Advances in Neural Information Processing Systems*, 2008.
[40] "PhysioNet," http://physionet.org.

[41] O. Salem, Y. Liu et al. "Online Anomaly Detection in Wireless Body Area Networks for Reliable Healthcare Monitoring," *J. of Biomedical and Health Informatics* 2014.
[42] "NIST," http://www.nist.gov/.
[43] R. W. Bukowski, R. D. Peacock et al. *Performance of home smoke alarms: analysis of the response of several available technologies in residential fire settings* NIST Fire Research Division 2007.
[44] G. Werner-Allen, K. Lorincz et al. "Fidelity and Yield in a Volcano Monitoring Sensor Network." *OSDI* 2006.

Vittorio P. Illiano is PhD student in the Department of Computing at Imperial College London, as part of the Intel ICRI on Sustainable and Connected Cities. His main research area is security in Wireless Sensor Networks, with a focus on anomaly detection and related data analysis techniques. He received the B.Sc. and M.Sc. in Computer Engineering from the University of Naples "Federico II".

Emil C. Lupu received his Diplome d'Ingenieur from the ENSIMAG, Grenoble, France and his PhD from Imperial College London. He is a Reader (Associate Professor) in the Department of Computing at Imperial College London and an Associate Director of Imperial's Institute for Security Science and Technology where he leads the Academic Centre of Excellence in Cyber Security Research (ACE-CSR). His work focuses on the engineering of resilient, adaptive and trustworthy systems across a broad range of contexts ranging from IoT to Cloud environments. He has made numerous contributions on policy-based network and systems management and security and serves on the editorial boards of IEEE Transactions on Network and Service Management (TNSM) and the Journal of Network and Systems Management (JNSM).