# Credibility Propagation for Robust Data Aggregation in WSNs

Mohsen Rezvani[1] Aleksandar Ignjatovic[1] Elisa Bertino[2] Sanjay Jha[1]

[1] University of New South Wales, Australia
{mrezvani,ignjat,sanjay}@cse.unsw.edu.au
[2] Department of Computer Science, Purdue University
bertino@cs.purdue.edu

**Technical Report**
**UNSW-CSE-TR-201414**
**May 2014**

THE UNIVERSITY OF
NEW SOUTH WALES


School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

**Abstract**

Trust and reputation systems are widely employed in WSNs in order to help decision making processes by assessing trustworthiness of sensor nodes in a data aggregation process. However, in unattended and hostile environments, some sophisticated malicious attacks such as collusion attacks can distort the computed trust scores and lead to low quality or deceptive services as well as to undermine the aggregation results. Thus, taking into account the collusion attacks for developing a secure trust-based data aggregation in unattended environments has become an important research issue.

In this paper, we address this problem by proposing a novel collaborative-based trust framework for WSNs, which is based on the introduced concept of credibility propagation. In this method, the trustworthiness of a sensor node is evaluated from the amount of credibility that such a node collects from other nodes. Moreover, we obtain the statistical parameters of sensors errors including sensors' variances from such credibility values. Accordingly, we propose an iterative filtering algorithm to recursively compute the credibility and variance of all sensors. Following this algorithm, an estimate of the true value of the signal can be effectively obtained through the maximum likelihood estimation. Furthermore, we augment the proposed trust framework with a collusion detection and revocation method as well as data streaming algorithm. Extensive experiments across a wide variety of configurations over both real-world and synthetic datasets demonstrate the efficiency and effectiveness of our approach.

# 1 Introduction

Trust and reputation systems have a significant role in supporting operation of a wide range of distributed systems, from wireless sensor networks (WSNs) and e-commerce infrastructure to social networks, by providing an assessment of trustworthiness of participants in such distributed systems. A trustworthiness assessment at any given moment represents an aggregate of the behaviour of the participants up to that moment and has to be robust in the presence of various types of faults and malicious behaviour. There are a number of incentives for attackers to manipulate the trust and reputation scores of participants in a distributed system, and such manipulation can severely impair the performance of such a system [1, 2].

In recent years, several approaches based on iterative filtering (IF) algorithms for trust and reputation systems have been proposed [3, 4, 5, 6, 7, 8, 9, 10]. In the past literature it was found that these algorithms exhibit better robustness compared to the simple averaging techniques; however, these work did not take into account more sophisticated collusion attack scenarios [11]. For example, we showed in [11] how the IF algorithms can be compromised by a collusion attack when the adversary has enough knowledge about the reputation computation method.

In this paper, we propose a novel collaborative reputation system for WSNs based on the idea of credibility propagation among the participants, called CrPr. Based on this idea, each sensor collects credibility from all other sensors according to the distance of its readings to the readings of other sensors in which such a distance depends on the sensors variances. On the other hand, each sensor gives credibility to other sensors based on both the likelihood of its readings to the true value of signal and the distance of its readings to the readings of other sensors. We exploit an iterative procedure for simultaneously computing the trustworthiness of each sensor and the variance of its errors. We then use the obtained values of the sensors variances to form an statistical estimator which if the sensors errors are independent and normally distributed, represents the maximum likelihood estimator (MLE). Furthermore, we augment the reputation system with a collusion detection and revocation method which identifies the compromised nodes and eliminates them according to the normality of the error behaviour in the sensor nodes.

Since readings keep streaming into aggregator nodes in WSNs, and since attacks can be very dynamic (such as *orchestrated* attacks [2]), we extend CrPr algorithm to data streaming and propose a real time version of the algorithm, called Str-CrPr, which builds a model of sensors errors and updates the model at any time instant.

We provide a thorough empirical evaluation of effectiveness and efficiency of our proposed reputation system by using real world dataset as well as testing the system on synthetically generated datasets. The results show that our method provides both higher accuracy and better collusion resistance than the existing IF methods.

Our contributions can be summarized as follows:

- A novel collaborative reputation system for WSNs which is effective in a wide range of sensor faults and not susceptible to collusion attacks.
- An iterative algorithm to estimate the true value of the signal based on an

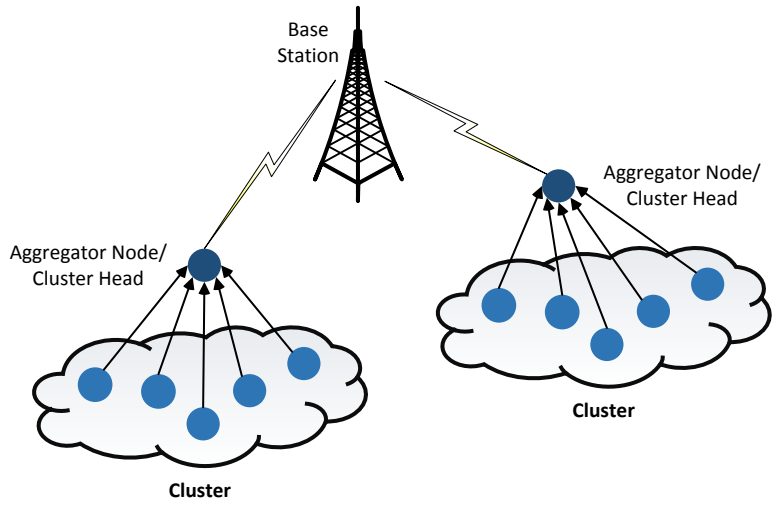



Figure 2.1: Network model for WSN.

interdependency relationship among credibility, variance and reputation values.

- A novel collusion detection method based on an estimate of normality of sensor errors in the proposed robust reputation framework.
- Extending the proposed reputation system to data streaming.

It is to be noted that while our reputation system is designed having WSNs in mind, it is straightforward to extend our solution for trust and reputation system in other applications such as online rating.

The rest of this paper is organized as follows. Section 2 formulates the problem and specifies the assumptions. Section 3 presents our novel reputation system. The real time version of our reputation system is described in Section 4. Section 5 describes our experimental results. Section 6 presents the related work. Finally, the paper is concluded in Section 7.

# 2 Preliminaries

## 2.1 Network Model

For the sensor network topology, we consider the abstract model proposed by Wagner in [12] presented on Figure 2.1. The sensor nodes are divided into disjoint clusters, and each cluster has a cluster head which acts as an aggregator. Data are periodically collected and aggregated by the aggregator. In this paper we assume that the aggregator itself is not compromised and concentrate on algorithms which make aggregation secure when the individual sensor nodes might be compromised and might be sending false data to the aggregator. We assume that each data aggregator has enough computational power to run an iterative filtering algorithm for data aggregation.

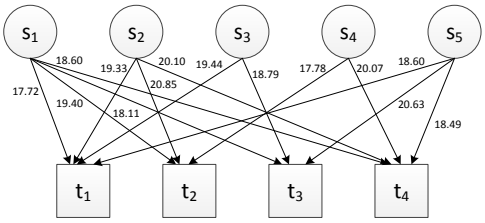



Figure 2.2: An example of ratings.

## 2.2 Basic Concepts and Notation

Galletti et al. recently proposed in [13] a mathematical framework for modelling collaborative reputation systems. In this paper, we use their notation and assumptions for describing our novel reputation system.

Let us consider a WSN with $n$ sensors $S_i$, $i = 1, \ldots, n$. We assume that the aggregator works on one block of readings at a time, each block comprising of readings at $m$ consecutive instants[1]. Therefore, a block of readings is represented by a matrix $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ where $\mathbf{x}_i = [x_i^1\ x_i^2\ \ldots\ x_i^m]^T$, $(1 \leq i \leq n)$ represents the $i^{th}$ $m$-dimensional readings reported by sensor node $S_i$. Let $\mathbf{r} = [r_1\ r_2\ \ldots\ r_m]^T$ denote the aggregate values for instants $t = 1, \ldots, m$, which some authors call a *reputation vector*[2] [4]. Suppose that we would like to obtain the reputation values from the existing readings $X$.

We can also visualize the readings model using a directed and weighted bipartite graph $G = (S, I, R)$, where $S$ is a set of S-typed nodes representing sensor nodes, $I$ is a set of I-typed nodes representing time instants, and $R$ is a set of directed edges from S-typed nodes to I-typed nodes [3]. For example, a reading given by a sensor $p \in S$ in time instant $t \in I$ is denoted as $x_p^t$, which is the weight associated with the edge $(p, t) \in R$ in the graph. Thus, the absence of an edge indicates that the sensor node did not report reading for the time instant. Table 2.1 contains a summary of notation used in this paper.

**Example 2.1.** *An example is shown by a readings matrix and a bipartite graph in Table 2.2 and Figure 2.2, respectively (In order to make the graph clear, we removed 6 edges from the graph in Figure 2.2). The sensor readings in the example are from sensed temperatures in Intel Lab dataset [14] at four different time instants* $I = \{t_1, t_2, t_3, t_4\}$ *and 5 sensors* $S = \{s_1, s_2, s_3, s_4, s_5\}$*. For example in the readings, sensor* $s_2$ *reported the temperature at time* $t_3$ *of value 18.95 (*$x_2^3 = 18.95$*). Given such a scenario, by taking the average of readings at each instant, the reputation vector for the instants is* $\mathbf{r} = \langle r_1, r_2, r_3, r_4 \rangle = \langle 18.79, 18.74, 18.69, 18.57 \rangle$.

Note that the main objective of a reputation system is to determine the reputation values for all time instants based on the readings reported by sensors

[1] In Section 4, we extend this assumption by taking into account the streaming data and propose our real time reputation system.

[2] We find such terminology confusing, because reputation should pertain to the level of trustworthiness rather than the aggregate value, but have decided to keep the terminology which is already in use. Throughout this report, the terms reputation value and aggregate value are used interchangeably.

[3] We use the graph visualisation method from [7] as an alternative method to present our model.

Table 2.1: Notation used in this paper.

| Notion | Meaning |
|---|---|
| $n$ | number of sensors |
| $m$ | number of readings for each sensor |
| $r_t$ | true value of the signal at time $t$ |
| $x_s^t$ | data from sensor $s$ at time $t$ |
| $\sigma_s$ | standard deviation of sensor $s$ $t$ |
| $\mathrm{L}(j,i)$ | the likelihood that sensor $j$ could have made the readings of sensor $i$ |
| $\mathrm{cr}(s)$ | credibility of sensor $s$ |
| $\mathrm{rep}(t)$ | estimate reputation value at time $t$ |
| $\mathrm{var}(s)$ | estimate variance of sensor $s$ |
| $N(i)$ | neighborhood of node $i$ in the readings graph |
| $T_{i,j}$ | set of time instants which both sensor $i$ and $j$ have reported for them |
| $n_i$ | set of sensor nodes which have reported at least for one common time instant with sensor $i$ |
| $d_\Gamma^S$ | the average number of readings per sensor |
| $\langle i, v_i, t_i, E_i, O_i \rangle$ | Str-CrPr Model with $i$ is a sensor identifier, $v_i$ is the estimated variance of sensor $i$, $t_i$ is the number of readings has been reported by sensor $i$, $E_i$ is the cumulative error of sensor $i$, and $O_i$ is the number of outlier readings reported by sensor $i$. |
| $w$ | the length of sliding window |
| $t_w$ | the last time of the model rebuilding |
| $\delta$ | threshold for number of outliers |
| $\varepsilon$ | threshold for variance divergence |
| $\kappa$ | relaxation time for rebuilding process |
| $d_t$ | variance divergence at time $t$ |
| $\tau$ | Pearson correlation coefficient |
| $c$ | number of malicious sensor nodes |
| $\eta$ | density factor |

which reflect the measurement of the true value of the signal. In the rest of this paper, we assume that the stochastic components of sensor errors are independent random variables with a Gaussian distribution; however, our experiments show that our method works quite well for other types of errors without any modification. However, if the error distribution of sensors is known, our algorithms can be adapted to other random distributions to achieve an optimal performance. Hence, we assume that the reading of sensor $s$ at time instant $t$ is

$$x_s^t = r_t + e_s^t \tag{2.1}$$

Table 2.2: A trace example of readings.

| | **sensor readings** | | | | |
|---|---|---|---|---|---|
| **instant** | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
| t=1 | 17.72 | 19.33 | 19.44 | 18.83 | 18.60 |
| t=2 | 19.40 | 20.85 | 18.63 | 17.78 | 20.33 |
| t=3 | 18.11 | 19.68 | 18.79 | 19.83 | 20.63 |
| t=4 | 18.60 | 20.10 | 18.57 | 20.07 | 18.49 |

where $r_t$ is the latent reputation at time instant $t$, that is hidden to the external world, and $e_s^t$ is a random variable with a zero-mean Gaussian distribution $e_s^t \sim \mathcal{N}(0, \sigma_s^2)$. Our goal is therefore to recover the values of $r_t$ from the sensor readings without knowing a priori sensors variances. We assume the normal distribution since it well reflects nature of the noise [15]. However, we note that the normal distribution assumption is not a limitation of our reputation system. We use the probability distribution for estimating the similarity among readings in the proposed credibility computation as well as testing the error behaviour of sensors in our collusion detection module. We can adopt other probability distributions with simple changes to both the data credibility computation model and the collusion detection module.

## 2.3 Threat Model

In this paper, we use a Byzantine attack model, where the adversary can compromise a set of sensor nodes and inject any false data through the compromised nodes [16]. When sensors are distributed in an unattended environment such as a cyber-physical system, taking into account this attack model is become a significant security challenge [17, 18].

For describing the threat model, we assume that sensors are deployed in a hostile unattended environment. Consequently, some nodes can be physically compromised. We assume that when a sensor node is compromised, all the information which is inside the node becomes accessible by the adversary. Thus, we cannot rely on cryptographic methods for preventing the attacks, since the adversary may extract cryptographic keys from the compromised nodes. We assume that through the compromised sensor nodes, the adversary can send false data to the aggregator and distort the aggregate values.

Furthermore, we assume that the attacker can either work individually or collaboratively. In individual (non-collusion) attacks, false data is injected independently. In the collaborative (collusion) attacks, the attacker controls multiple sensors that coordinately report false data [19]. For both individual and collaborative attackers, we consider *Slandering* attacks that the attacker manipulates the reputation of some instants by reporting false data either to increase or decrease the reputation values. We also consider an *Orchestrated* attack that the attacker launches a sophisticated collusion attack in order to undermine the performance of the entire reputation system [11]. Moreover, we assume that the attacker can obtain or estimate the parameters used in the reputation system, and adjust its attack strategies accordingly.

Finally, we assume that the base station and aggregator nodes cannot be

compromised in this adversary model; there is an extensive literature proposing how to deal with the problem of compromised aggregators [20, 21, 22, 23]; in this paper we limit our attention to the lower layer problem of false data being sent to the aggregator node by the compromised individual sensor nodes, which has received much less attention in the existing literature.

### 2.4 Problem Statement

Our proposed reputation system is motivated by our recent work on discovering a new and sophisticated collusion attack against the existing IF algorithms for reputation system [11]. In this paper, our objective is to propose a novel reputation computation algorithm with the following features:

- the proposed algorithm is robust with respect to outliers. If only a very small fraction of readings $x_s^t$ are far from some form of a "consensus" of other sensors, such readings have very little impact on the final aggregated reputations produced by the system;
- the proposed algorithm is robust to collusion attacks, where a group of sensors tries collaboratively to skew the reputation values by an orchestrated effort;
- the proposed algorithm is "statistically sound"; for example, if individual readings $x_s^t$ provided by any sensor $s$ are "correct values" plus some zero mean Gaussian noise independent for each instant, then the algorithm should produce an output close to the optimal output which is produced by the MLE;
- sensors variances are not an input to the proposed algorithm, because such kind of information is unavailable in practice; the algorithm uses an adaptation procedure which automatically provides such output merely from the statistical features of the "raw" inputs.

## 3 Collaborative Reputation System

Since some of our terminology is also used either in everyday English or academic fields, in this section, we first define the terms "credibility" and "variance" precisely before describing and analyzing the proposed iterative algorithm to compute them. We first assume that all sensors report readings for all time instants and then we extend our model for supporting the sparsity pattern of the readings.

### 3.1 Variance Estimation Principle

In the presence of stochastic errors for sensors readings in WSNs, a reputation computation algorithm should produce estimates which are close to the optimal ones in information theoretic sense. Thus, for example, if the noise present in each sensor is a Gaussian independently distributed noise with a zero mean, then the estimates produced by such an reputation algorithm should have a variance close to the Cramer-Rao lower bound (CRLB), i.e, in such a case it

should be close to the variance of the MLE. However, such estimation should be achieved *without* supplying the algorithm the variances of the sensors.

In the proposed reputation system, we aim to achieve an optimal estimate of sensors variances. With such estimates of the variances, we employ a MLE-like technique to estimate the true value of the signal. We argue that this objective satisfies the above principle according to the following fact[1]:

**Proposition 3.1** (Minimum Variance Estimator)**.** *Assume that the readings of $n$ sensors all have normal distributions but with different and* ***known*** *standard deviations $\sigma_1, \ldots, \sigma_n$. Assume that using these sensors we have obtained measurements $X_1, \ldots, X_n$ for a quantity $r$ of interest. In this case, the MLE is the weighted average of readings of all sensors and it is expressed as follows:*

$$\hat{r} = \sum_{i=1}^{n} \frac{\frac{1}{\sigma_i^2}}{\sum_{k=1}^{n} \frac{1}{\sigma_k^2}} X_i. \tag{3.1}$$

*and such an estimate has minimal possible variance.*

*Proof.* Since the standard deviations of sensors are now known, in this case we can form the likelihood function with only one parameter, the expected value $r$:

$$\mathcal{L}_n(r) = \prod_{i=1}^{n} \frac{1}{\sigma_i\sqrt{2\pi}}\, e^{-\frac{1}{2}\frac{(X_i-r)^2}{\sigma_i^2}} = \left(\prod_{i=1}^{n} \frac{1}{\sigma_i\sqrt{2\pi}}\right) e^{-\frac{1}{2}\sum_{i=1}^{n}\frac{(X_i-r)^2}{\sigma_i^2}}$$

Differentiating with respect to $r$ and setting the derivative equal to zero we get

$$\frac{d}{dr}\mathcal{L}_n(r) = \left(\prod_{i=1}^{n} \frac{1}{\sigma_i\sqrt{2\pi}}\right) e^{-\frac{1}{2}\sum_{i=1}^{n}\frac{(X_i-r)^2}{\sigma_i^2}} \sum_{i=1}^{n} \frac{(X_i - r)}{\sigma_i^2}$$

$$\frac{d}{dr}\mathcal{L}_n(r) = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{n} \frac{X_i}{\sigma_i^2} - r\sum_{i=1}^{n} \frac{1}{\sigma_i^2} = 0 \quad \Leftrightarrow \quad r = \frac{\sum_{i=1}^{n} \frac{X_i}{\sigma_i^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i^2}}$$

Thus, in this case the MLE is a weighted average of readings of all sensors, with weights inversely proportional to their variances. Thus, we define MLE estimator for this case as follows:

$$\hat{r} = \sum_{i=1}^{n} \frac{\frac{1}{\sigma_i^2}}{\sum_{j=1}^{n} \frac{1}{\sigma_j^2}} X_i$$

It is easy to see that such an estimator is unbiased. Moreover it is well known, in this case the MLE has the smallest (asymptotic) variance and we can say that the MLE is efficient or attains the CRLB [25].

We obtain the variance of the above estimator as follows:

[1] A similar proposition with a different proof proposed in [24]

$$\mathbb{E}\left[(\hat{r}-r)^2\right] = \mathbb{E}\left[\left(\sum_{i=1}^{n}\frac{\frac{1}{\sigma_i^2}X_i}{\sum_{j=1}^{n}\frac{1}{\sigma_j^2}} - r\right)^2\right]$$
$$= \mathbb{E}\left[\frac{1}{\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}\right)^2}\left(\sum_{i=1}^{n}\frac{X_i - r}{\sigma_i^2}\right)^2\right]$$
$$= \frac{1}{\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}\right)^2}\mathbb{E}\left[\sum_{i=1}^{n}\left(\frac{X_i - r}{\sigma_i^2}\right)^2 + 2\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{X_i - r}{\sigma_i^2}\right)\left(\frac{X_j - r}{\sigma_j^2}\right)\right]$$

We now use the fact that if $X_i$ and $X_j$ are independent then $E((X_i-\mu)(X_j-\mu)) = \mathbb{E}(X_i-\mu)\mathbb{E}(X_j-\mu) = 0\times 0 = 0$. Thus

$$E\left[(\hat{r}-r)^2\right] = \frac{1}{\left(\sum_{j=1}^{n}\frac{1}{\sigma_j^2}\right)^2}E\left[\sum_{i=1}^{n}\left(\frac{X_i - r}{\sigma_i^2}\right)^2\right]$$
$$= \frac{1}{\sum_{j=1}^{n}\frac{1}{\sigma_j^2}}$$

□

## 3.2 Definitions

The general idea of our proposed reputation system is based on an interdependency relationship among the credibility of sensors, the reputation values at time instants, and the variance of sensors. In this work, we define:

- *Credibility* of a sensor node: it reflects how much other sensor nodes support the node based on the similarity among the readings of such a sensor and the current estimate of the true value of the signal.
- *Reputation* value at a time instant: it is an estimate of the true value of signal in that particular time instant.
- *Variance* of a sensor: it is the distance of the sensor's readings from the estimate of the true value of the signal.

In this paper, the credibility of a sensor node reflects the accumulated evidence from other sensors for how well-suited the readings of such a sensor are to serve as an estimation of the true value of the signal. In other words, the credibility of a node can be measured by the mean of credibilities which such a node obtains from all other nodes. Each node gives the credibility to other nodes in accordance to the similarity between their readings for all time instants.

As we assumed the normal random distribution model for noise in sensor readings, we define the similarity between readings of a sensor node and the reputation values using the probability density function for this random distribution. Let us consider sensor node $j$ with standard deviation $\sigma_j$; this sensor

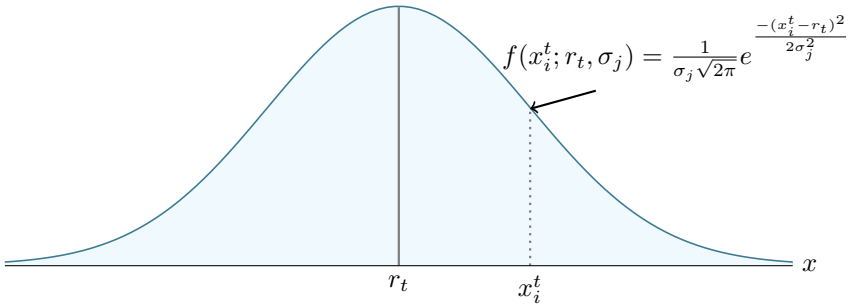



Figure 3.1: credibility of sensor $i$ given by sensor $j$ at time $t$.

node "is going to assess" the credibility of the readings of another node $i$, by estimating the (normalised) likelihood that it could have made such readings itself (see Figure Figure 3.1), i.e., the credibility that sensor $j$ confers to the readings of sensor $i$ should be equal to

$$\mathrm{L}(j,i) = \left( \prod_{t=1}^{m} \frac{1}{\sigma_j\sqrt{2\pi}} e^{\frac{-(x_i^t - r_t)^2}{2\sigma_j^2}} \right)^{\frac{1}{m}} = \frac{1}{\sigma_j\sqrt{2\pi}} e^{\frac{-\frac{1}{m}\sum_{t=1}^{m}(x_i^t - r_t)^2}{2\sigma_j^2}} \tag{3.2}$$

We now define the credibility of a sensor $i$ as the aggregate of credibilities conferred by other sensors, i.e., as normalised likelihood that all other sensors might have made readings of sensor $i$, i.e., as

$$\mathrm{cr}(i) = \left( \prod_{\substack{j=1 \\ j\neq i}}^{n} \mathrm{L}(j,i) \right)^{\frac{1}{n-1}} = \left( \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{1}{\sigma_j\sqrt{2\pi}} e^{\frac{-\frac{1}{m}\sum_{t=1}^{m}(x_i^t - r_t)^2}{2\sigma_j^2}} \right)^{\frac{1}{n-1}} \tag{3.3}$$

Equation (3.3) simply gives high credibility to nodes who report readings very close to other nodes ("the community sentiment"), because the amount of credibility of each node depends on the similarity of its readings to the current estimate of the true value of the signal. In addition, the variance of each sensor has a significant role in amount of credibility which such a node can grant to other nodes. As can be seen in Equation (3.3), the sensor nodes with lower variance value grant high credibility to sensors with readings close to the readings of that sensor.

Given the credibility value for sensor nodes by Equation (3.3), we approximate the reputation vector $\mathbf{r} = \langle r_1, r_2, \ldots, r_m \rangle$ using a weighted average, where the weight of a sensor node is the normalized value of its credibility. Thus, we compute the reputation value at time instant $t$ as follows:

$$r_t = \mathrm{rep}(t) = \sum_{i=1}^{n} \frac{\mathrm{cr}(i)}{\sum_{k=1}^{n} \mathrm{cr}(k)} x_i^t \tag{3.4}$$

where term $\sum_{k=1}^{n} \mathrm{cr}(k)$ is used to normalize the credibility values in order to make the sum of sensors weights to 1.

Table 3.1: Final credibility and variance values for sensors in the example graph in Figure 2.2.

| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
|---|---|---|---|---|---|
| Credibility | 0.270 | 0.250 | 0.335 | 0.245 | 0.275 |
| Variance | 0.945 | 1.106 | 0.493 | 1.150 | 0.913 |

Given the reputation values by Equation (3.4), we can approximate the variance of a sensor as the average squared Euclidean distance of its readings to the reputation values. Thus, we compute the variance of sensor $i$ as follows:

$$\text{var}(i) = \frac{1}{m}\sum_{t=1}^{m}(x_i^t - \text{rep}(t))^2 \tag{3.5}$$

The above formulation of credibility, reputation, and variance enable our model to provide the proposed required features of a reputation system in Section 2.4 via an iterative algorithm. Table 3.1 demonstrates the final credibility and variance values of the sensors for the example graph shown in Figure 2.2. The next section describes our iterative algorithm to compute these values; we now explain why we introduced the credibility notion.

In the classical IF algorithm using the reciprocal function the weight given to readings of a sensor $i$ when an approximation of the reputation vector is computed is the reciprocal of its estimated variance. If in any iteration of the algorithm such approximate reputation vector gets close to readings of a particular sensor, since the reciprocal has a pole at 0, the iterative procedure gives to that sensor's readings weights converging to 1 and to all other sensors weights converging to 0, which results in suboptimal performance. Such situation cannot arise in our model because the credibility of a sensor derives from both the similarity of its readings with the current reputation vector and the variance values of other sensors. Thus, all sensors get non zero weights without any regularisation, which also degrades the performance of the IF algorithm.

## 3.3 Computing Credibility and Variance

In this section, we describe an algorithm to find the credibility and variance values of all sensors in the system. Note that from equations (3.3), (3.4) and (3.5) we have mutual and transitive recursive definitions for credibility, reputation and variance concepts. Figure 3.2 shows the recursive relationship among these three concepts in our reputation model. In this figure, arrows show the dependency between the concepts. As can be seen in this figure, the credibility value of a sensor depends on both reputation values and variances of sensors; the reputation values are computed using the credibility values of the sensors; and the variance value of a sensor is measured by the distance of its readings from the reputation values for all time instants.

According to the aforementioned recursive relationship, we propose an iterative method for computing credibility, variance values of sensors, and reputation values. We denote the credibility, reputation and variance values at iteration $l$ by $\text{cr}^{(l)}(i)$, $\text{rep}^{(l)}(t)$, and $\text{var}^{(l)}(i)$, respectively. We also use the values obtained

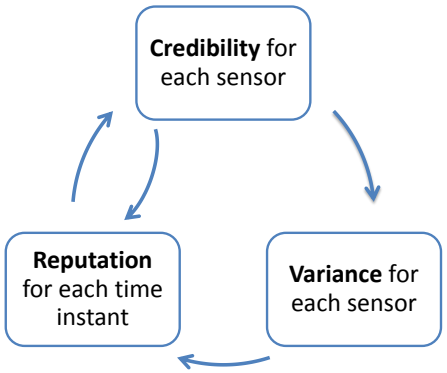



Figure 3.2: Recursive relationship among credibility, reputation and variance.

from iteration $l$ to compute the values for iteration $l+1$. In subsequent sections, we analyze how much the error decreases based on the number of iterations.

In the iterative algorithm from the initial values of variances, credibility values at the next iteration are computed for all sensors from values obtained at the previous round of iteration. Then, using the credibility values, the reputation values are re-estimated for all time instants. After that, the new values for the sensor variances are obtained by using the new reputation values. Thus, the equations (3.3), (3.4) and (3.5) can be now re-written as follows:

$$\mathrm{cr}^{(l+1)}(i) = \left( \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\sqrt{2\pi \mathrm{var}^{(l)}(j)}} e^{\frac{-\frac{1}{m}\sum_{t=1}^{m}(x_i^t - \mathrm{rep}^{(l)}(t))^2}{2\mathrm{var}^{(l)}(j)}} \right)^{\frac{1}{n-1}} \tag{3.6}$$

$$\mathrm{rep}^{(l+1)}(t) = \sum_{i=1}^{n} \frac{\mathrm{cr}^{(l+1)}(i)}{\sum_{k=1}^{n} \mathrm{cr}^{(l+1)}(k)} x_i^t \tag{3.7}$$

$$\mathrm{var}^{(l+1)}(i) = \frac{1}{m} \sum_{t=1}^{m} (x_i^t - \mathrm{rep}^{(l+1)}(t))^2 \tag{3.8}$$

In order to compute the credibility values in the first iteration, we need to initialize the variances of all sensors. We propose a similar value of variance for all sensors which is computed as the mean of total variance of all readings. Thus, we first compute the sum of variances of all sensors using Lemma 3.2. We presented a similar lemma for computing total variances of sensors when the readings are biased in [11]. Therefore, we used the similar method for proving Lemma 3.2 for unbiased sensors readings.

**Lemma 3.2** (Total Variance for Unbiased Readings)**.** *Let $\bar{x}^t$ be the mean of readings in time $t$, then, we have*

$$\bar{x}^t = \frac{1}{n} \sum_{j=1}^{n} x_j^t,$$

*and the statistic*

$$S(t) = \frac{n}{m(n-1)} \sum_{i=1}^{n} \sum_{t=1}^{m} \left( x_i^t - \bar{x}^t \right)^2$$

*is an unbiased estimator of the sum of the initial variances of all sensors,* $\sum_{i=1}^{n} v_i^{(0)}$.

*Proof.* Remind that $\mathbf{x}_i = \{x_i^t \ : \ t = 1 \cdots m\}$ is the unbiased readings of sensor $i$. Now we form the second central moment of the sum of $\mathbf{x}_i$ for all sensors as follows:

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{n}\left(\mathbf{x}_i - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j\right)^2\right] &= \frac{1}{n^2}\mathbb{E}\left[\sum_{i=1}^{n}\left(\sum_{j=1}^{n}(\mathbf{x}_i - \mathbf{x}_j)\right)^2\right] \\
&= \frac{1}{n^2}\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_k)\right] \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(\mathbb{E}[\mathbf{x}_i^2] - \mathbb{E}[\mathbf{x}_i\mathbf{x}_k] - \mathbb{E}[\mathbf{x}_i\mathbf{x}_j] + \mathbb{E}[\mathbf{x}_j\mathbf{x}_k])
\end{aligned}$$

Since the readings $\mathbf{x}_i$ are unbiased, $\mathbb{E}[\mathbf{x}_i^2]$ is equal to variance $v_i$ of sensor $i$. In addition, we assume that the sensor noise is generated by independent random variables, thus

$$\mathbb{E}[\mathbf{x}_i\mathbf{x}_j] = \begin{cases} 0 & \text{if } i \neq j\,, \\ v_i & \text{if } i = j\,. \end{cases}$$

Given the above equations, we have:

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{n}\left(\mathbf{x}_i - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j\right)^2\right] &= \frac{1}{n^2}\sum_{i=1}^{n}(n^2\,\mathbb{E}[\mathbf{x}_i^2] - n\,\mathbb{E}[\mathbf{x}_i^2]) \\
&= \sum_{i=1}^{n} v_i - \frac{1}{n}\sum_{i=1}^{N} v_i \\
&= \frac{n-1}{n}\sum_{i=1}^{n} v_i
\end{aligned}$$

Thus, we obtain

$$\sum_{i=1}^{n} v_i = \frac{n}{n-1}\mathbb{E}\left[\sum_{i=1}^{n}\left(\mathbf{x}_i - \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j\right)^2\right].$$

By approximating the expected value with the sample mean we get

$$\sum_{i=1}^{n} v_i \approx \frac{n}{m(n-1)}\sum_{t=1}^{m}\sum_{i=1}^{n}\left(x_i^t - \bar{x}^t\right)^2.$$

□

By using Lemma 3.2 and assuming equal value of variances for all sensors, the initial value of variance for all sensors is computed as follows:

$$\mathrm{var}^{(0)}(i) = \frac{1}{m \times (n-1)} \sum_{j=1}^{n} \sum_{t=1}^{m} \left( x_j^t - \frac{1}{n} \sum_{k=1}^{n} x_k^t \right)^2 \tag{3.9}$$

According to Equation (3.6), the credibility of each sensor depends on both sensors variances and reputation values. Given these identical initial values for sensors' variances, we obtain the initial reputation values as the sample mean of the sensors' readings (see Proposition 3.1):

$$\mathrm{rep}^{(0)}(t) = \frac{1}{n} \sum_{i=1}^{n} x_i^t \tag{3.10}$$

In addition, the sensors' variances are obtained from Equation (3.8) for all iterations $l > 0$. Thus, we form Equation (3.6) for computation of credibility of a sensor as follows:

$$\mathrm{cr}^{(l+1)}(i) = \left( \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{1}{\sqrt{2\pi \mathrm{var}^{(l)}(j)}} e^{\frac{\mathrm{var}^{(l)}(i)}{2\mathrm{var}^{(l)}(j)}} \right)^{\frac{1}{n-1}} \quad \text{for all} \quad l > 1. \tag{3.11}$$

Note that for the first iteration, the credibility of sensors is computed using Equation (3.6), because the initial variance is obtained from Equation (3.9). However, we employ Equation (3.11) for subsequent iterations to improve the computational complexity of our iterative algorithm which is formally investigated in Section 3.7.

Algorithm 1 shows our iterative algorithm for computing credibility and variance values for all sensors using the above equations. By computing the variance values for the sensors, we can obtain the final estimate of reputations by a MLE-like method described in the subsequent section.

Table 3.2 shows how the values of credibility, reputation and variance are updated in the first six iterations for the example graph shown in Figure 2.2.

## 3.4 Computing the Final Reputation

Given the matrix $X = \{x_i^t\}$ where $x_i^t \sim r_t + \backslash(0, \sigma_i^2)$ and the estimated variance vector $\overrightarrow{\mathrm{var}}$, we propose to recover $\mathbf{r} = \langle r_1, r_2, \ldots, r_m \rangle$ using an approximate form of the Maximum Likelihood Estimation. We suggested a similar approximation in [11].

We can now obtain an estimation which corresponds to MLE formula for the case of zero mean normally distributed errors, but with estimated rather than true variances. Therefore, we assume that the expected value $r_t$ of the measurements is the true value of the quantity measured, and is the only parameter in the likelihood function. Thus, in the expression for the likelihood function for normally distributed unbiased case. that is,

$$\mathcal{L}_n(r_t) = \prod_{i=1}^{n} \frac{1}{\sigma_i \sqrt{2\pi}} e^{\frac{-(x_i^t - r_t)^2}{2\sigma_i^2}}$$

**Algorithm 1** Credibility and variance computation.

**procedure** CredVarianceComputation($X = \{x_i^t\}$)
  Initialize $\text{var}^0(i)$ using (3.9)
  Initialize $\text{rep}^0(i)$ using (3.10)
  $l \leftarrow 0$
  **repeat**
    **for** each sensor node $i$ **do**
      **if** $l = 0$ **then**
        Compute $\text{cr}^{(l+1)}(i)$ using (3.6)
      **else**
        Compute $\text{cr}^{(l+1)}(i)$ using (3.11)
      **end if**
    **end for**
    **for** each time instant $t$ **do**
      Compute $\text{rep}^{(l+1)}(t)$ using (3.7)
    **end for**
    **for** each sensor node $i$ **do**
      Compute $\text{var}^{(l+1)}(i)$ using (3.8)
    **end for**
    $l \leftarrow l + 1$
  **until** reputations and variances have converged
  **Return** $\overrightarrow{\text{var}}$ and $\overrightarrow{\text{cr}}$
**end procedure**

Table 3.2: Credibility and variance values after each iteration for sensors in the example graph in Figure 2.2.

| $l$ | $s_1$ | | $s_2$ | | $s_3$ | | $s_4$ | | $s_5$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| | CR | VAR | CR | VAR | CR | VAR | CR | VAR | CR | VAR |
| 0 | 1 | 0.920 | 1 | 0.920 | 1 | 0.920 | 1 | 0.920 | 1 | 0.920 |
| 1 | 0.272 | 0.962 | 0.264 | 1.083 | 0.328 | 0.510 | 0.254 | 1.149 | 0.282 | 0.900 |
| 2 | 0.269 | 0.951 | 0.254 | 1.097 | 0.333 | 0.499 | 0.246 | 1.152 | 0.277 | 0.907 |
| 3 | 0.270 | 0.948 | 0.252 | 1.103 | 0.335 | 0.496 | 0.245 | 1.151 | 0.276 | 0.911 |
| 4 | 0.270 | 0.946 | 0.251 | 1.105 | 0.335 | 0.494 | 0.245 | 1.150 | 0.275 | 0.912 |
| 5 | 0.270 | 0.946 | 0.251 | 1.106 | 0.335 | 0.494 | 0.245 | 1.150 | 0.275 | 0.913 |

we replace $\sigma_i^2$ by the obtained variance $\text{var}(i)$ from Algorithm 1. Moreover, by differentiating the above formula with respect to $r_t$ and setting the derivative equal to zero we get

$$r_t = \sum_{i=1}^{n} \frac{\frac{1}{\text{var}(i)}}{\sum_{k=1}^{n} \frac{1}{\text{var}(k)}} x_i^t \quad \text{for all} \quad t = 1, \cdots, m. \tag{3.12}$$

Equation (3.12) provides an estimate of the true value of the reputations measured as a weighted average of sensors readings, with the readings given a weight inversely proportional to the estimation of their error variance provided

Table 3.3: Final sensors weights and reputation values in the example graph in Figure 2.2.

| Sensors | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ |
|---|---|---|---|---|---|
| Weights | 1.058 | 0.904 | 2.027 | 0.870 | 1.095 |
| Instants | t=1 | t=2 | t=3 | t=4 | |
| Reputations | 18.874 | 19.292 | 19.295 | 19.012 | |

by our iterative algorithm:

$$\mathbf{r} = \sum_{i=1}^{n} w_i \mathbf{x}_i \tag{3.13}$$

Table 3.3 demonstrates the final sensors weights and reputation values for the example graph shown in Figure 2.2 according to the Equation (3.13).

## 3.5 Collusion Detection and Revocation

As we described the proposed reputation system computes the credibility, variance and reputation values based on the readings of all sensor by assuming stochastic error behaviour for all sensors. Although our experiments show that this method is more robust than other IF algorithms for several different attack scenarios, the attacker can still alter considerably the reputation results of the IF algorithms (see Section 5.8). Thus, in this section we augment the reputation system with our novel collusion detection technique proposed in [26] in order to further diminish the impact of the compromised nodes. We will first describe our proposed collusion detection scheme and then discuss the proposed compromised nodes revocation approach.

### Detection Method

Upon computing the reputation values by Equation (3.13), we carry out a collusion detection and revocation method based on an analysis of the features of error distribution of the sensor nodes. In the existing approaches, compromised sensor nodes are usually detected as outliers from some form of average of all readings. Instead, we propose a finer analysis based on a sequence of sensor readings, by considering how differences between readings of the individual sensor nodes and the estimate obtained by Equation (3.13) are distributed. The main idea behind our method is that, while faulty or compromised sensors might skew the estimate, their action can only make non-compromised sensor appear biased, but the variability of such sensors around such a value will still have a distribution close to a normal distribution; on the other hand, the difference between the values provided by the compromised nodes will have highly non normal distribution, reflecting their essentially non-stochastic (colluding) behaviour. Accordingly, we assume a sensor with a non-Gaussian error distribution is likely to be a compromised node.

In order to analyse the error behaviour of sensor nodes, we first compute the sensors errors based on the distances of each sensor readings to the obtained reputation from Equation (3.13). After that, we employ a hypothesis testing method to assess the normality of the obtained error values for each sensor node.

Thus, let $\mathbf{e}_s = \{e_s^t \ : \ t = 1, \cdots, m\}$ be the vector of error terms for a sensor $s$, defined as

$$\mathbf{e}_s = \mathbf{x}_s - \mathbf{r}$$

where $\mathbf{x}_s = \{x_s^t \ : \ t = 1, \cdots, m\}$ is a sequence of readings from sensor $s$ and $\mathbf{r} = \langle r_1, r_2, \ldots, r_m \rangle$ denote the reputation values obtained from the previous phase of our framework.

The problem of deciding whether a sensor node $s$ is compromised can be formulated as a hypothesis testing problem with null and alternative hypotheses as follows:

- Null hypothesis $\mathbf{H_0}$: The sequence of errors $\mathbf{e}_s$ is drawn from a Normal distribution.
- Alternative hypothesis $\mathbf{H_1}$: The sequence of errors $\mathbf{e}_s$ is not drawn from a Normal distribution.

In order to judge the compromise sensor nodes, we employ the Kolmogorov-Smirnov test (K-S test) [25] on sample errors of each sensor. Using the estimates for the sample mean and the sample variance we normalise the errors; the Kolmogorov-Smirnov statistic then quantifies a distance between the empirical distribution of such normalised samples of sensor errors $\mathbf{e}_s$ and the $\mathcal{N}(0, 1)$ Normal distribution.

#### Revocation Method

The proposed collusion detection scheme classifies sensor nodes in two disjoint sets: the set of the compromised, and the set of the non-compromised nodes. We can now re-apply our proposed IF algorithm on non-compromised sensors readings only to produce a more accurate estimation of the true value of the signal. Thus, we will achieve new set of values for credibility and variance of non-compromised nodes as well as new reputation values. We then compute the variance of compromised nodes according to the distance of their readings to the new reputation values.

## 3.6 Basic CrPr Algorithm

In this section, we describe the basic CrPr algorithm which includes the credibility and variance computation, reputation computation, and collusion detection modules. We simply wish to compute the sensors' variances as well as reputation vector for a matrix of sensors readings.

At the most basic level, the algorithm would proceed as in Algorithm 2. As we have mentioned, our algorithm operates on batches of consecutive readings of sensors, proceeding in several stages. In the first stage, we obtain the sensors variances by employing the proposed iterative procedure for credibility and variance computation (see Section 3.3). Based on such an estimation of variance of each sensor, in the next stage of CrPr algorithm, we provide an estimation of the reputation vector calculated using a MLE-like method (see Section 3.4). In

the third stage, we apply our collusion detection and revocation method to diminish the contribution of the compromised nodes. After revoking the readings of untrusted sensors, we re-run our credibility and variance computation as well as the MLE-like method for computing the final reputation vector according to the remaining readings (stage 1 and 2). Finally, we re-compute the variance of compromised nodes according to the distance of their readings to the final reputation vector.

**Algorithm 2** Basic CrPr.

```
1: procedure CrPr(X = {x_i^t})
2:     var⃗ ← CredVarianceComputation(X)                    ▷ Section 3.3
3:     Compute r using (3.13)                               ▷ Section 3.4
4:     colluders ← ∅
5:     for each sensor node s do                            ▷ Section 3.5
6:         Compute e_s ← x_s − r
7:         Test the normality of errors e_s using K-S test
8:         if e_s is non-normal then
9:             colluders ← colluders ∪ {s}
10:        end if
11:    end for
12:    if colluders ≠ ∅ then
13:        X̂ ← X − colluders
14:        var⃗ ← CredVarianceComputation(X̂)                ▷ Section 3.3
15:        Compute r using (3.13)                           ▷ Section 3.4
16:        for each sensor node s ∈ colluders do
17:            var(s) ← (1/m) ‖x_s − r‖_2^2
18:        end for
19:    end if
20:    Return var⃗ and r
21: end procedure
```

## 3.7 Algorithmic Complexity

In order to assess the computational complexity of the basic CrPr algorithm, we must evaluate the complexity of its three main sections: Algorithm 1 which computes credibility and variance of sensors, the MLE-like method (Equation (3.13)), and the collusion detection and revocation method.

We evaluate the time complexity of Algorithm 1 in the worst case when all sensors report readings for all time instants. We must model the complexity of two main parts of the algorithm including the non-iterative computations (lines 2-3 and the first iteration) and the iterative part (lines 6-19). The complexity of the computation of initial variances, initial reputations, and all computations in first iteration can be modeled respectively by $O(n \times m)$, $O(n \times m)$, and $O(n^2 \times m)$. Thus, the complexity of computing initial values and the first iteration of the algorithm is dominated by the complexity of line 8 which is in $O(n^2 \times m)$. Although this looks expensive, for most real world datasets such as the Intel Berkeley Lab dataset [14], the number of edges is more or less linear in the number of time instants. Thus, the complexity can be reduced to $O(n \times m)$.

In order to model the complexity of the iterative part of Algorithm 1, we must model the complexity of credibility, reputation, and variance computa-

tions. Clearly, the complexity of computing credibility values for all sensors in a single iteration is $O(n^2)$. Moreover, if the worst case of the total number of readings is $n \times m$, computing the reputation and variance values requires a total $O(n \times m)$ time. Accordingly, each iteration in our algorithm requires a total $O(n^2 + n \times m) = O(n \times m)$ time, and for $k$ iterations, the total running time for the iteration part of the algorithm is $O(k \times n \times m)$. By accumulating the complexity of the two main parts of Algorithm 1, the complexity of the algorithm in the worst case can be represented by $O(k \times n \times m)$.

In the worst case, the complexity of our MLE-like method for computing the reputation vector can be simply represented by $O(n \times m)$. In addition, the complexity of the collusion detection method depends on computing the sensors errors and normality testing of the errors which require a total $O(n \times m)$. Clearly, the complexity of Algorithm 1 dominates the complexity of other two sections in Algorithm 2. Therefore, CRPR algorithm totally runs in $O(k \times n \times m)$ in the worst case when the readings matrix is dense.

It is clear that the complexity of CRPR algorithm is bounded in the convergence speed. We will show how fast our algorithm converges due to an exponential decrease on residual error in the algorithm (see Section 5.7).

## 3.8 Sparsity Pattern

In general, the structure of readings in real world datasets may be sparse. In order to consider the sparsity in sensors readings, we define an adjacency matrix $A$ as follows:

$$A = \{a_i^t\} \qquad a_i^t = \begin{cases} 1 & \text{iff sensor } i \text{ reports reading at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

By using the adjacency matrix $A$, if $a_i^t = 0$, then $x_i^t = 0$ in the readings matrix $X$. In other words, the entries must not be considered as readings but instead as missing values. Thus, we rewrite the previous equations for computing credibility, reputation, and variance by considering both the adjacency matrix and the missing values in the readings matrix.

In order to compute the credibility of a sensor in a sparse readings matrix, we must consider two points for rewriting Equation (3.3): 1) a sensor achieves credibility only for the time instants when it reported readings; 2) every two sensors give credibility to each other if and only if they reported readings for some common time instants. First, we define $T_{i,j}$ as the set of time instants which both sensors $i$ and $j$ have reported readings for those instants. In other words, $T_{i,j} = \{t \mid a_i^t = 1 \wedge a_j^t = 1\}$. Now, we rewrite Equation (3.3) by taking into account these two points as follows:

$$\mathrm{cr}(i) = \left( \prod_{\substack{j \in n_i \\ j \neq i}}^{n} \frac{1}{\sigma_j \sqrt{2\pi}} e^{\frac{-\frac{1}{m} \sum_{t=1}^{m} (x_i^t - r_t)^2}{2\sigma_j^2}} \right)^{\frac{1}{n-1}} \tag{3.14}$$

where $n_i$ is a set of all sensor nodes which have reported at least for one common time instant with sensor $i$. In other words, $n_i = \{j \mid |T_{i,j}| \geq 1\}$

By using the same technique, the equations (3.4) and (3.5) for computing the reputation vector and sensors variance become, respectively

$$\mathrm{rep}(t) = \sum_{i \in N(t)} \frac{\mathrm{cr}(i)}{\sum_{k \in N(t)} \mathrm{cr}(k)} x_i^t \tag{3.15}$$

$$\mathrm{var}(i) = \frac{1}{|N(i)|} \sum_{t \in N(i)} (x_i^t - \mathrm{rep}(t))^2 \tag{3.16}$$

where $N(i)$ indicates the *neighborhood* of node $i$ in the readings graph $G$ which is the set of vertices adjacent to $i$. Also, $|N(i)|$ is the degree of vertex $i$ which is the total number of vertices adjacent to $i$. By using matrix $A$, we have $N(i) = \{t \mid a_i^t = 1\}$ and $N(t) = \{i \mid a_i^t = 1\}$.

The iterative version of the above equations can be simply achieved by the same method which we used in Section 3.3. We can also rewrite the equation for computing the initial value of variance for sensors as follows:

$$\mathrm{var}^{(0)}(i) = \frac{1}{(n-1)} \sum_{j=1}^{n} \frac{1}{|N(j)|} \sum_{t \in N(j)} \left( x_j^t - \frac{1}{|N(t)|} \sum_{k \in N(t)} x_k^t \right)^2 \tag{3.17}$$

By using a similar method, we can rewrite the MLE-like equation (3.12) as follows:

$$r_t = \sum_{i \in N(t)} \frac{\frac{1}{\mathrm{var}(i)}}{\sum_{k \in N(t)} \frac{1}{\mathrm{var}(k)}} x_i^t \quad \text{for all} \quad t = 1, \cdots, m. \tag{3.18}$$

## 3.9 High Credibility to Sensors with More Readings

According to the credibility computation formula (3.14), the credibility of a sensor node is computed by a geometric mean of all credibilities which are given by other sensors. Although this ensures that the credibility of a sensor that has consistently received with high endorsement by other sensors will be high, there is still one possible way in which a sensor can obtain a high credibility value unjustifiably: a malicious sensor node can report only one reading very close to the community sentiment and then the sensor will obtain a high credibility value by only reporting such a reading. In this section, we refine the previous normalization method to address the issue.

In order to take into account the number of readings for computing the sensor credibility in Equation (3.14), we add a hyperbolic function to the previous simple averaging technique for the normalizing the credibility values. We choose a hyperbolic function which increases sharply for the number of readings up to the average number of readings per sensor, it rises slightly for the number of readings larger than the average and it remains steady around one when the number of readings is very close to the double of the average. By empirical experiments, we selected function $\mathrm{g}(|N(i)|)$ as follows:

$$\mathrm{g}(|N(i)|) = \tanh\left(\frac{|N(i)|}{d_\Gamma^S}\right) \qquad d_\Gamma^S = \frac{1}{n} \sum_{i=1}^{n} |N(i)|$$

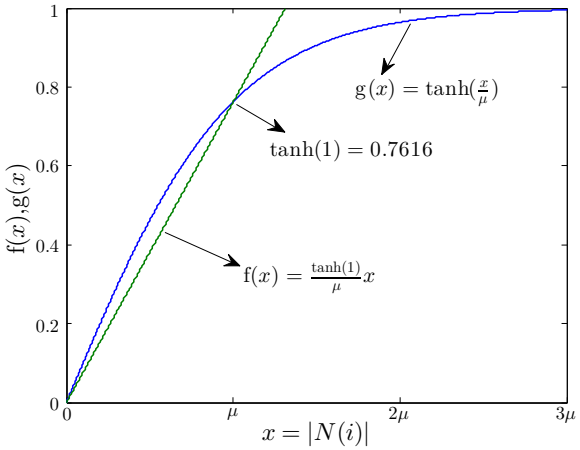



Figure 3.3: Plot of $\mathrm{g}(|N(i)|) = \tanh\left(\frac{|N(i)|}{\mu}\right)$ while $\mu = d_\Gamma^S$.

where, $N(i)$ indicates the number of readings for sensor $i$ and $d_\Gamma^S$ is the average number of readings per sensor (the average degree of a S-typed nodes in the graph representation).

Figure 3.3 shows a plot of function $\mathrm{g}(|N(i)|)$. By evaluating the Intel Lab dataset [14], we found that the number of readings for most of sensor nodes are very similar. Thus, we selected this hyperbolic function which gives a weight close to one to the average value, while it penalises the sensor nodes which their number of readings are much lower than the average value.

We can now refine the credibility computation (Equation (3.14)) by using function $\mathrm{g}(|N(i)|)$ as follows:

$$\mathrm{cr}(i) = \mathrm{g}(|N(i)|) \left( \prod_{\substack{j \in n_i \\ j \neq i}}^{n} \frac{1}{\sigma_j \sqrt{2\pi}} e^{\frac{-\frac{1}{m}\sum_{t=1}^{m}(x_i^t - r_t)^2}{2\sigma_j^2}} \right)^{\frac{1}{n-1}} \tag{3.19}$$

# 4 Credibility Propagation to Data Streaming

A large variety of WSN applications are characterized by real-time data streaming. In these applications, sensor readings continuously arrive to the reputation system. The challenge for data streaming applications is to achieve the trustworthiness of sensor nodes with low computational complexity, while swiftly adopting to the changes in the sensors behaviours.

This section aims at extending CrPr to data streaming, specifically achieving online reputation system for sensors readings. The resulting algorithm, called Str-CrPr, involves the following five main steps (Algorithm 3):

1. The first window of readings is used by CrPr algorithm to build the initial stream model for sensor behaviors (Section 4.1).

2. For each subsequent readings, the current model is used for computing the reputation value; if the value is close to the incoming readings, the stream

model is updated accordingly, otherwise the counter for the number of outliers will be incremented for appropriate sensor nodes and the reputation value will be recomputed using only non-outlier readings (Section 4.2).

3. Sensors behaviour is checked for change detection using the distribution of errors for sensor nodes (Section 4.3).

4. Upon triggering the change detection test, or if the number of outliers exceeds a threshold, the stream model is rebuilt based on the readings of the current window and the details of the previous model (Section 4.4).

5. If there are some new sensors reporting in the readings, the stream model is updated for these new sensors (Section 4.5).

The idea of making a stream model in Str-CrPr is inspired from [27, 28], albeit they used the stream models in the context of data clustering. In this algorithm, $\mathbf{x}^t = \langle x_1^t, x_2^t, \ldots, x_n^t \rangle$ indicates a set of readings reported by sensors in time $t$.

**Algorithm 3** Credibility propagation to data streaming.

```
1: procedure Str-CrPr(x^1, ..., x^w ..., x^t, ...)
2:     Str-CrPr Model ← CrPr(⟨x^1, ..., x^w⟩)              ▷ Section 4.1
3:     Window ← ⟨x^1, ..., x^w⟩
4:     t_w ← w
5:     for each time instant t > w do
6:         Update Str-CrPr Model                           ▷ Section 4.2
7:         Window ← x^t
8:         if Restart criterion then                       ▷ Section 4.3
9:             Rebuild Str-CrPr Model                      ▷ Section 4.4
10:            t_w ← t
11:        end if
12:        for each new comer sensor k in x^t do
13:            Update Str-CrPr Model for new comer k       ▷ Section 4.5
14:        end for
15:    end for
16: end procedure
```

## 4.1 Str-CrPr Model

The main idea behind our Str-CrPr algorithm is to build a statistical model of sensors' behaviours and to subsequently update the model according to the stream of readings. The readings stream model which is available at any time instant, consists of a set of 5-tuple $\langle i, v_i, t_i, E_i, O_i \rangle$, where $i$ is a sensor identifier (we assume that each sensor has a unique numeric identifier in the range of [1,n]), $v_i$ is the estimated variance of sensor $i$, $t_i$ is the number of readings reported by sensor $i$, $E_i$ is the cumulative error of sensor $i$, and $O_i$ is the number of outlier readings reported by sensor $i$.

In order to initialize the stream model, we execute the basic CrPr algorithm over the first window of sensors readings (see line 2 in Algorithm 3). As a result, we have the variance values for all sensors which have reported readings in the first window. Moreover, we set the initial values of $t_i$, $E_i$ and $O_i$ to zero for

all sensor nodes. These values will be updated according to the subsequent readings. Now, we have our initial stream model according to the first window of readings.

## 4.2 Model Update

For each new readings $\mathbf{x}^t$ reported by sensors at time $t$, $t > w$, we update the current stream model as well as compute the reputation value $r_t$. Algorithm 4 shows our update mechanism running for each new reading at time $t$. Clearly, at the first step, we estimate the reputation value $r_t$ using our MLE-like equation (3.12). Note that the equation needs the sensor variances. Thus, we use the sensor variances from the current stream model for this computation.

**Algorithm 4** Updating the stream model for incoming readings.

```
1: procedure Str-CrPrUpdate(Str-CrPr Model, x^t)
2:     Compute r_t using (3.12)
3:     for each sensor i reported readings in time t do
4:         t_i ← t_i + 1
5:         if |r_t − x_i^t| > 3σ_i then
6:             O_i ← O_i + 1
7:             x^t ← x^t − {x_i^t}
8:         end if
9:     end for
10:    Re-Compute r_t using (3.12)
11:    for each sensor i reported readings in time t do
12:        Compute E_i^t using (4.1)
13:        E_i ← E_i^t
14:    end for
15: end procedure
```

At the second step of our update procedure, given an estimate of reputation value $r_t$, we can now detect the outlier readings in $\mathbf{x}^t$. Among the many outlier detection methods proposed for WSNs [29, 30], we have selected a simple statistical test known as $3\sigma$ as it minimizes the expected detection time. However, our streaming algorithm fits well with other alternatives in outlier detection techniques. In the $3\sigma$ statistical test, all values that are more than three times the standard deviation away from the estimated reputation value are labeled as outliers. We also use the sensors variances from the current stream model for this outlier detection step.

Upon detecting the reading of sensor $i$ at time $t$ as an outlier, the stream model is updated by incrementing the number of outlier reports for the sensor node. After that, the reputation value $r_t$ will be recomputed by eliminating the detected outlier readings from $\mathbf{x}^t$. Although our MLE-like equation is robust against outlier readings, eliminating the outlier readings makes the reputation computation more accurate.

Given the final estimate of reputation value $r_t$, we can now update the cumulative error value for the sensors which have reported readings in time $t$. Let us define the error as the variance of the sensor readings from the reputation values computed by the MLE-like method from the last time of rebuilding the model be $w+1$. Thus, the cumulative error of sensor $i$ from time $w+1$ to $t$ can

be computed as follow:

$$E_i^{(t)} = \frac{1}{t-w} \sum_{k=w+1}^{t} \left(x_i^k - r_k\right)^2$$
$$= \frac{1}{t-w} \left((t-w-1) \times E_i^{(t-1)} + \left(x_i^t - r_t\right)^2\right)$$

As the error of each sensor for the time interval of the previous model building, $1 < t \leq w$, is presented by the estimated variance of sensor in the current model, we only need to compute the cumulative error for the readings reported after the time of previous model rebuilding. Moreover, both the number of readings and the cumulative error for each sensor are stored in the stream model. Thus, the above recursive equation for computing the cumulative error for sensor $i$ at time $t$ can be written in the following form:

$$E_i^{(t)} = \begin{cases} 0 & t \leq t_w, \\ \frac{1}{t_i} \left((t_i - 1) \times E_i^{(t-1)} + (x_i^t - r_t)^2\right) & t > t_w. \end{cases} \tag{4.1}$$

where $t_w$ is the time when the last model was built. Therefore, it takes $O(1)$ time to update the cumulative error for each sensor. We will investigate the time and space complexity of Str-CrPr algorithm in Section 4.6.

## 4.3 Restart Criterion

A key difficulty of our streaming algorithm is to decide when to do a rebuild process. Although a part of the stream model changes during the update process for each incoming readings, the sensors' variances must be recomputed through the rebuild process. Thus, the restart criterion plays a significant role in triggering the stream model rebuilding.

Two restart criteria have been jointly considered in Str-CrPr algorithm. The simpler option is based on the number of detected outlier readings detected during the update process; when the number of outlier readings for the majority of sensors exceeds threshold $\delta \times w$, the restart criterion is triggered. A more sophisticated restart criterion is based on detecting the statistical distribution changes in sensor variances. While quite a few change point detection methods have been proposed [31, 32], none of them works with our statistical model for sensors variances. Thus, a new approach for change detection is presented in this paper. The general idea behind our distribution change detection is based on the following observation.

**Observation 4.1.** *While the variance of a sensor has no significant changes, the new readings decrease the cumulative values of the sensor errors, because a larger number of readings makes our computation of sample variance of errors more accurate. However, upon changes to the sensor variance, the cumulative error will start to increase. This can be explained by the fact that the reputation values, which are computed according to the previous variances values, are far from the incoming sensor readings.*

Figure 4.1 illustrates the trend of the distance between the cumulative errors and variance values of sensors according to the above observation. For this

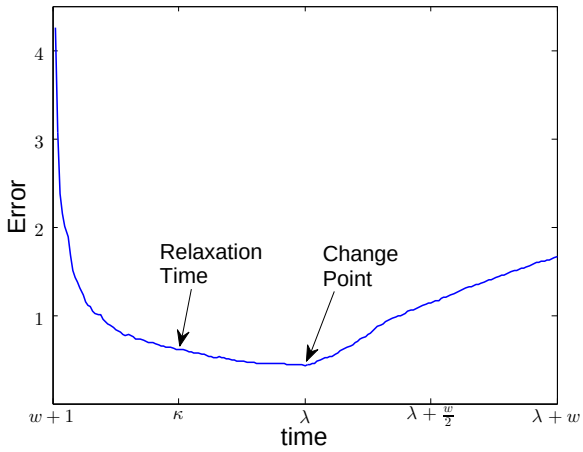



Figure 4.1: The trend of the cumulative error of sensors.

figure, we computed the cumulative error using Equation (4.1) from time $w+1$ when the computation of sensors variances for a whole of window of readings is completed. Clearly, the distance between the cumulative errors and sensor variances decreases from time $w+1$ to time $\lambda$. At the time $\lambda$, when the sensors variances are doubled, the trend of the distance increases.

In order to detect a statistical distribution change on sensors variances based on the above observation, we focus on the distance after an amount of relaxation time. The relaxation time, $\kappa$, is defined as the number of readings which are needed to compute the cumulative error of sensors in order to be statistically robust for comparing the sensors variances. We start the computation of the cumulative error after the previous rebuilding process from a whole window of readings, say time $w$. Then, we have a relaxation time frame until time $\kappa \times w$ which is used for receiving sufficient readings for robustly computing the cumulative errors. Since sensor nodes may have different variance scales, we normalize the distance between the cumulative errors and sensor variances. Thus, the *variance divergence* $d_t$ is defined as the normalized distance between the cumulative error and the sensor variance of all sensor nodes at time $t$,

$$d_t = \frac{\sqrt{\frac{\sum_{i=1}^{n}(v_i - E_i)^2}{n}}}{\sum_{j=1}^{n} v_j} \tag{4.2}$$

Now, we define a threshold $\varepsilon$ for the variance divergence $d_t$ to detect the statistical distribution changes of sensor variances. In other words, when the variance divergence $d_t$ exceeds the threshold $\varepsilon$, the restart criterion is triggered.

Note that the restart process is triggered when either the number of outlier readings for the majority of sensors exceeds threshold $\delta \times w$ or the variance divergence exceeds threshold $\varepsilon$ after the relaxation time $\kappa \times w$ from the last rebuilding process. Clearly, we define both outlier threshold $\delta$ and relaxation time $\kappa$ as a fraction of window size $w$, because the maximum values for both of the parameters is the window size. In section 5.9, we investigate the sensitivity of the effectiveness and efficiency of Str-CrPr algorithm with respect to all of these parameters.

## 4.4 Model Rebuilding

Upon triggering the restart criterion, a new stream model is rebuilt by launching CrPr algorithm over the current window of readings. Thus, the sensor variances in the stream model are updated by the results of CrPr algorithm. Moreover, the values of $t_i$, $E_i$, and $O_i$ for all sensors are reset to zero. Finally, the value of $t_w$ is set to the current time instant.

Since we are using a sliding window for storing the incoming readings, the rebuilding process will be performed over the most recent sensor readings. Moreover, we exploit the previous values of sensors variances as initial values of the variances for the rebuilding process. This idea not only makes the change of the sensors variances smoothly, but also, as our experiments show, it increases the convergence speed of our iterative algorithms.

## 4.5 Newcomer Sensor

A newcomer sensor is a sensor node that has not reported any readings during the rebuilding window time. In other words, there is no 5-tuple corresponding to a newcomer sensor node, because the first reading of such a node is reported in the time $t$, $t > w$. Clearly, existing the newcomer sensors have important impact on reputation computation, outlier detection and distribution change detection in Str-CrPr algorithm as there is no knowledge about the variance values of such sensors in the stream model.

In order to address the newcomer sensor issues, we compute the variance of such a sensor based on its cumulative error during a relaxation time interval for its readings. Moreover, the model update procedure excludes the readings of a newcomer sensor during the reputation computation until its variance is estimated. The procedure computes the cumulative error for such a newcomer sensor using Equation (4.1). Once the relaxation time interval for a newcomer sensor finishes, the stream model is updated by adding a new tuple $\langle k, v_k, t_k, E_k, O_k \rangle$, where $k$ is a unique identifier for such a newcomer sensor node; $t_k$ and $E_k$ are defined similar to other sensor nodes in the stream model; the variance of such sensor, $v_k$, is equal the cumulative error; thus $v_k = E_k$. Also, the number of outliers for the newcomer sensor is initialized to zero, thus $O_k = 0$. After adding the new 5-tuple to the stream model, Str-CrPr algorithm treats the newcomer sensor as all other nodes.

## 4.6 Memory Usage and Complexity Analysis

In this section, we study the complexity and memory usage of Str-CrPr algorithm, which are important efficiency factors for streaming algorithms. The algorithms are required to have a constant memory usage and a small computing time in the whole computation process [27]. The memory usage of Str-CrPr algorithm mainly consists of all readings in the sliding window and the 5-tuples in the stream model $\langle i, v_i, t_i, E_i, O_i \rangle$. The temporary variables employed in update process, change detection and newcomer handling consume very small. Clearly, the memory usage for keeping the sensor readings in a window with size $w$ is $O(w \times n)$. Also, the memory usage for the stream model is $O(n)$. Thus, the memory usage of Str-CrPr algorithm can be represented by $O(w \times n)$. Moreover, the memory usage remains steady in the streaming process.

The time complexity of Str-CrPr algorithm depends on the complexity of the algorithm for each time instant. At each time $t$ when a new reading arrives, the updating operation in Section 4.2 is executed which has linear complexity with respect to the number of readings in the time; thus its complexity is $O(n)$ (see Algorithm 4). The evaluation of the restart criterion in Section 4.3 calculates both the number of outlier readings and the variance divergence which takes time $O(n)$. The model rebuilding in Section 4.4 performs CrPr algorithm over the current window of readings. The complexity of this step is $O(k \times n \times w)$ in the worst case (see Section 3.7). The newcomer sensor handling procedure in Section 4.5 computes $E_k$ by simple operations like assignment, summation and multiplication on results from last time step, and thus takes a constant computing time $O(1)$. Therefore, the overall time complexity of Str-CrPr algorithm depends on the rebuilding process by applying CrPr algorithm. However, while there is no change in statistical distribution of sensors' errors, the time complexity of Str-CrPr algorithm remains in $O(n)$.

# 5 Experiments

In this section, we detail the steps taken to evaluate the robustness and efficiency of our reputation system. The objective of our experiments is to evaluate the robustness and efficiency of our approach for estimating the true value of the signal based on the sensor readings in the presence of faults and collusion attacks.

## 5.1 Experimental Environment

Although there are a number of real world datasets for evaluating reputation systems and data aggregation in sensor networks such as Intel Lab dataset [14], SensorScope [33], Great Duck Island (GUI) [34], and NAMOS [35], none of them provide a clear ground truth. Thus, we conducted our experiments by both using SensorScope as a real-world dataset and generating synthetic datasets with parameters similar to the real world dataset.

The SensorScope project is a collaboration between environmental scientists and hardware/software engineers at EPFL which aims at facilitating the adaption of WSNs as a common tool by a community with no professional knowledge in sensor networking [36]. The deployed sensor network in this project consists stations with sensor nodes that are capable of measuring 9 distinct environmental quantities: air temperature and humidity, surface temperature, incoming solar radiation, wind speed and direction, precipitation, soil water content, and soil water suction [36]. We selected the temperature measurements of a dataset collected from deploying a small sensor network at the Grand-St-Bernard pass at 2400 m between Switzerland and Italy. The dataset was collected every 2 minutes over 43 days by these station at 23 weather stations. A summary of statistical parameters of the dataset is presented in the Table 5.1.

For generating the synthetic datasets, we exploit the statistical parameters of one day readings from the SensorScope dataset. In order to model the temperature measurement in our simulation, we generate the true value of the signal by using sine function $\mathrm{f}(t) = 20 + \mathrm{Sin}\left(2\pi\frac{t}{m} - \frac{\pi}{2}\right)$. Similar to the SensorScope dataset, each sensor reports the readings every two minutes; and we collect the

readings for a day as a block for evaluating our reputation system. Thus, we set $m = 720$. In addition, we consider a zero mean Gaussian noise for sensors readings with different variance values for the sensors (see Equation (2.1)). In [11], we recently proposed a method for estimating the sensors bias. Thus, we can eliminate the bias from sensors' readings by using the same method. Moreover, similar to the SensorScope dataset, we assume that all sensor nodes report the readings for all time instants. However, we evaluate the behaviour of our algorithm for sparsity patterns in Section 5.6. If not mentioned otherwise, we generate the synthetic datasets according to the parameters listed in Table 5.2.

For each experiment which is based on synthesis datasets, we perform the algorithms over 100 different synthetically generated datasets, and then results were averaged. All the experiments have been conducted on an HP PC with 3.30GHz Intel Core i5-2500 processor with 8Gb RAM running a 64-bit Windows 7 Enterprise. The program code has been written in MATLAB R2012b.

Table 5.1: SensorScope dataset statistics.

| Parameter | SensorScope Dataset |
|---|---|
| Number of sensors | 23 |
| Number of time instants | 30,249 |
| Number of readings | 588,524 |
| Average # of readings per sensor | 25,588 |

Table 5.2: Experimental parameters for synthetic datasets.

| Parameter | Value |
|---|---|
| $n$ | 23 |
| $m$ | 720 |
| Convergence threshold | $10^{-12}$ |
| Number of repeat | 100 |
| True value of the signal | $\mathrm{f}(t) = 20 + \mathrm{Sin}\left(2\pi\frac{t}{m} - \frac{\pi}{2}\right)$ |
| The level of significance in K-S test | $\alpha = 0.05$ |

In all experiments, we compare CrPr algorithm against four other IF techniques proposed for reputation systems. For all parameters of other algorithms used in the experiments, we set the same values as used in the original papers where they were introduced.

The first IF method considered computes the trustworthiness of sensor nodes based on the distance of their readings to the current state of the estimated reputation [4]. We investigate two proposed discriminant functions $\mathrm{g}(\mathbf{d}) = \mathbf{d}^{-1}$ and $\mathrm{g}(\mathbf{d}) = 1 - k_l\mathbf{d}$ in our experiments and call these methods as *dKVD-Reciprocal* and *dKVD-Affine*, respectively.

The second IF method we consider is a correlation based ranking algorithm proposed by Zhou et al. in [3]. In this algorithm, trustworthiness of each sensor is obtained based on the correlation coefficient between the sensor readings and the current estimate of the true value of the signal. In other words, this method gives credit to sensor nodes whose readings correlate well with the estimated true value of the signal. Based on this idea, the authors proposed an iterative

Table 5.3: Summary of different IF algorithms.

| Name | Discriminant Function |
|---|---|
| dKVD-Reciprocal | $w_i^{l+1} = \left(\frac{1}{m}\left\|\mathbf{x}_i - \mathbf{r}^{l+1}\right\|_2^2\right)^{-1}$ |
| dKVD-Affine | $w_i^{l+1} = 1 - k\frac{1}{m}\left\|\mathbf{x}_i - \mathbf{r}^{l+1}\right\|_2^2$ |
| Zhou | $w_i^{l+1} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{x_i^t - \bar{\mathbf{x}}^t}{\sigma_{\mathbf{x}_i}}\right)\left(\frac{r^t - \bar{\mathbf{r}}}{\sigma_r}\right)$ |
| Laureti | $w_i^{l+1} = \left(\frac{1}{m}\left\|\mathbf{x}_i - \mathbf{r}^{l+1}\right\|_2^2\right)^{-\frac{1}{2}}$ |

algorithm for estimating the true value of the signal by applying a weighted averaging technique. They argued that correlation coefficient is a good way to quantify the similarity between two vectors. Thus, they employed Pearson correlation coefficient between sensor readings and the current state of estimate signal in order to compute the sensor weight. We call this method as *Zhou*.

The third algorithm considered has been proposed by Laureti et al. in [5] and is an IF algorithm based on a weighted averaging technique similar to *dKVD-Reciprocal* algorithm. The only difference between these two algorithms is in the discriminant function. The authors in [5] exploited discriminant function $\mathrm{g}(\mathbf{d}) = \mathbf{d}^{-\beta}$ and $\beta = 0.5$. We call this method as *Laureti*.

Table 5.3 shows a summary of aggregation and discriminant functions for all of the above four different IF methods.

We use the Root Mean Square (RMS) error as the accuracy comparison metric in all experiments which is as follows:

$$RMS\ Error = \sqrt{\frac{\sum_{t=1}^{m}(r_t - \hat{r_t})^2}{m}} \tag{5.1}$$

where $r_j$ and $\hat{r_j}$ denote the true value and the estimated value of the reputation for time instant $t$, respectively.

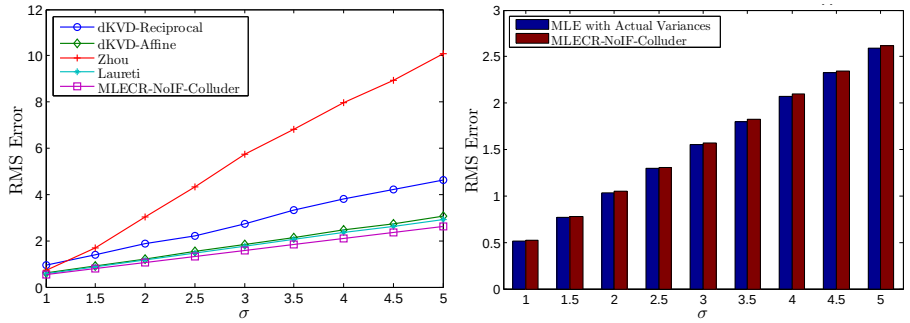
We first conduct experiments by injecting only Gaussian noise into sensor readings. In the second part of the experiments, we investigate the behaviour of these approaches by emulating a simple, non-colluding attack scenario presented in [4, 37]. We then evaluate these approaches in the case of our sophisticated attack scenario presented in [11]. For all these three scenarios, we investigate the performance of the collusion detection module in details. After that, we evaluate the algorithms for different readings' resolutions, clustering variances and various sparsity patterns. We also analyze the properties of CrPr algorithm in terms of error and convergence. Finally, we analyze Str-CrPr algorithm for parameters sensitivity and efficiency metrics.

## 5.2 Accuracy without Attacks

In the first batch of experiments we assume that there are no malicious sensor nodes. Thus, the errors are fully stochastic; we generate the errors of sensors with zero-mean Gaussian distributions. In order to evaluate the performance of CrPr algorithm in comparison with the performance of existing IF algorithms,

we consider unbiased errors with different variances for sensor nodes. We have chosen to present the case with the error of a sensor $s$ at time $t$ given by $e_s^t \sim \mathcal{N}(0, s \times \sigma^2)$, considering different values for the baseline sensor variance $\sigma^2$. Figure 5.1(a) reports the performance of CrPr algorithm for estimating the true value of the signal as well as the performance of other IF algorithms. The results show that in this experiment, the performance of our approach superior to other IF algorithms as it has a smaller RMS error.

Figure 5.1(b) reports the accuracy results of CrPr algorithm and the information theoretic limit for the minimal variance provided by the CRLB, achieved, for example, using the MLE with the *actual, exact variances* of sensors, which are NOT available to our algorithm (see Proposition 3.1). As one can see from these results, our proposed approach closely matches the minimal possible variance coming from the information theoretic lower bound.




(a) Unbiased error (b) Unbiased error

Figure 5.1: Accuracy for *No Attack* scenarios with different variances.

## 5.3 Robustness Against Simple Attacks

In order to evaluate the robustness of our algorithm against simple attack scenarios, we use two types of malicious behaviour proposed in [4, 37] over both SensorScope dataset and synthetic datasets: random readings and a promoting attack. For both of the simple attack scenarios, we selected a subset of readings from SensorScope dataset as the baseline dataset which includes the temperature values measured by all 23 sensors for a day in October 23, 2007.

For the random rating scenario, we modify 20% of the readings from the baseline dataset by injecting uniformly random real values in the range of [-5,5] for those sensors.

In slandering and promoting attacks, one or more sensor nodes falsely reports lower and higher values for their readings, respectively, about one or more time instants [2][1]. The attacks can be conducted by either an individual attacker or a coalition of attackers. The attacker can achieve the control of many sensor nodes, referred to as malicious nodes, and conduct either a slandering attack

[1]Actually, slandering and promoting attacks are defined in reputation systems for online rating applications where the attacker falsely produces negative and positive ratings for an item, respectively. We evaluate our reputation system against such attacks by injecting the false readings into a real dataset in wireless sensor networks.

(decreasing the reputation value by providing lower values for its readings) or a promoting attack (increasing the reputation value by providing higher values for its ratings) [19]. Note that the objective of this attack scenario is to skew the aggregate values through reporting outlier readings by a number of compromised nodes.

We first evaluate CrPr algorithm along other IF algorithms against both random readings and a promoting attack scenario by considering only 20% of the sensor nodes as malicious nodes involved in the scenarios. In the promoting attack scenario, malicious nodes always report the lowest temperature value (we considered -5 for experiments over SensorScope dataset) except for their preferred time instants, which they report the highest temperature value (we considered +5 for experiments over SensorScope dataset). We also assume that the malicious nodes are promoting only for 10% of all time instants. In the next experiment, we evaluate the robustness of our algorithm over similar attack scenarios by taking into account different values of variances for sensors errors as well as employing variable number of compromised nodes by generating synthetic datasets.

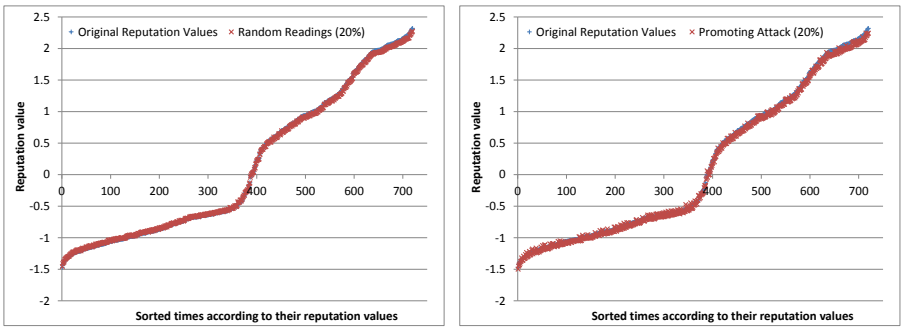
Let $\mathbf{r}$ and $\tilde{\mathbf{r}}$ be the reputation vectors before and after injecting each scenario (random readings and promoting attack), respectively. In the proposed reputation system, the vectors are the results of Equation (3.13). Table 5.4 reports the values of the 1-norm difference between these two vectors, $||\mathbf{r} - \tilde{\mathbf{r}}||_1 = \sum_{t=1}^{m} |r_t - \tilde{r}_t|$ as well as the ratio of this difference, $\frac{1}{m} ||\mathbf{r} - \tilde{\mathbf{r}}||_1$ for CrPr algorithm along with the other IF algorithms. In this table, the results of *Average* algorithm show how the attacker can skew the sample mean of readings in these two scenarios; and all of the IF algorithms are significantly more robust than *Average*. One can see that the reputations given by CrPr algorithm take less into account than other methods for both attack scenarios. In this experiment, we received instability for *Zhou* approach and we therefore eliminate its results in Table 5.4.

Figure 5.2(a) and Figure 5.2(b) show the perturbations due to the injection of the random readings and the promoting attack, respectively. As can be seen, the perturbations change only slightly when using CrPr algorithm.

Table 5.4: 1-norm absolute error between reputation vectors

| | $\|\mathbf{r} - \tilde{\mathbf{r}}\|_1 \left(\frac{1}{m} \|\mathbf{r} - \tilde{\mathbf{r}}\|_1\right)$ | | | | |
|---|---|---|---|---|---|
| | *Average* | *dKVD-Reciprocal* | *dKVD-Affine* | *Laureti* | CrPr |
| Random Readings | 142.73 (0.20) | 51.96 (0.07) | 102.29 (0.14) | 62.41 (0.09) | 14.41 (0.02) |
| Promoting Attack | 797.83 (1.11) | 76.90 (0.11) | 111.64 (0.16) | 197.09 (0.27) | 25.73 (0.04) |

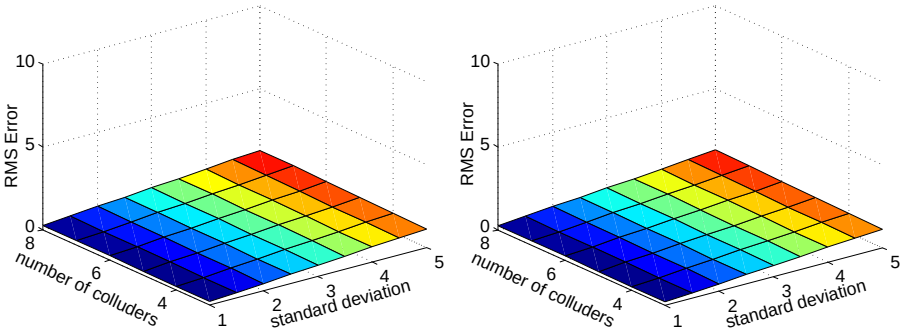
In order to evaluate the robustness of CrPr algorithm by taking into account both different values of sensors' variance and various number of compromised nodes, we assume that the attacker compromises $c$ ($c < n$) sensor nodes and reports random and outlier readings by these nodes. We also generate synthetically datasets for these experiments. Figure 5.3 shows the accuracy of CrPr algorithm in the presence of such simple attack scenarios. Comparing the RMS errors of the algorithm for these attack scenarios and the previous experiments (see Figure 5.1), it can clearly be seen that our approach achieves the accuracy of *No Attack* scenario; thus, this validates the robustness of CrPr against both the random readings and the promoting attack scenarios. In next




(a) Random Readings (b) Promoting Attack

Figure 5.2: Perturbations of reputation vector.

section, we show that this robustness is approximately stable in the case of proposed sophisticated attack scenario in [11], while other IF algorithms are significantly compromised against such an attack scenario.




(a) Random Readings (b) Promoting Attack

Figure 5.3: Accuracy of CrPr with simple attack scenarios.

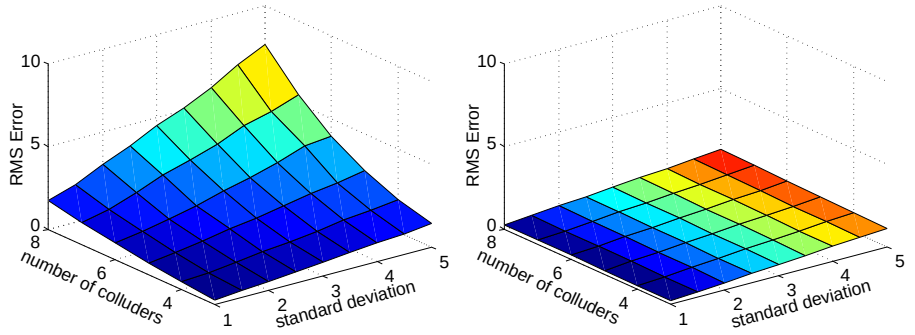
## 5.4 Robustness Against Collusion Attacks

We in [11] very recently proposed a novel sophisticated attack scenario against existing IF algorithms for trust and reputation systems when an adversary employs several malicious nodes in order to launch a collusion attack. In this attack scenario, all but one malicious node distorts the simple average of readings by reporting outlier readings, while the remaining malicious node reports a value very close to such distorted average thus making such reading appear to the IF algorithm as a highly reliable reading. As a result, the IF algorithms will converge to the value provided by the last malicious node.

In this experiment, we assume that the adversary employs $c$ ($c < n$) malicious nodes to launch the above attack scenario. The attacker uses the first $c - 1$ malicious nodes to generate outlier readings in order to skew the simple average of all readings. The adversary then falsifies the last node's reading by

injecting the values very close to such skewed average. This collusion attack scenario makes the IF algorithm to converge to a wrong stationary point. In order to investigate the accuracy of CrPr algorithm with respect to this collusion attack scenario, we synthetically generate several datasets with various numbers of compromised nodes ($c$). We in [11] applied the attack scenario against the *dKVD-Reciprocal*, *dKVD-Affine*, *Zhou* and *Laureti* approached and showed how the attacker skews the results of the IF algorithms. Moreover, since we showed that *dKVD-Affine* method is the least sensitive to the attacked scenario. Therefore in this paper, we compare the accuracy of CrPr algorithm with only *KVD-Affine* method for the experiment.

Figure 5.4 reports the accuracy of CrPr algorithm along with the accuracy of the *dKVD-Affine* method in the presence of the collusion attack scenario. It can be clearly seen that our algorithm is superior to *dKVD-Affine* algorithm in terms of accuracy against the attack scenario. Moreover, by comparing the accuracy of the algorithm in this experiment with the results from *No Attack* experiment in Figure 5.1, we can argue that our reputation system is robust against the collusion attack scenario. The reason is that our approach not only provides high accuracy against this attack, it also actually approximately reaches the accuracy of the scenarios without any false data by colluders.




(a) *dKVD-Affine* (b) CrPr

Figure 5.4: Accuracy with respect to the collusion attack.

## 5.5 Readings Resolution and Clustered Variances

Medo and Wakeling [38] investigated the effect of different scales of data over several IF algorithms and showed that the data resolution has significant impact on the performance of the IF algorithms. In order to analyze the behaviour of CrPr algorithm over the low resolution readings and different variance patterns, we perform experiments with methodology presented in [38]. We exploited an *individual* variance pattern for sensor nodes for generating synthetic datasets in the previous experiments, because the variance of each sensor $s$ was computed using $s \times \sigma^2$ for considering different values for the baseline sensor variance $\sigma^2$. In this section, we employ a clustered pattern for sensors variances by uniformly randomly selecting the variance of sensor $s$ from the distribution $U[\sigma_{min}; \sigma_{max}]$ by considering different values for $\sigma_{min}$ and $\sigma_{max}$.

For these experiments, we created synthetic datasets with parameters similar

to the parameters in Table 5.2. The scale of the true value of the signal is in the range of $R = [5, 50]$. Thus, for each value of $R$, we generate the true value of the signal by using sine function $\mathrm{f}(t) = R + 5 \times \mathrm{Sin}\left(2\pi\frac{t}{m} - \frac{\pi}{2}\right)$. Also, we consider a zero-mean Gaussian noise for readings of each sensor $i$ with standard deviation $\sigma_i$ which is randomly selected by a uniform distribution $U[\sigma_{min}; \sigma_{max}]$, where $\sigma_{min}$ and $\sigma_{max}$ are real values which will be defined for each experiment.

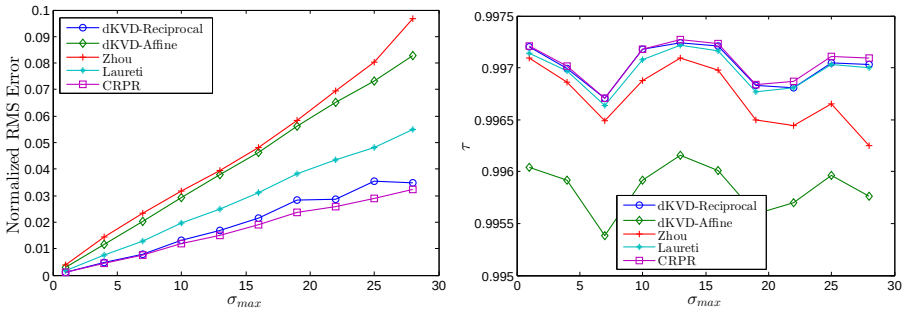
In order to investigate the accuracy of our reputation systems, we evaluate a normalized RMS error, $RMS/(R-1)$ (see (5.1) for RMS Error) for each experiment. Moreover, we consider Pearson correlation coefficient [25] to investigate the performance of our algorithm for ranking of sensors according to $\sigma_i^2$ with the estimated variance given by $\mathrm{var}(i)$. The correlation coefficient, denoted by $\tau$, is a measure of the strength of the linear relationship between the true and estimated sensor variances and takes on values ranging between +1 and -1. A value of $\tau = 0$ indicates no linear relationship, +1 indicates a perfect positive linear relationship, and -1 indicates a perfect negative linear relationship. We choose the correlation coefficient metric, because our objective is compare our results with the experimental results in [38].

For the first experiment, we keep a constant reading resolution $R = 30$, a constant value of $\sigma_{min} = 0$, and various values of $\sigma_{max}$ in the range of $[1, 29]$. By choosing the range and at the worst case, a highest noisy sensor with $\sigma_i = \sigma_{max} = 29$ could potentially report a very low temperature for the highest temperature environment circumstance, and vice versa. Figure 5.5 shows the performance of CrPr algorithm along with the performance of the other IF algorithms for this experiment. It can be clearly seen that our reputation method is less sensitive to the increasing amount of sensor noise by providing less RMS error for estimating reputation values as well as more accurate sensors variances based on the value of Pearson's $\tau$.

By comparing the results of this experiment with the experiments in Section 5.2, we found that when the sensors variances are not clustered, the IF algorithms, particularly *dKVD-Reciprocal* and *Laureti*, are able to achieve significantly higher accuracy than clustered variance values for sensors. However, CrPr algorithm generates a very high accuracy for estimating reputation vector and sensor variances in both variance patterns (individual and clustered). Moreover, we observe a strong relationship between two performance metric, RMS error and $\tau$ for three algorithms *dKVD-Reciprocal*, *Laureti* and CrPr. These results can be explained by the fact that these three algorithms place harsh sanction against the high noisy sensors when they compute the sensor weights (see Table 5.3).

In order to investigate the effect of changing the readings' resolution, we performed simulations where the sensor errors was fixed in proportion to the width of the readings scale, and varied the value of $R$ in the range of $[5, 50]$ while generating the readings by the previous equation for $\mathrm{f}(t)$. We also set $\sigma_{min} = \sigma_{max}/8$ and $\sigma_{max} = R - 1$, so that the maximum possible sensor errors cover the readings' scale. Figure 5.6 shows the performance of CrPr algorithm along with the performance of the other IF algorithms for this experiment. In this experiment, algorithm *Zhou* experienced very high RMS errors as well as very low correlation values between the estimated and real sensors variances by increasing the resolution scales. We did not display the results of this algorithm to make the graphs more clear.

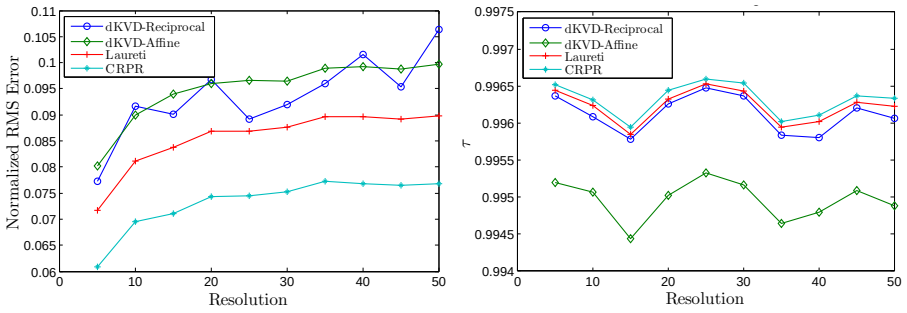
As we can see in Figure 5.6, CrPr algorithm is superior to the other IF




(a) (b)

Figure 5.5: Accuracy with clustered variances and a constant readings scale.

algorithms as it has smaller RMS error and higher correlation value for its variance estimation. Medo and Wakeling [38] reported that *Laureti* algorithm is the least sensitive algorithm when the resolution scale changes. We compare the trend of the RMS error and correlation value in Figure 5.6. The comparison shows that this experiment not only validates their results, but it demonstrates that CrPr algorithm exactly achieves the flexibility of *Laureti* while at the same time achieving the lowest RMS error and the best sensors variances estimation.




(a) (b)

Figure 5.6: Accuracy with different readings resolutions.

## 5.6 Sparsity Pattern in Sensor Readings

In Section 3.8, we described how CrPr algorithm supports the sparsity pattern in sensors readings. We also proposed a method to give higher credibility to the sensors with higher number of readings in Section 3.9. In order to evaluate the performance of CrPr algorithm over sparse readings, we define a density factor $0 < \eta \leq 1$, which is the proportion of number of readings for each sensor. Clearly, a value of $\eta = 1$ indicates no sparsity pattern.

For the experiments in this section, we selected a subset of readings from the SensorScope dataset as the baseline dataset which included the temperature values measured by all 23 sensors throughout of a day, October 23, 2007. In this experiment, we consider various values for density factor, $\eta$, in the range

of $[0.5, 1]$. Moreover, for injecting the sparsity pattern based on each value of the density factor, we change the original SensorScope dataset by uniformly randomly removing $m \times (1 - \eta)$ readings for each sensor.

Let $\mathbf{r}$ and $\tilde{\mathbf{r}}$ be the reputation vectors before and after injecting the sparsity patterns. Table 5.5 reports the values of the 1-norm difference between these two vectors, $||\mathbf{r} - \tilde{\mathbf{r}}||_1 = \sum_{t=1}^{m} |r_t - \tilde{r}_t|$ as well as the ratio of this difference, $\frac{1}{m} ||\mathbf{r} - \tilde{\mathbf{r}}||_1$ for CrPr algorithm along with the other IF algorithms. Clearly, the experiment results show that, increasing the density factor improves the accuracy of all the IF algorithms. This can be explained by the fact that all of these algorithms are using a kind of collaborative technique among sensor nodes for estimating the reputation values as well as sensors trustworthiness; and the density of the sensors readings has a significant effect in the performance of each collaborative method [39].

Table 5.5: 1-norm absolute error between reputation vectors

| | $\|\mathbf{r} - \tilde{\mathbf{r}}\|_1$ $\left(\frac{1}{m} ||\mathbf{r} - \tilde{\mathbf{r}}||_1\right)$ | | | | |
|---|---|---|---|---|---|
| | *Average* | *dKVD-Reciprocal* | *dKVD-Affine* | *Laureti* | CrPr |
| $\eta = 0.5$ | 130.47 (0.18) | 99.82 (0.14) | 105.24 (0.15) | 101.64 (0.14) | 118.66 (0.16) |
| $\eta = 0.6$ | 105.79 (0.15) | 77.05 (0.11) | 84.87 (0.12) | 82.17 (0.11) | 93.47 (0.13) |
| $\eta = 0.7$ | 84.24 (0.12) | 60.19 (0.08) | 67.12 (0.09) | 65.24 (0.09) | 72.24 (0.10) |
| $\eta = 0.8$ | 63.45 (0.09) | 45.01 (0.06) | 50.28 (0.07) | 49.09 (0.07) | 52.40 (0.07) |
| $\eta = 0.9$ | 40.27 (0.06) | 28.60 (0.04) | 32.16 (0.04) | 31.50 (0.04) | 32.43 (0.05) |

## 5.7 Analysis of Error and Convergence

In this section, we perform a set of experiments to analyze the properties of our iterative algorithm in terms of error and convergence. Thus, we investigate two types of errors for both credibility and variance values computed in each iteration of CrPr algorithm over the SensorScope dataset. For each of credibility and variance values, we define the maximum error by choosing the worst-case error for all sensor nodes. Therefore, the maximum errors at iteration $l$ is computed as follows:

$$error_{cr}^{(l)} = \max_i \left| \mathrm{cr}^{(\infty)}(i) - \mathrm{cr}^{(l)}(i) \right|$$

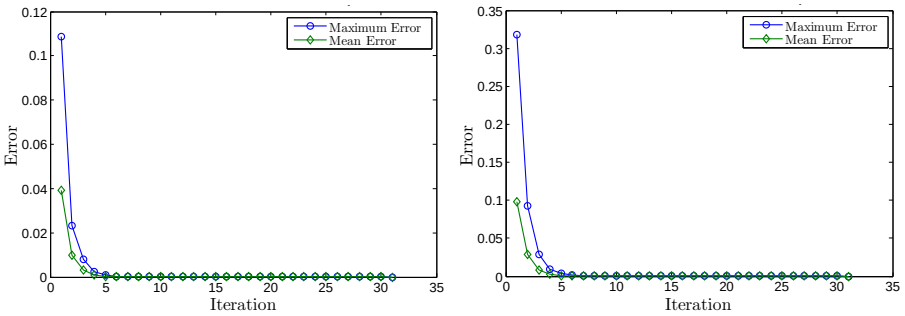
$$error_{var}^{(l)} = \max_i \left| \mathrm{var}^{(\infty)}(i) - \mathrm{var}^{(l)}(i) \right|$$

We also define the mean error of credibility and variance values over all sensors as follows:

$$error_{cr}^{(l)} = \frac{1}{n} \sum_{i=1}^{n} \left| \mathrm{cr}^{(\infty)}(i) - \mathrm{cr}^{(l)}(i) \right|$$

$$error_{var}^{(l)} = \frac{1}{n} \sum_{i=1}^{n} \left| \mathrm{var}^{(\infty)}(i) - \mathrm{var}^{(l)}(i) \right|$$

Figure 5.7 illustrates how the aforementioned errors decline for both credibility and variance values. Clearly, the algorithm has converged after 31 iterations. For all experiments, we set convergence threshold with an error $\left\|\mathrm{var}^{(l+1)} - \mathrm{var}^{(l)}\right\|_2$ less than $10^{-12}$.




(a) Credibility (b) Variance

Figure 5.7: Convergence and error of CrPr algorithm.

Table 5.6: Confusion matrix for collusion detection.

| Actual Class | Predicted Class | |
|---|---|---|
| | Colluder | Benign |
| Colluder | True Positive (TP) | False Negative (FN) |
| Benign | False Positive (FP) | True Negative (TN) |

## 5.8 Collusion Detection Performance

As we described, an important module of our reputation framework is a collusion detection system, which is a binary classification technique for classifying the sensor nodes in two groups: compromised and non-compromised nodes. Based on the results of this collusion detection system, we eliminate the contributions of detected compromised nodes and then re-run the credibility computation phases of our framework in order to obtain the final reputation based on only the readings of non-compromised sensor nodes. Therefore, the performance of the collusion detection system has a significant role in improving the accuracy of the proposed reputation system.

The detection performance of the module is evaluated by its accuracy, precision, and recall measurements for each experimental scenario. A higher value shows that the collusion detection module is superior. The accuracy is the proportion of the total number of predictions that were correct; the recall or true positive rate is the proportion of colluders that were correctly detected; precision is the proportion of the detected colluders that were correct. Accuracy, precision and recall measurements are calculated based on a confusion matrix in Table 5.6 as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \times 100 \tag{5.3}$$

$$Recall = \frac{TP}{TP + FN} \times 100 \tag{5.4}$$

Table 5.7: Performance of our collusion detection module.

| Metric | No Attack | Random Readings | Promoting Attack | Collusion Attack |
|---|---|---|---|---|
| Accuracy | 95.27 | 99.19 | 99.22 | 98.71 |
| Precision | 0 | 96.24 | 96.44 | 96.35 |
| Recall | 0 | 100 | 100 | 98.32 |

For all experiments described in previous sections, we obtained the confusion matrix as well as the accuracy, precision and recall measurements for collusion detection module. Table 5.7 shows the performance results of the collusion detection module for the previous scenarios based on the average values of three metrics accuracy, precision and recall for each experiment. The table shows that the collusion detection mechanism not only is able to successfully detect the compromised nodes with high accuracy, precision and recall values in the attack scenarios, but it provides an acceptable accuracy for *No Attack* scenarios. Thus, applying this mechanism on completely clean readings has no impact on the performance of reputation computation process. Note that we can only investigate the accuracy metric for *No Attack* scenarios, because there is no compromised node for them and therefore $TP = 0$. Consequently, the precision and recall measurements are zero for all the cases in the scenarios.

## 5.9 Str-CrPr Performance

This section reports the empirical evaluation of effectiveness and efficiency of Str-CrPr algorithm. Let us first present the experimental settings before discussing the results.

### Experimental Settings

Our experiments for evaluating the performance of Str-CrPr algorithm have been conducted on a subset of the SensorScope dataset. This subset includes the temperature values measured by all 23 sensors throughout 30 days, from September 26, 2007 to October 25, 2007. We selected this duration because some sensors did not reported data for first days. However, Str-CrPr algorithm needed enough readings from all sensor nodes to build a comprehensive streaming model. We assumed that the data keeps streaming into the algorithm. The 30-day streaming data consists of about 467,000 reported data. Similar to section 5.3, we compiled the original published dataset and then created a matrix form of readings from the dataset. Consequently, the matrix form of the readings consists of 21,509 time instants with density factor of 94.07%. We applied Str-CrPr algorithm on the streaming dataset to estimate the aggregate value for each time instant.

### Effectiveness

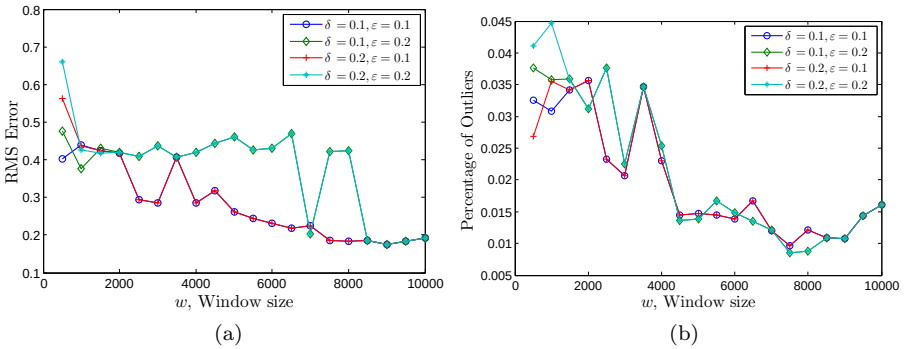
The effectiveness of Str-CrPr algorithm depends on the accuracy of the algorithm which can be measured by two criteria: RMS error of the aggregate values and the *percentage of outliers*. The RMS error represents the accuracy of the algorithm for estimating the true value of the signal. Since there is no

clear ground truth for the SensorScope dataset, we first execute CrPr algorithm over the entire dataset and obtain an estimate of the true value of the signal. This estimate is then considered as a ground truth for computing the RMS error of Str-CrPr algorithm with respect to different parameters values. In other words, in this section, the RMS error criteria is the distance between the reputation values from Str-CrPr and CrPr algorithms. Note that, the results of these two algorithms are identical when the windows size for the latter algorithm is equal to the number of time instants in the dataset (value of $m$).

The second criterion, the *percentage of outliers*, evaluates the time effectiveness of a data streaming model [27]. As we discussed, outliers may be caused by either faults in the sensor nodes or change in the stream model. Also, the contribution of outliers are eliminated from reputation computations. Thus, this makes it desirable to keep the number of outliers as low as possible. Since the number of outliers is reset after each model rebuilding, we keep updating the total number of outliers during each model rebuilding. Thus, the percentage of outlier can be obtained as follows:

$$percentage\ of\ outliers = \frac{\sum_{models} \sum_{i=1}^{n} O_i}{Total\ \#\ of\ readings} \tag{5.5}$$

Figure 5.8 compares the effectiveness of Str-CrPr algorithm on the SensorScope dataset depending on the window length $w$. Figure 5.8(a) shows that Str-CrPr algorithm achieves high accuracy with an RMS error less than 0.2 for $w > 7500$. Moreover, the least percentage of outliers is achieved by the similar window length (around 0.01%), while the highest outlier percentage is only 0.045% which is achieved by the smallest window length ($w = 500$), as shown in Figure 5.8(b). Interestingly, the results show that the lower variance divergence threshold ($\varepsilon = 0.1$) provides high accuracy with respect to both criterion, the RMS error and the outliers percentage. This can be explained by the fact that the lower value of this threshold raises the sensitivity of the model restart criterion, and therefore increases the number of model rebuilding processes. Our evaluation results reported in the next section, where we analyze the number of model rebuilding for the same experiments, validates this explanation.




(a) (b)

Figure 5.8: Accuracy of Str-CrPr depending on window length $w$.

Figure 5.9 displays the influence of the restart parameters on the accuracy of Str-CrPr algorithm with respect to the percentage of outliers. For both of these experiments, we set the window size $w = 720$ which is the maximum

number of readings per sensor during a day. We can see that the algorithm has lower percentage of outliers when both of restart parameters are less than 0.2. However, larger values for the parameters increase the number of outliers, because they decrease the sensitivity of the algorithm for model rebuilding.

Moreover, the percentage of outliers remains steady for large values of the variance divergence threshold ($\varepsilon > 0.2$) with respect to both values of outlier threshold, as shown in Figure 5.9(b). The reason is that large values of $\varepsilon$ diminish the contribution of the variance divergence in the restart criterion. Consequently, the outlier threshold is the only criteria for making a decision about model rebuilding. This also verifies the high distance between the outlier percentage values appropriate to two different values for outliers threshold $\delta$. Likewise, the large values of outliers threshold ($\delta > 0.2$) eliminate its contribution in the restart criterion when we set a tough variance divergence threshold ($\varepsilon = 0.1$) as shown in Figure 5.9(a). However, the larger value of the variance divergence threshold increases the role of outlier threshold for making a decision about model rebuilding.

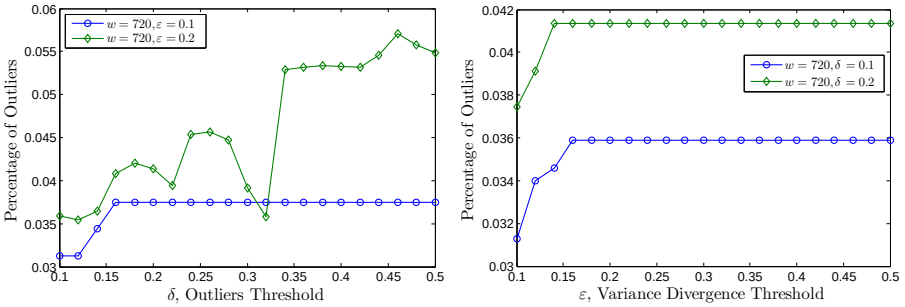



Figure 5.9: Accuracy of Str-CrPr depending on restart parameters $\delta$ and $\varepsilon$.

### Efficiency

We presented the efficiency of Str-CrPr algorithm by formulating its memory usage and time complexity in section 4.6. We showed that the memory usage of the algorithm only depends on the number of sensors and the window size. However, the main parameter which influences the processing time of the algorithm is the number of model rebuilding. In this section, we assess the efficiency of Str-CrPr algorithm by analysing the processing time and the number of model rebuilding.

Table 5.8 gives the elapsed time (in seconds)[2] and the number of model rebuilding of Str-CrPr on various parameter settings, focusing on the most representative results. As one can see, the number of model rebuilding significantly declines by increasing the threshold of variance divergence. This can be explained by a very small percentage of outliers presented by Str-CrPr in the previous experiment, as shown in Figure 5.8(b). In other words, the restart criterion mostly depends on the variance divergence of sensor nodes. Moreover,

[2]The elapsed time measured by tic/toc functions in MATLAB R2012b.

Table 5.8: Elapsed time (in seconds)/number of model rebuilding of Str-CrPr.

| Restart | $\delta = 0.1$ $\varepsilon = 0.1$ | $\delta = 0.1$ $\varepsilon = 0.2$ | $\delta = 0.2$ $\varepsilon = 0.1$ | $\delta = 0.2$ $\varepsilon = 0.2$ |
|---|---|---|---|---|
| $w = 1000$ | 3.72/10 | 2.79/4 | 3.7/9 | 2.61/2 |
| $w = 2000$ | 3.12/4 | 3.11/4 | 3.13/4 | 3.11/4 |
| $w = 3000$ | 4.87/11 | 3.6/6 | 5/11 | 3.61/6 |
| $w = 4000$ | 4.55/8 | 2.99/3 | 4.54/8 | 2.98/3 |
| $w = 5000$ | 4.32/6 | 3.6/4 | 4.35/6 | 3.58/4 |
| $w = 6000$ | 4.76/5 | 3.06/2 | 4.74/5 | 3.05/2 |
| $w = 7000$ | 5.47/5 | 4.8/4 | 5.48/5 | 4.79/4 |
| $w = 8000$ | 4.92/4 | 3.13/2 | 4.92/4 | 3.13/2 |
| $w = 9000$ | 4.38/3 | 4.39/3 | 4.39/3 | 4.38/3 |
| $w = 10000$ | 4.61/3 | 4.61/3 | 4.62/3 | 4.61/3 |

there is a general downward trend in the number of rebuildings when increasing the window size, although the algorithm experiences some fluctuation which may be caused by various changes in the behaviour of sensor errors in the SensorScope dataset. We also can see a steady trend for number of rebuildings on all different values of restart parameters when $w > 9000$. The reason is explained by the results in Figure 5.8(b), where we observe that the percentage of outliers remains steady for these window size values.

Table 5.8 also shows that the computational cost (elapsed time) increases with both window size and number of model rebuildings. Interestingly, the trend of the elapsed time fluctuates for large values of window size (around $w > 5000$). For example, the elapsed time of the algorithm for $w = 8000$ with the first restart parameter settings (first column in the table) is higher than the elapsed time observed by same experiment with $w = 7000$ while the number of rebuildings for the former experiment is less. This high computation cost is due to the sensitivity of CrPr algorithm to large windows (value of $m$) as we have shown by formally analyzing the time complexity of the algorithm in Section 3.7.

# 6 Related Work

Trust and reputation systems play critical role in WSNs as a method of resolving a number of important problems, such as secure routing, fault tolerance, false data detection, compromised node detection, secure data aggregation, cluster head election, outlier detection, etc [40]. There are three areas of work related to our research: IF algorithms, trust and reputation systems for WSNs, and secure data aggregation with compromised node detection in WSNs.

Several papers have proposed IF algorithms for trust and reputation systems [3, 4, 5, 6, 7, 8, 9, 10]. DeKerchove and Dooren in [4] proposed an IF algorithm for computing reputation of objects and raters in a rating system. They introduced both reciprocal and affine discriminant functions. We investigated the performance of their methods in Section 5. The primary idea of the algorithm proposed in [3] is to gives high credit to users whose ratings correlate nicely with the estimated true ratings of objects. Laureti et al. in [5] proposed an

IF algorithm based on a weighted averaging technique where the weights are computed through a simple reciprocal discriminant function. Li et al. in [7] proposed six different algorithms, which are all iterative and are very similar. The only difference among the algorithms is their choice of norm and aggregation function. Ayday et al. proposed a slightly different iterative algorithm in [8]. Their main differences from other algorithms are: 1) the ratings have a time-discount factor, so in time, their importance will fade out; and 2) the algorithm maintains a black-list of users who are especially bad raters. Liao et al. in [9] proposed an iterative algorithm which beyond simply using the rating matrix, also uses the social network of users. The main objective of author in [10] is to introduce a "Bias-smoothed tensor model", which is a Bayesian model, of rather high complexity. Medo et al. in [38] performed a comparative evaluation of several IF algorithms with different scales of discrete ratings and user variances. They also showed that where the rating resolution is low, increased noise in user's ratings improves the overall performance of the algorithms. Galletti et al. in [13] provided a mathematical framework to model collaborative reputation systems. They also proposed sufficient conditions for convergence of the systems using basic results of the fixed point theory. Alfaro and Shavlovsky in [24] very recently proposed *CrowdGrader*, a tool that lets students submit and collaboratively grade solutions to homework assignments. They employed an IF algorithm to estimates the consensus grades and of the grading accuracy of each student, which is very similar to the IF algorithm proposed in [4] with reciprocal discriminant function. Fouss et al. in [15] proposed a few reputation models based on a simple consumer-provider interaction model. The main idea of the proposed reputations frameworks is based on using Expectation-Maximization algorithm for obtaining the maximum likelihood of the parameters of a probabilistic model for ratings with considering the observed and unobserved values. Such IF algorithms consider simple cheating behaviour by adversaries. However, none of them take into account sophisticated malicious scenarios such as collusion attacks. For example, Rezvani et al. [11] proposed a collusion attack against existing IF algorithms and showed that most of existing IF algorithms are vulnerable against the proposed attack [11]. We showed that our approach is robust against this collusion attack.

Our work is also closely related to the trust and reputation systems in WSNs. Authors in [41] proposed a general reputation framework for sensor networks in which each node develops a reputation estimation for other nodes by observing its neighbors which make a trust community for sensor nodes in the network. Xiao et al. [42] proposed a trust based framework which employs correlation to detect faulty readings. Moreover, they introduced a ranking framework to associate a level of trustworthiness with each sensor node based on the number of neighboring sensor nodes are supporting the sensor. Li et al. [43] proposed PRESTO, a model-driven predictive data management architecture for hierarchical sensor networks. PRESTO is a two tier framework for sensor data management in sensor networks. The main idea of this framework is to consider a number of proxy nodes for managing sensed data from sensor nodes. Lim et al. [44] proposed a cyclic framework based on an interdependency relationship between network nodes and data items for assessing their trust scores based. Sun et al. [45] proposed a combination of trust mechanism, data aggregation, and fault tolerance to enhance data trustworthiness in Wireless Multimedia Sensor Networks (WMSNs) which considers both discrete and continuous data

streams. Tang et al. [46] proposed a trust framework for sensor networks in Cyber Physical System (CPS). An example of deployment of sensors in CPS is a battle-network system in which the sensor nodes are employed to detect approaching enemies and send alarms to a command center. Although fault detection problems have been addressed by applying trust and reputation systems in the above research, none of them take into account sophisticated malicious scenarios such as collusion attacks in adversarial environments.

Reputation and trust concepts can be used to address the compromised node detection and secure data aggregation problems in WSNs. Alzaid [47] proposed a secure aggregation scheme to address bad mouthing, ballot stuffing, replay and newcomer attacks; however the scheme is limited to detecting the On/Off attack launched from only one child cell. Ho et al. [48] proposed a framework to detect compromised sensor nodes in WSN and then apply a software attestation for the detected nodes. They reported that the revocation of detected compromised nodes can not be performed due to a high risk of false positive in the proposed scheme. The main idea of false aggregator detection in the scheme proposed in [49] is to employ a number of monitoring nodes which are running aggregation operations and providing a MAC value of their aggregation results as a part of MAC in the value computed by the cluster aggregator. High computation and transmission cost required for MAC-based integrity checking in this scheme makes it unsuitable for deployment in WSN. Lim et al. [37] proposed a game-theoretical defense strategy to protect sensor nodes and to guarantee a high level of trustworthiness for sensed data. Moreover, a number of approaches have been proposed in the area of secure tiny aggregation in WSNs [20, 21, 22]. These studies focus on detecting false aggregation operations by an adversary, that is, when data aggregator nodes obtain data from source nodes and produce wrong aggregated values. Such approaches do not address neither the problem of false data being provided by the data sources nor the problem of collusion. However, when an adversary injects false data by a collusion attack scenario, it can affects the results of the honest aggregators and thus the base station will receive skewed aggregate value. In this case, the compromised nodes will attest their false data and consequently the base station assumes that all reports are from honest sensor nodes. Although the aforementioned research takes into account false data injection for a number of simple attack scenarios, to the best of our knowledge, no existing work addresses this issue in the case of a sophisticated attack of colluding adversaries compromising a number of nodes in a manner which employs high level knowledge about data aggregation algorithm used.

# 7 Conclusions

In this paper, we introduced a novel collaborative reputation system which not only perform accurately in the presence of different types of faults and simple attacks such as random ratings and promoting attack, but also is robust against the sophisticated collusion attacks which most of the existing IF algorithms are vulnerable.

# Bibliography


[1] Audun Jøsang and Jennifer Golbeck. Challenges for robust trust and reputation systems. In *Proceedings of the 5 th International Workshop on Security and Trust Management*, Saint Malo, France, 2009.

[2] Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42(1):1:1–1:31, December 2009.

[3] Yan-Bo Zhou, Ting Lei, and Tao Zhou. A robust ranking algorithm to spamming. *EPL (Europhysics Letters)*, 94(4):48002–48007.

[4] Cristobald de Kerchove and Paul Van Dooren. Iterative filtering in reputation systems. *SIAM J. Matrix Anal. Appl.*, 31(4):1812–1834, 2010.

[5] P. Laureti, L. Moret, Y.-C. Zhang, and Y.-K. Yu. Information filtering via Iterative Refinement. *EPL (Europhysics Letters)*, 75:1006–1012, September 2006.

[6] Y.-K. Yu, Y.-C. Zhang, P. Laureti, and L. Moret. Decoding information from noisy, redundant, and intentionally distorted sources. *Physica A Statistical Mechanics and its Applications*, 371:732–744, November 2006.

[7] Rong-Hua Li, Jeffrey Xu Yu, Xin Huang, and Hong Cheng. Robust reputation-based ranking on bipartite rating networks. In *SDM'12*, pages 612–623, 2012.

[8] Erman Ayday, Hanseung Lee, and Faramarz Fekri. An iterative algorithm for trust and reputation management. In *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory - Volume 3*, ISIT'09, pages 2051–2055, 2009.

[9] H. Liao, G. Cimini, and M. Medo. Measuring quality, reputation and trust in online communities. *ArXiv e-prints*, August 2012.

[10] Bee-Chung Chen, Jian Guo, Belle Tseng, and Jie Yang. User reputation in a comment rating environment. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 159–167, 2011.

[11] Mohsen Rezvani, Aleksandar Ignjatovic, Elisa Bertino, and Sanjay Jha. Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks, 2014.

[12] David Wagner. Resilient aggregation in sensor networks. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, SASN '04, pages 78–87, New York, NY, USA, 2004. ACM.

[13] A. Galletti, G. Giunta, and G. Schmid. A mathematical model of collaborative reputation systems. *Int. J. Comput. Math.*, 89(17):2315–2332, November 2012.

[14] The Intel lab dataset. *available at:* `http://berkeley.intel-research.net/labdata/`, 2004.

[15] François Fouss, Youssef Achbany, and Marco Saerens. A probabilistic reputation model based on transaction ratings. *Inf. Sci.*, 180(11):2095–2123, June 2010.

[16] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-rotaru, and Herbert Rubens. Mitigating byzantine attacks in ad hoc wireless networks. Technical report, Department of Computer Science, Johns Hopkins University, Tech, 2004.

[17] Alvaro Cardenas, Saurabh Amin, Bruno Sinopoli, Annarita Giani, Adrian Perrig, and S. Shankar Sastry. Challenges for securing cyber physical systems. In *Workshop on Future Directions in Cyber-physical Systems Security*. DHS, July 2009.

[18] Pedram Radmand, Alex Talevski, Stig Petersen, and Simon Carlsen. Taxonomy of wireless sensor network cyber security attacks in the oil and gas industries. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, AINA '10, pages 949–957, Washington, DC, USA, 2010. IEEE Computer Society.

[19] Yan Sun and Yuhong Liu. Security of online reputation systems: The evolution of attacks and defenses. *Signal Processing Magazine, IEEE*, 29(2):87 –97, march 2012.

[20] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 278–287, New York, NY, USA, 2006. ACM.

[21] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. SDAP: a secure hop-by-hop data aggregation protocol for sensor networks. In *MobiHoc*, pages 356–367, 2006.

[22] Sankardas Roy, Mauro Conti, Sanjeev Setia, , and Sushil Jajodia. Secure data aggregation in wireless sensor networks. *Information Forensics and Security, IEEE Transactions on*, 7(3):1040–1052, 2012.

[23] Bartosz Przydatek, Dawn Song, and Adrian Perrig. Sia: Secure information aggregation in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 255–265, New York, NY, USA, 2003. ACM.

[24] Luca de Alfaro and Michael Shavlovsky. Crowdgrader: Crowdsourcing the evaluation of homework assignments. *CoRR*, abs/1308.5273, 2013.

[25] Larry Wasserman. *All of statistics : a concise course in statistical inference*. Springer, New York, 2010.

[26] Mohsen Rezvani, Aleksandar Ignjatovic, Elisa Bertino, and Sanjay Jha. Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks. Technical Report UNSW-CSE-TR-201319, School of Computer Science and Engineering, UNSW, July 2013.

[27] Xiangliang Zhang, Cyril Furtlehner, Cecile Germain-Renaud, and Michele Sebag. Data stream clustering with affinity propagation. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1, 2013.

[28] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *In 2006 SIAM Conference on Data Mining*, pages 328–339, 2006.

[29] Yang Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 12(2):159–170, 2010.

[30] Suat Ozdemir and Yang Xiao. Ftda: outlier detection-based fault-tolerant data aggregation for wireless sensor networks. *Security and Communication Networks*, 6(6):702–710, 2013.

[31] Jaxk Reeves, Jien Chen, Xiaolan L Wang, Robert Lund, and Qi Qi Lu. A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*, 46(6):900–915, 2007.

[32] S. Muthukrishnan, Eric van den Berg, and Yihua Wu. Sequential change detection on data streams. In *Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, ICDMW '07, pages 551–550, Washington, DC, USA, 2007. IEEE Computer Society.

[33] The SensorScope lausanne urban canopy experiment (LUCE) project. *Data set available at: http://sensorscope.epfl.ch/index.php/LUCE*, 2006.

[34] University of California at Berkeley. Habitat monitoring on great duck island. *http://www.greatduckisland.net/*, 2004.

[35] NAMOS: Networked aquatic microbial observing system. *Data set available at: http://robotics.usc.edu/~namos/data/jr_oct/web/*, 2005.

[36] Guillermo Barrenetxea, François Ingelrest, Gunnar Schaefer, Martin Vetterli, Olivier Couach, and Marc Parlange. Sensorscope: Out-of-the-box environmental monitoring. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 332–343, Washington, DC, USA, 2008. IEEE Computer Society.

[37] Hyo-Sang Lim, G. Ghinita, E. Bertino, and M. Kantarcioglu. A game-theoretic approach for high-assurance of data trustworthiness in sensor networks. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 1192 –1203, april 2012.

[38] Matus Medo and Joseph R. Wakeling. The effect of discrete vs. continuous-valued ratings on reputation and ranking systems. *CoRR*, abs/1001.3745, 2010.

[39] Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78, September 2007.

[40] Trust mechanisms in wireless sensor networks: Attack analysis and countermeasures. *Journal of Network and Computer Applications*, 35(3):867 – 880, 2012.

[41] Saurabh Ganeriwal, Laura K. Balzano, and Mani B. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Trans. Sen. Netw.*, 4(3):15:1–15:37, June 2008.

[42] Xiang-Yan Xiao, Wen-Chih Peng, Chih-Chieh Hung, and Wang-Chien Lee. Using SensorRanks for in-network detection of faulty readings in wireless sensor networks. In *Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access*, MobiDE '07, pages 1–8, New York, NY, USA, 2007. ACM.

[43] Ming Li, Deepak Ganesan, and Prashant Shenoy. PRESTO: feedback-driven data management in sensor networks. In *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, NSDI'06, pages 23–23, 2006.

[44] Hyo-Sang Lim, Yang-Sae Moon, and Elisa Bertino. Provenance-based trustworthiness assessment in sensor networks. In *Proceedings of the Seventh International Workshop on Data Management for Sensor Networks*, DMSN '10, pages 2–7, 2010.

[45] Yan Sun, Hong Luo, and Sajal K. Das. A trust-based framework for fault-tolerant data aggregation in wireless multimedia sensor networks. *IEEE Trans. Dependable Secur. Comput.*, 9(6):785–797, November 2012.

[46] Lu-An Tang, Xiao Yu, Sangkyum Kim, Jiawei Han, Chih-Chieh Hung, and Wen-Chih Peng. Tru-Alarm: Trustworthiness analysis of sensor networks in cyber-physical systems. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 1079–1084, 2010.

[47] Hani Mohammed Alzaid. *Secure data aggregation in wireless sensor networks.* PhD thesis, Queensland University of Technology, 2011.

[48] Jun-Won Ho, M. Wright, and S.K. Das. ZoneTrust: Fast zone-based node compromise detection and revocation in wireless sensor networks using sequential hypothesis testing. *Dependable and Secure Computing, IEEE Transactions on*, 9(4):494 –511, july-aug. 2012.

[49] Suat Ozdemir and Hasan Çam. Integration of false data detection with data aggregation and confidential transmission in wireless sensor networks. *IEEE/ACM Trans. Netw.*, 18(3):736–749, June 2010.