2015-11-12

Secure Data Aggregation for Sensor Networks in the Presence of Collusion Attack using Local Outlier Factor

Anes Yessembayev
University of Miami, anes@bk.ru

UNIVERSITY OF MIAMI

SECURE DATA AGGREGATION FOR SENSOR NETWORKS
IN THE PRESENCE OF COLLUSION ATTACK
USING LOCAL OUTLIER FACTOR

By

Anes Yessembayev

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science

Coral Gables, Florida

December 2015

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

SECURE DATA AGGREGATION FOR SENSOR NETWORKS IN THE PRESENCE OF COLLUSION ATTACK USING LOCAL OUTLIER FACTOR

Anes Yessembayev

Approved:

____________________ Dilip Sarkar, Ph.D. Associate Professor of Computer Science	____________________ Geoff Sutcliffe, Ph.D. Professor of Computer Science
____________________ Subramanian Ramakrishnan, Ph.D. Associate Professor of Mathematics	____________________ Dean of the Graduate School

YESSEMBAYEV, ANES (M.S., Computer Science)
Secure Data Aggregation for Sensor Networks (December 2015)
in the Presence of Collusion Attack
using Local Outlier Factor

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Dilip Sarkar.
No. of pages in text. (55)

Aggregation of data from multiple sensor nodes is usually done by simple methods such as averaging or, more sophisticated, iterative filtering methods. However, such aggregation methods are highly vulnerable to malicious attacks where the attacker has knowledge of all sensed values and has ability to alter some of the readings. In this work, we develop and evaluate algorithms that eliminate or minimize the influence of altered readings. The basic idea is to consider altered data as outliers and find algorithms that effectively identify altered data as outliers and remove them. Once the outliers have been removed, use some standard technique to estimate a true value.

We calculate local outlier factor (LOF) for each data point. We propose methods for computing threshold values from these LOF values. The data points that have LOF value above a calculated threshold value are removed before computing an estimated signal from the data points reported by the sensors. Thus, the proposed data aggregation algorithm operates in two phases: removal of outliers and computation of an estimated true value from the remaining sensor data. Extensive evaluation of the proposed algorithms show that they significantly outperform all existing methods. For simple cases, we have developed expressions for calculating LOF values.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The present level of processes automation requires extensive use of various sensors. Due to unreliability of sensors they are deployed redundantly. Data from multiple sensor nodes is accumulated and combined by an *aggregator node*. An aggregator node not only collects readings from sensors, but also minimizes or eliminates the influence of readings from faulty or compromised sensors. Secure data aggregation algorithms for sensor networks aim to provide mechanisms for eliminating or resisting data distortion. These algorithms are usually run on an aggregator node or a *base station*.

1.1 Data Aggregation Methods

Probably the earliest and easiest method of data aggregation is simple averaging of readings from all sensors. However, simple averaging method has some major drawbacks, because it does not consider the existence of bias errors or faulty sensors and not to mention malicious attacks. Only one faulty sensor may reduce the accuracy of aggregated result significantly. In addition, the method does not doubt any sensor's reading. This makes the method highly vulnerable even to a *simple attack*, where the attacker skews reading of one or more sensors to a certain degree to alter estimated reading.

Iterative Filtering (IF) algorithms offer refined approaches (see [8] and [18] for examples). They initially assign one weight to the reading of each sensor and then weights are recalculated at each iteration based on the distance of the readings from the estimated value obtained in the previous iteration. They reduce the effect of a simple attack. But the weakest point of these iterative filtering algorithms is the use of a predetermined procedure for assigning an initial weight to each sensor's reading.

An iterative algorithm is vulnerable to a *malicious attacker* who has knowledge of all readings and has power to *alter* two or more readings [18]. A malicious attacker can force the iterative filtering algorithm to converge to a desired value by altering readings of the compromised sensors (for details of an example situation see [18]).

To overcome weakness of the iterative filtering algorithm in [8], a *Robust Data Aggregation Method* (RDAM) was proposed in [18]. The main idea of the algorithm is to estimate a set of non-equal initial weights for the readings. The objective of the method is to calculate smaller initial weights for the readings of the compromised sensors. Results of extensive empirical evaluation of the RDAM method against other methods demonstrated highest accuracy for both simple and collusion attacks (for details see [18]).

However, any estimation of mean and standard deviation is extremely sensitive to presence of outliers [1],[14]. Thus, any aggregation algorithm that estimates true values from sensor-readings before removing outliers is susceptible to errors. In this work, we propose and evaluate a set of two phase data aggregation algorithms. In the first phase the outliers are identified and then, in the second phase an estimate is calculated.

1.2 Overview of the Thesis

In this work, we consider variations of the *collusion attack*. Our extensive evaluations discovered that in certain conditions the RDAM cannot overcome the influence of malicious attacker. In this work several two phase algorithms are proposed and evaluated.

The first phase of the proposed method employs a variant of *Local Outlier Factor* (LOF) calculation method [6] to estimate the degree that an object is an outlier because of collusion attack, sensor fault, noise, or a combination of them. This gives a flexible instrument to exclude suspected sensor-readings before estimation of a true value. Note that this method, unlike methods described previously, removes sensor-readings from compromised or bad sensors, which improve estimated values, while decrease the amount of calculations in the second phase of the proposed method. We present a set of different methods, including IF and RDAM, after LOF method. Also we test some variations of the algorithms in cooperation with *k-means clustering method* [9]. Finally, we introduce two LOF-based non-iterative algorithms that we found most accurate almost in all considered attacks. Moreover, the methods work without IF portion, that excludes disadvantages and vulnerabilities related to it.

The rest of thesis is organized as follows. Chapter 2 describes the problem statement. Here we briefly present most widely used topology of sensor networks and review their potential vulnerabilities. Chapter 3 briefly reviews work related to studies here. We describe existing data aggregation as well as outlier detection methods. Chapter 4 presents basic definitions and steps for computation of LOF. Chapter 5 presents our novel algorithms. We describe different combinations of using *LOF* method that we apply in our algorithms. Chapter 6 presents some theoretical results for computing LOF values when part of the data is from collusion attack. Chapter 7 shows our experimental results. Finally, the conclusion is provided in Chapter 8.

Chapter 2

Background

2.1 Model

The sensor network topology used for our work is an abstract model proposed in [20] (see Figure 2.1). A sensor network is built of a *base station* and a set of sensor *clusters*. Each cluster has a *cluster head* that gathers data from all *sensor-nodes* connected to it. A cluster head is also known as an *aggregator node*, because it gathers readings from multiple *sensor nodes*. The main functions of an aggregator node are collecting data from its sensor nodes, aggregating the raw data to produce an estimated reading, and communicating the processed data to the base station. Each sensor node has a micro-controller with one or multiple sensors. The micro-controller is equipped with relatively small memory and computing power, while an aggregator has bigger memory and higher computing power. A base station has larger memory and computing power, in addition to communication capabilities.

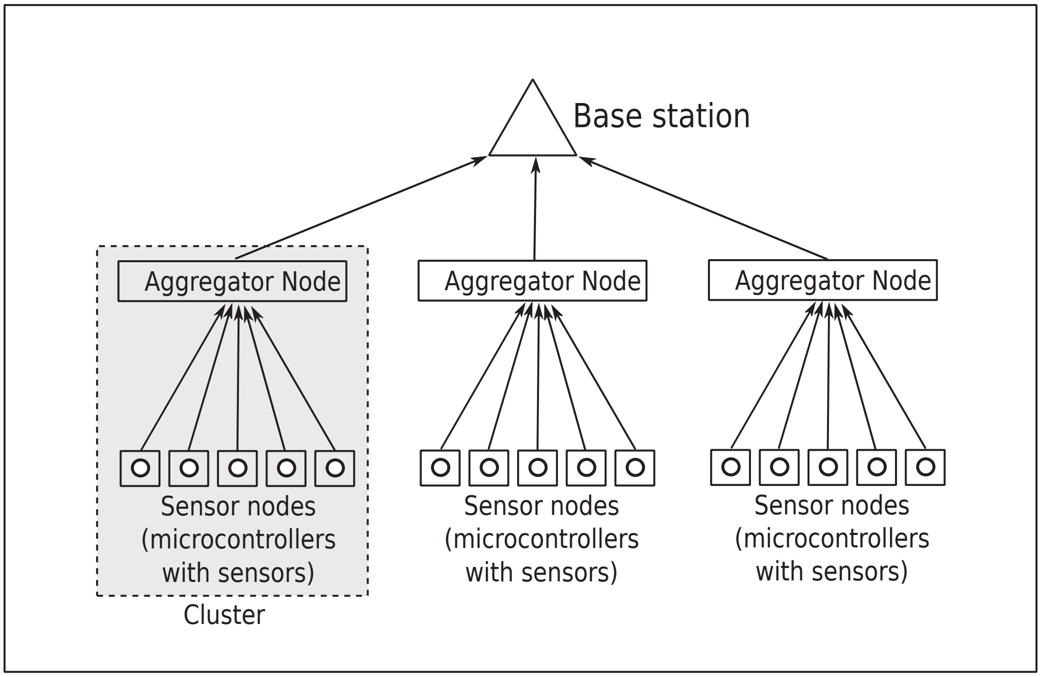

Base station
Aggregator Node
Aggregator Node
Aggregator Node
Sensor nodes (microcontrollers with sensors)
Sensor nodes (microcontrollers with sensors)
Sensor nodes (microcontrollers with sensors)
Cluster

Figure 2.1: Sensor network topology.

2.2 Problem Statement

Sensor networks often operate in unattended environments and are deployed distributively, which makes them highly susceptible to failure and physical attacks. This creates threats to sensor networks security. Although in general, *security* can be defined as the combination of availability, confidentiality, and integrity, in our work we focus only on *integrity*. In addition, it is assumed that the aggregator nodes and the base station are not compromised. Assuming these limitations, we identify possible threats to the sensor network and propose solutions to overcome them (see Figure 2.2).

Classification of threats to sensor networks and their solutions

Nowadays, inexpensive sensors are embedded in numerous devices. These inexpensive senors are not only unreliable, but they are also susceptible to failure. Moreover, they are usually insecure and their outputs are easily manipulated by malicious attackers.

Any sensor reading further from *true* value can be considered as an outlier, and in the literature they are categorized into several classes (see [13],[19], and [17]).

1. *Noise:* data with greater variance.

2. *Spike:* data with one or more out-of-bound readings.

3. *Stuck-at:* data with quasi-zero variance.

Consequently we cannot depend on readings from a single sensor. However, low cost of sensors allows one to compensate for their unreliability through use of multiple sensors. For example, after one receives readings from multiple sensors and apply a filtering technique, e.g. *Iterative Filtering*, the contributions of reading from unreliable sensors are reduced. Furthermore, this approach minimizes errors because of simple attacks mentioned in Chapter 1.

However, this technique cannot overcome a *malicious attack*, if and when the attacker has knowledge of all readings, details about filtering technique, and capability to alter some of the readings. An example of a malicious attack is described in [18], where a *collusion attack* forces the *Iterative filtering algorithm* to converge to a skewed value. Rezvani et al. in [18] proposed an algorithm to improve *Iterative Filtering* and it is capable of reducing impacts of colluded readings. However, in our experiments we found conditions under which algorithms for reducing influence of colluded readings, such as RDAM method, are not effective. As a solution for such types of attacks we propose a set of variations of outlier detection algorithms that identify and, note, completely remove the impact of compromised readings. Considering that a micro-controller is very limited in its performance we set a criteria for the algorithms that they do not require high computational efforts.

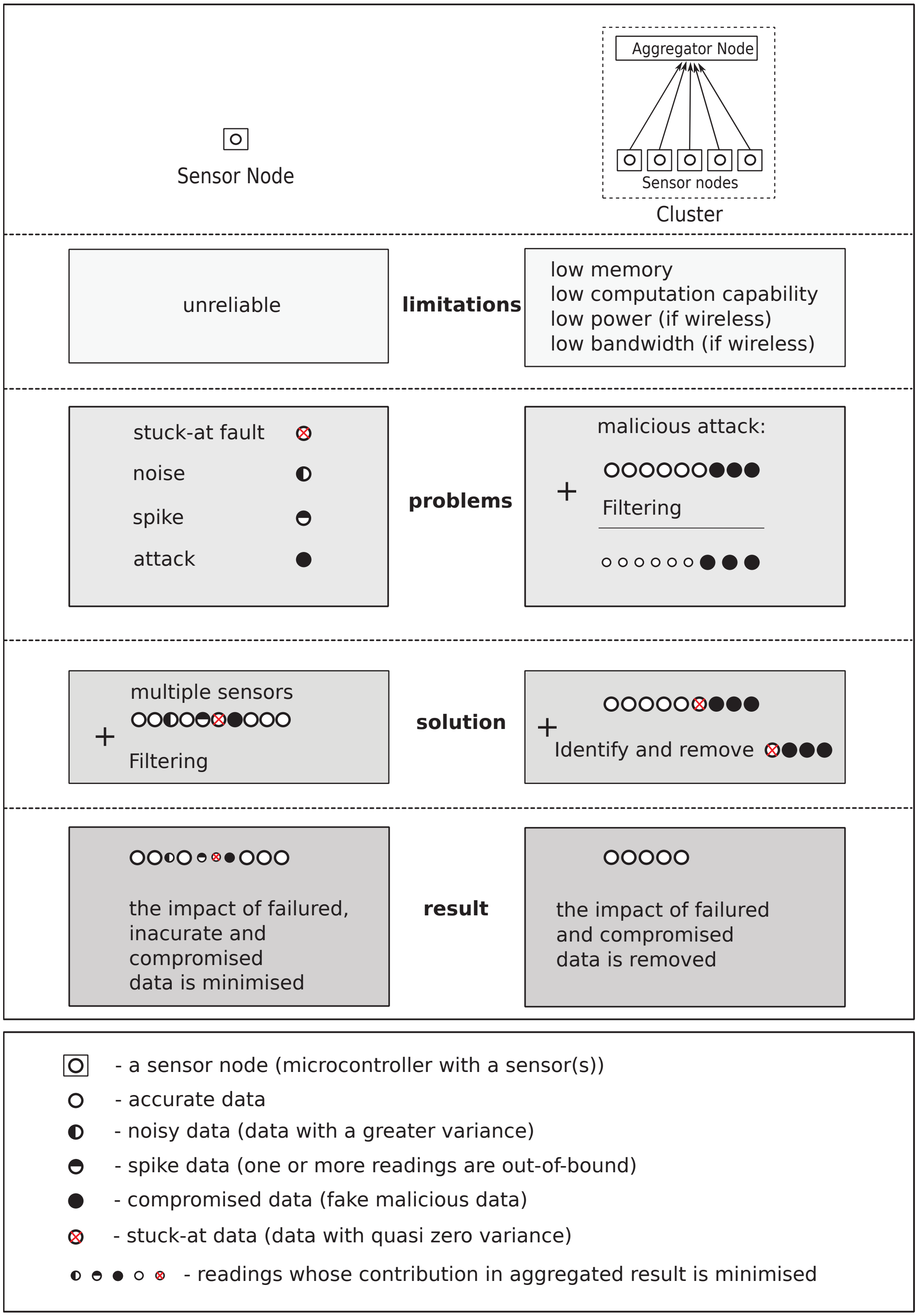

Sensor Node
Aggregator Node
Sensor nodes
Cluster
unreliable
limitations
low memory
low computation capability
low power (if wireless)
low bandwidth (if wireless)
stuck-at fault ⊗
noise ◐
spike ◒
attack ●
problems
malicious attack:
○○○○○○●●●
+
Filtering
○○○○○○●●●
multiple sensors
○○◐○◒⊗●○○○
+
Filtering
solution
○○○○○⊗●●●
+
Identify and remove ⊗●●●
○○◐○◒⊗●○○○
the impact of failured, inacurate and compromised data is minimised
result
○○○○○
the impact of failured and compromised data is removed
▣ - a sensor node (microcontroller with a sensor(s))
○ - accurate data
◐ - noisy data (data with a greater variance)
◒ - spike data (one or more readings are out-of-bound)
● - compromised data (fake malicious data)
⊗ - stuck-at data (data with quasi zero variance)
◐ ◒ ● ○ ⊗ - readings whose contribution in aggregated result is minimised

Figure 2.2: Sensor network threats and solutions.

Chapter 3

Related Work

The extensive use of cheap and unreliable sensors mandates development of new algorithms for aggregating data securely, because these sensors are not only unreliable, but they are also easily compromised by malicious attackers. Also, it is necessary to take into account that new types of attacks continue to emerge with time. Thus, a robust algorithm that is resistant to known attacks, also should be resistant to emerging attacks. There are a number of research approaches and works related to our topic.

In our examples and experiments we consider a data model depicted in Table 3.1, where n - is total number of sensors, m - is number of readings from each sensor, $x_s^{(t)}$ - data from sensor s at time t.

Table 3.1: Data Model

Sensor readings				
instant	$s = 1$	$s = 2$	...	$s = n$
$t = 1$	$x_1^{(1)}$	$x_2^{(1)}$	...	$x_n^{(1)}$
$t = 2$	$x_1^{(2)}$	$x_2^{(2)}$	...	$x_n^{(2)}$
...	...	...	...	...
$t = m$	$x_1^{(m)}$	$x_2^{(m)}$	...	$x_n^{(m)}$

3.1 Iterative Filtering Algorithms

Certainly the most studied method related to our research is Iterative Filtering Algorithms [16],[23],[26],[8],[15],[2]. In general, the majority of the work was dedicated for rating networks, where users give ratings to objects. Mizzaro proposed an algorithm for the assessment of scholarly papers [16]. Yu et al. [23] and, more recently, Zhou et al. [26] presented iterative algorithms for rating networks. However, these reputation-based algorithms may not always converge. Subsequently, a number of studies have proposed novel methods to overcome the convergence issue. C. de Kerchove and P. Van Dooren introduced a convergent Iterative Filtering algorithm using three discriminant functions: inverse, exponential, and Affine [8]. Li et al. in [15] presented six iterative algorithms, where users reputation is calculated using the aggregated difference between the users rating and the corresponding objects ranking. Ayday et al. developed an iterative algorithm with probabilistic and belief propagation-based approach (see details in [2]). However, a fundamental problem of all IF algorithms is that they are primarily aimed against simple cheaters and do not consider severe malicious attacks such as a *collusion attack* that we mentioned in Chapter 1. Let us consider an example to illustrate this issue.

Example 1. *Let us consider a dataset X with $m = 3$ readings from each of the $n = 8$ sensors. Sensors s_1 - s_5 are providing their* true *readings to the aggregator, while s_6 - s_8 are under the influence of a malicious attacker and providing* manipulated *readings. The reading from s_8 at time* t *is equal to the average of all readings of all sensors. Table 3.2 shows how the algorithm converges quickly to the average readings from s_8. This happens because the algorithm assigns equal initial weights for each sensor at the first round.*

Table 3.2: Iterative Filtering Algorithm converges to colluded readings from s_8 sensor.

	Sensor readings								aggregate values		
instant	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$			
$t = 1$	10.73	9.61	9.77	10.20	10.42	15.45	15.13	**11.6157**			
$t = 2$	10.75	9.63	9.73	10.21	10.43	15.75	15.93	**11.7757**			
$t = 3$	10.74	9.69	9.92	10.32	10.9	15.98	15.39	**11.8486**			
round	Sensor weights								t=1	t=2	t=3
1	1	1	1	1	1	1	1	1	11.6157	11.7757	11.8486
2	0.9786	0.2258	0.2652	0.4417	0.7246	0.0631	0.0712	3.2000E+09	11.6157	11.7757	11.8486
3	0.9786	0.2258	0.2652	0.4417	0.7246	0.0631	0.0712	1.1625E+18	**11.6157**	**11.7757**	**11.8486**

3.2 Enhanced Iterative Filtering Algorithm

To reduce effect of collusion attack, Rezvani et al. proposed a *Robust Data Aggregation Method* (RDAM), which is an improvement to IF method. In this method, unequal initial weights are calculated using the readings available to the aggregator [18]. This enhancement not only makes an IF algorithm collusion attack resistant, but it also reduces the number of required iterations to converge to an estimated value. The method is based on the assumption that the distribution of stochastic components of sensor errors is known or can be estimated.

Example (Cont.). *With the same data the algorithm reduces the impact of readings from s_6 - s_8 (see Table 3.3). The algorithm estimates unequal initial weights for the readings at the first round. This allows to avoid converging to the values of s_8. Although the algorithm in the example converges to the values of s_4, in real data the algorithm usually converges to unique values, which actually occurred in our extensive simulations.*

Table 3.3: RDAM avoids converging to colluded readings from s_8 sensor.

	Sensor readings								aggregate values		
instant	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$			
$t = 1$	10.73	9.61	9.77	**10.20**	10.42	15.45	15.13	11.6157			
$t = 2$	10.75	9.63	9.73	**10.21**	10.43	15.75	15.93	11.7757			
$t = 3$	10.74	9.69	9.92	**10.32**	10.9	15.98	15.39	11.8486			
round	Sensor weights								t=1	t=2	t=3
1	0.0722	0.3551	0.0707	0.4689	0.0055	0.0072	0.0011	0.0193	10.0699	10.0865	10.1772
2	2.5168	4.5607	10.5808	57.2282	3.9345	0.0317	0.0345	0.3734	10.1473	10.1527	10.2842
3	3.3182	3.2789	6.6131	408.4660	5.6548	0.0326	0.0356	0.4145	10.1980	10.2076	10.3222
4	3.9912	2.7799	5.2341	2.07E+05	6.9363	0.0332	0.0363	0.4413	10.2000	10.2100	10.3200
5	4.0060	2.7741	5.2145	1.38E+10	6.9255	0.0332	0.0363	0.4417	**10.2000**	**10.2100**	**10.3200**

3.3 Statistical Estimation

If the readings from the sensors have no outliers, a statistical method for estimating true values is described here. The main assumption for the method is that every sensor has a certain error in its readings, because no sensor gives accurate readings continuously. This suggests to find a method for estimation of a true value in such a way that all the readings are considered, but weights vary based on temporal variation pattern of the readings. The sensor that has least variance is considered to have provided better readings. We propose to use the variance of readings of a sensor to determine a weight that is applied to the readings of the sensor when a true value is estimated from readings of all the sensors. A top-down description of the method starts from Equation (3.1).

$$w_s = \frac{\frac{1}{\upsilon_s}}{\sum_{i=1}^{n} \frac{1}{\upsilon_i}}, \tag{3.1}$$

where υ_s is defined as

$$\upsilon_s = \frac{1}{m-1} \cdot \sum_{t=1}^{m} (\mu_s - x_s^{(t)})^2, \tag{3.2}$$

and μ_s is the mean of all readings from sensor s

$$\mu_s = \frac{1}{m} \cdot \sum_{t=1}^{m} x_s^{(t)} \tag{3.3}$$

After the weights are calculated, estimates of the true values are obtained using the equation below.

$$\overline{r}^{(t)} = \sum_{s=1}^{n} w_s \cdot x_s^{(t)} \tag{3.4}$$

This method should be used only if the sensor readings are free of outliers or outliers have been identified and removed from the readings. Next we describe algorithms useful for outlier detection.

3.4 Outlier Detection Algorithms

Detection of outliers in a dataset is a well studied subject. But the information age, especially big-data and *Internet of Everything* (IoE), has seen renewed interest in the subject to fit the current needs. In our study, we use a recently proposed method and it is described in Chapter 4. In this section we provide a brief overview of the existing outlier detection method.

In general, outlier detection methods form five main classes: statistical, nearest neighbor, clustering, classification, and spectral decomposition [25]. In statistical methods outliers are defined as objects that do not fit in assumed distribution [3],[24],[22]. Nearest neighbor methods use a distance as a mean to distinguish outliers [5],[6].

Clustering algorithms use similarity metrics [12]. Most popular and probably old clustering algorithm is *k-means* [9], which we also apply in our experiments. Classification methods use training phase when they create mathematical models for regular data and outliers. After that they classify new data using these models [21],[11]. Spectral decomposition methods use principal component analysis to create a model structure. Readings that do not fit the structure are considered as outliers [7].

The methods in last three groups are characterized as requiring high computational efforts. However, sensor micro-controllers are limited in their performance. This motivates us to use in our algorithm a method that demands for relatively low computing power.

There is the *Local Outlier Factor* algorithm among *Nearest neighbor methods* group proposed by Breunig M. et al. for finding anomalous objects by measuring the local deviation of a given object with respect to its neighbors [6]. The method is able to detect outliers even if they form a dense cluster together regardless the data distribution. The method has a complexity $O(n^2)$, where n is the number of data objects and is considered not a good choice for devices with limited computing power. However, our theoretical analysis in Chapter 6 shows that original equations described in Chapter 4 can be significantly simplified and adapted for the type of attacks considered here.

Chapter 4

An Algorithm for LOF and Its Application

4.1 Steps for Calculations of LOF

In this section we present basic definitions and steps for computation of *local outlier factor* (LOF) presented in [6]. The idea of LOF is based on the concept of local density. Computation of LOF involves six steps (see Figure 4.1). Since each step is essential for obtaining a final value of each data object and we will use these definitions for developing some novel computation algorithms for special cases, we formally define computation of each step. First we present an informal definition from [10].

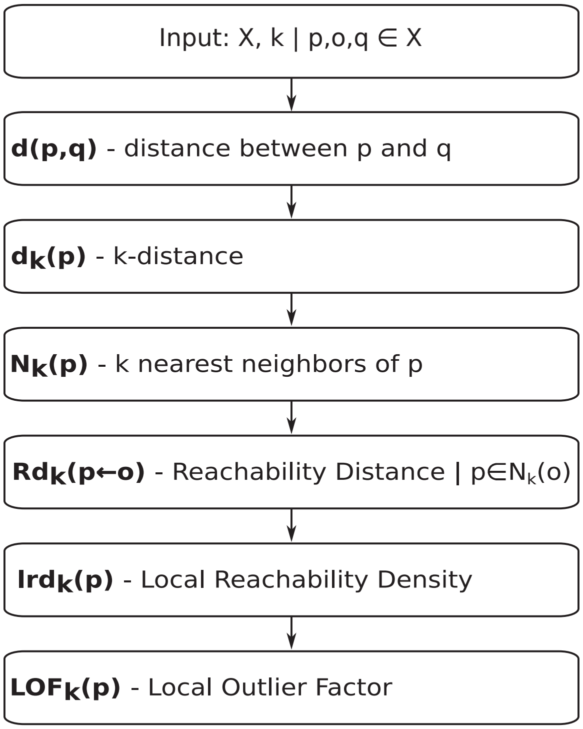

Input: X, k | p,o,q ∈ X
d(p,q) - distance between p and q
$d_K(p)$ - k-distance
$N_K(p)$ - k nearest neighbors of p
$Rd_K(p \leftarrow o)$ - Reachability Distance | $p \in N_k(o)$
$lrd_K(p)$ - Local Reachability Density
$LOF_K(p)$ - Local Outlier Factor

Figure 4.1: Outlier Detection algorithm.

Definition 1. *(*Informal definition [10]:*) An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.*

□

For ease of communication, we use symbols o, p, and q to denote objects in a dataset X. The notation $d(p, q)$ is used to represent distance between two objects p and q. A formal definition for $d(p, q)$ is provided next.

Definition 2. (Distance $d(p, q)$:) *Distance between two points p and q, denoted as $d(p, q)$, is a nonnegative number, such that (i) $d(p, p) = 0$, (ii) $d(p, q) > 0$, if $p \neq q$, and (iii) for three distinct points o, p, and q if $d(o, p) = x$ and $d(p, q) = y$, then $d(o, q) \leq (x + y)$.*

□

Above definition encompasses many methods for measuring the distance between two points, including Euclidean distance. For ease of conveying concepts, most of the definitions are accompanied by a running example.

Example 2. *Let us consider four data points $a(0)$, $b(1)$, $c(2)$, and $d(5)$. The number in the parenthesis represents the position on the x-axis. These points are shown in Figure 4.2. We use Euclidean distance between points and the values of distance for all distinct pair of points are shown as following.*

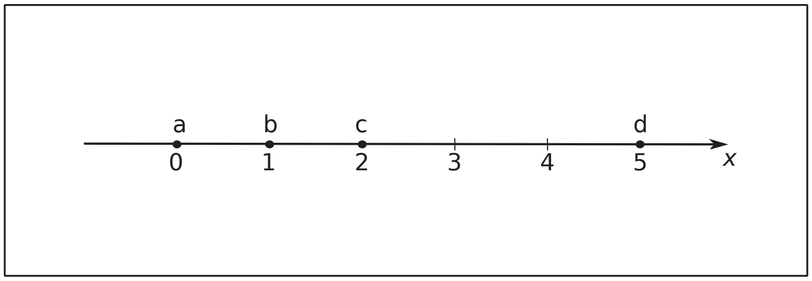


Figure 4.2: Four data points used to illustrate calculations in the algorithm that follows.

d(a, b) = 1; d(a, c) = 2; d(a, d) = 5; d(b, c) = 1; d(b, d) = 4; d(c, d) = 3.

□

In the next step, the outlier-detection algorithm calculates $d_k(p)$. It should be noted that $d_k(p)$ is neither distance of the furthest point nor the number of points in the neighborhood of the point p. It is true that the neighborhood of p will have at least k points, but the actual number may be much more than k.

Definition 3. *(k-distance $d_k(p)$ [6]) For any positive integer k, the k-distance of object p, denoted as $d_k(p)$, is defined as the distance $d(p, o)$ between p and an object $o \in X$ such that:*

(i) for at least k objects $q \in X \setminus \{p\}$ it holds that $d(p, q) \leq d(p, o)$, and

(ii) for at most $(k - 1)$ objects $q \in X \setminus \{p\}$ it holds that $d(p, q) < d(p, o)$.

□

In our example, $d_2(.)$ of all the points are 2, but this is not true in general.

Example (Cont.). *We consider four data points $a(0)$, $b(1)$, $c(2)$, and $d(5)$ from Example 4.2. We use $k = 2$ for this example.*

$d_2(a) = d(a, c) = 2$*; (c is the* second *nearest neighbor);*

$d_2(b) = d(b, a) = 1$*; (a/c is the* second *nearest neighbor);*

$d_2(c) = d(c, a) = 2$*; (a is the* second *nearest neighbor);*

$d_2(d) = d(d, b) = 4$*; (b is the* second *nearest neighbor);*

□

After $d_k(\cdot)$, the k-distance, calculation of all the points, the next step is to compute $N_k(\cdot)$. Intuitively, $N_k(p)$ is the set of all points in the neighborhood of p whose distance is less than or equal to $d_k(p)$. Formally, $N_k(p)$ is defined as

Definition 4. *(k-distance neighbors $N_k(p)$ [6]) Given the k-distance $d_k(p)$ of p, k-distance neighbors of an object p are the objects whose distance from p is not greater than the k-distance of p, that is, $N_k(p) = \{q \in X \setminus \{p\} \mid d(p, q) \leq d_k(p)\}$.*

□

Example (Cont.). *We continue to use the same four points in Figure 4.2. Below are 2-distance neighbors of all four points.*

$N_2(a) = \{b, c\}$; $\quad N_2(b) = \{a, c\}$; $\quad N_2(c) = \{a, b\}$; $\quad N_2(d) = \{b, c\}$

Reachability Distance of an object p with respect to another object o, denoted as $Rd_k(p \leftarrow o)$, is defined next. The definition is followed by computation of $Rd_2(b \leftarrow a)$ for points b and a in our example. *It is important to note that we have extended the scope of the definition in [6].* Since readings from all sensors could be identical in some cases, the k-distance could be zero in those cases. Our definition takes care of those special cases.

Definition 5. *(Reachability Distance $Rd_k(p \leftarrow o)$) The Reachability Distance of object p with respect to another object o is defined as*

$$Rd_k(p \leftarrow o) = \max\{d_k(o), d(p, o), \epsilon\}, \tag{4.1}$$

where ϵ is a small constant that is introduced to avoid division by zero operation in further calculations.

□

For the objects b and a in the Example 2, $Rd_2(b \leftarrow a) = \max\{d_2(a), d(a, b), \epsilon\}$ $= \max\{2, 1, \epsilon\} = 2$. As will be clear later, computation of $Rd_k(p \leftarrow o)$ for all pairs of points in X may not be necessary.

For an object $p \in X$, the values of $Rd_k(o) \in N_k(p)$ and cardinality of $N_k(p)$ are used to compute *Local Reachability Density*, $lrd_k(p)$.

Definition 6. *(Local Reachability Density $lrd_k(p)$ [6]) The* local reachability density *of an object* p *is defined as*

$$lrd_k(p) = \frac{|N_k(p)|}{\sum_{q \in N_k(p)} Rd_k(q \leftarrow p)}, \tag{4.2}$$

where $|N_k(p)|$ is the number of objects in $N_k(p)$.

□

Example (Cont.). *Now we illustrate computation of $lrd_2(a)$.*

$$lrd_2(a) = \frac{|N_k(a)|}{\sum_{q \in N_k(a)} Rd_k(q \leftarrow a)} = \frac{|N_2(a)|}{Rd_2(b \leftarrow a) + Rd_2(c \leftarrow a)}$$

Now,

$$\begin{aligned} Rd_2(b \leftarrow a) &= \max\{d_2(a), d(a,b), \epsilon\} \\ &= \max\{2, 1, \epsilon\} = 2, \end{aligned}$$

and

$$\begin{aligned} Rd_2(c \leftarrow a) &= \max\{d_2(a), d(a,c), \epsilon\} \\ &= \max\{2, 2, \epsilon\} = 2 \end{aligned}$$

Thus, $lrd_2(a) = \frac{2}{2+2} = 0.5$

Following similar steps, we obtain values of $lrd_2(b)$, $lrd_2(c)$, *and* $lrd_2(d)$*:*

$$lrd_2(b) = \frac{|N_2(b)|}{Rd_2(a \leftarrow b) + Rd_2(c \leftarrow b)} = \frac{2}{1+1} = 1;$$

$$lrd_2(c) = \frac{|N_2(c)|}{Rd_2(a \leftarrow c) + Rd_2(b \leftarrow c)} = \frac{2}{2+2} = 0.5;$$

$$lrd_2(d) = \frac{|N_2(d)|}{Rd_2(b \leftarrow d) + Rd_2(c \leftarrow d)} = \frac{2}{4+4} = 0.25;$$

□

The final step in the algorithm is to compute *Local Outlier Factor*, $LOF_k(p)$ of all the objects $p \in X$. For defining $LOF_k(p)$ of an object $p \in X$, value of $lrd_k(p)$, values of $lrd_k(q) \in N_k(p)$, and cardinality of $N_k(p)$ are used, but the expression can be simplified as shown below.

Definition 7. *(*Local Outlier Factor $LOF_k(p)$ *[6]) The* local outlier factor *of* p *is defined as*

$$LOF_k(p) = \frac{\sum_{q \in N_k(p)} \frac{lrd_k(q)}{lrd_k(p)}}{|N_k(p)|} = \left(\frac{\sum_{q \in N_k(p)} lrd_k(q)}{|N_k(p)| * lrd_k(p)}\right). \tag{4.3}$$

□

Example (Cont.). *We complete the example after calculating $LOF_2(.)$ of all objects in the example.*

$$\begin{aligned}
LOF_2(a) &= \frac{lrd_2(b) + lrd_2(c)}{|N_k(a)| * lrd_2(a)} = \frac{1 + 0.5}{2 * 0.5} = 1.5;\\
LOF_2(b) &= \frac{lrd_2(a) + lrd_2(c)}{|N_k(b)| * lrd_2(b)} = \frac{0.5 + 0.5}{2 * 1} = 0.5;\\
LOF_2(c) &= \frac{lrd_2(a) + lrd_2(b)}{|N_k(c)| * lrd_2(c)} = \frac{0.5 + 1}{2 * 0.5} = 1.5;\\
LOF_2(d) &= \frac{lrd_2(b) + lrd_2(c)}{|N_k(d)| * lrd_2(d)} = \frac{1 + 0.5}{2 * 0.25} = 3;
\end{aligned}$$

After sorting these local outlier factors we obtain,

$LOF_2(d) = 3$; $LOF_2(a) = 1.5$; $LOF_2(c) = 1.5$; $LOF_2(b) = 0.5$

Obviously, top first outlier in this example is the object d.

4.2 Selecting k for Computing LOF in the Presence of Collusion Attack

Weaknesses of the Local Outlier Factor method are complexity $O(n^2)$ and selecting the right k is not obvious. According to [6] the optimal value of k is defined as $k = (n_g - 1)$, where n_g is the number of reliable objects. For our case the simplest way to select k is based on the assumption: *there are considerably more 'normal' observations than 'abnormal' observations (outliers/anomalies) in the data.* That hypothesis is enough to set

$$k = \lceil n/2 \rceil \tag{4.4}$$

to get decent results, where n is total number of observations.

Assume we have 10 data points, 3 of which are colluded (see Figure 4.3).

By calculating Local Outlier Factors (4.3) based on the average distance from every point p to the nearest $k = \lceil n/2 \rceil = 5$ points q we get that Local Outlier Factors of colluded data points are significantly higher, that gives a mathematical instrument to exclude them from further calculations. However, the simplicity has a drawback. Non-adaptive k leads to the same number of calculations for different number of colluders, which in some cases can make the value of LOF for outlier not so clearly prominent from reliable objects.

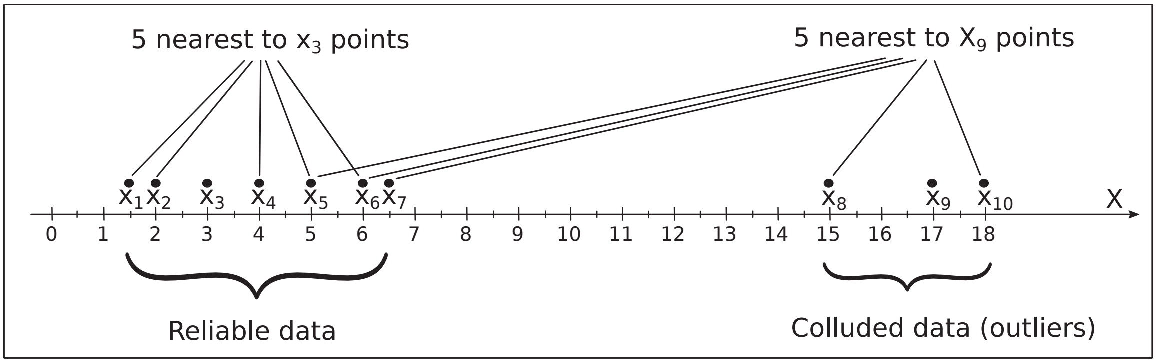


Figure 4.3: Local Outlier Factor method Conception.

There is a better way to determine the optimal k that uses *k-means*[1] *clustering method.* The method splits all objects into a given number of clusters in which each object belongs to the cluster with the nearest mean, serving as a prototype of the cluster. Assuming that objects form two clusters: reliable n_g and colluded n_b objects, k-means clustering method allows to find the optimal k defined in (4.5).

$$k = n_g - 1 \tag{4.5}$$

Using this method allows us to consider the optimal k in calculations (see Figure 4.4 and Section 5.1.4). In this case values of LOF for outliers reach the maximum, which makes them clearly distinguishable from reliable objects.

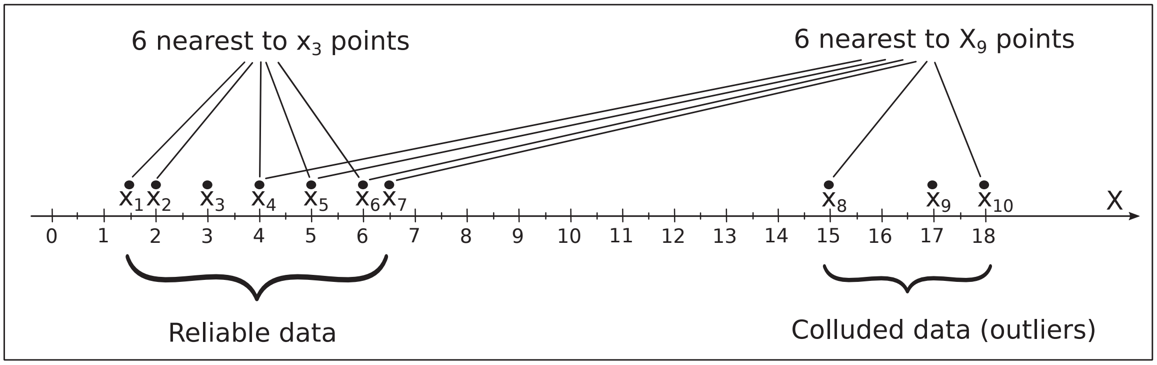


Figure 4.4: Local Outlier Factor with optimal k.

[1] In this context symbol k is a part of the name of the method and it is not related to the k-distance definition.

Chapter 5

Proposed Two Phase Data Aggregation Algorithms

In this chapter we present our algorithms. Logically and functionally our algorithms consist of two phases: 1) Detection and removal of outliers, 2) True value estimation with remaining data (see Figure 5.1). Table 5.1 contains notations used in descriptions for the algorithms.

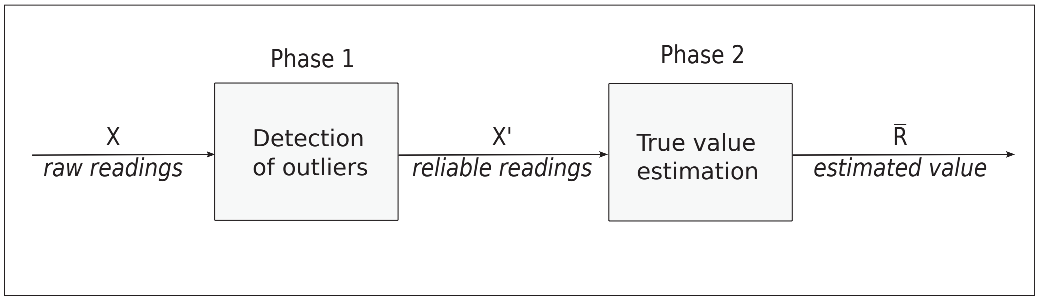


Figure 5.1: Two phases of our algorithms.

Table 5.1: Notations.

$r^{(t)}$	true value of the signal at time t
$\overline{r}^{(t)}$	estimated value of the signal at time t
$e_s^{(t)}$	noise (error) of sensor s at time t
S	a vector including numbers of all sensors
S'	a vector including numbers of reliable sensors
X	an array of source data
X'	an array of reliable data
n	number of sensors
m	number of readings from each sensor
c	number of compromised sensors
$x_s^{(t)}$	data from sensor s at time t
w_s	weight that reflects the trustworthiness of a sensor s
$\overline{X}$	a vector of average readings from each sensor
$\overline{x}_s$	an average reading from sensor s
$\overline{X}'$	a vector of average readings from reliable sensors
$\overline{LOF}_k(X)$	an average of all $LOF_k(x_s)$
$\overline{LOF}_k(\overline{X})$	an average of all $LOF_k(\overline{x}_s)$

5.1 Detection of Outliers

Following the logic of the construction of our algorithms, in this section we describe methods that we apply in the first phase of our algorithms. This phase is dedicated to the detection of outliers.

5.1.1 Detection of Outliers in a Single Step

As the name suggests the proposed algorithm detects outliers (that is, colluded data) and excludes them from further calculations in a single step. In Chapter 4 we described how to obtain a sequence of $LOF_k(x_1), LOF_k(x_2), ..., LOF_k(x_n)$ for a vector $X = x_1, x_2, ..., x_n$. As discussed earlier, higher the value of LOF of an object, higher the probability of the object being a colluded object. The main challenge here is to set a criteria to split the sequence of LOF values so that only colluded objects are excluded.

Example 3. *For the data points depicted on the Figures 4.3 and 4.4, for $k = 5$ corresponding LOF parameters are shown in the Table 5.2.*

Table 5.2: Data points and corresponding LOF values

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
value of x_s	1.5	2	3	4	5	6	6.5	15	17	18
$LOF_5(x_s)$	1.41	1.23	0.87	0.67	0.87	1.23	1.41	1.93	2.36	2.57
$\overline{LOF}_5(X)$	1.456									
$LOF_5(x_s) < \overline{LOF}_5(X)$	true	true	true	true	true	true	true	false	false	false

Our empirical observations suggest that LOF_k of an outlier is above $\overline{LOF}_k$, which is defined as the average of all LOF_k values:

$$\overline{LOF}_k(X) = \frac{\sum_{s=1}^{n} LOF_k(x_s)}{n} \tag{5.1}$$

Example (Cont.). *So, for the dataset in Table 5.2 $\overline{LOF}_5(X) = (1.41 + 1.23 + 0.87 + 0.67 + 0.87 + 1.23 + 1.41 + 1.93 + 2.36 + 2.57)/10 = 1.456$*

Thus, data points $x_8(15), x_9(17)$, and $x_{10}(18)$, whose LOF values are greater than $\overline{LOF}_5$, are identified as outliers, and in reality are.

□

Using the notations introduced in Table 3.1, the Algorithm 1 provides a pseudo code description of the Outliers Detection in a Single Step.

Algorithm 1: Outliers Detection in a Single Step Algorithm

```
Input  : $X[m, n]$
Output: $S'$
Compute $k = \lceil n/2 \rceil$
Compute $\overline{x}_s = \frac{\sum_{t=1}^{m} x_s^{(t)}}{m}$ for each sensor $1 \leq s \leq n$
Compute $LOF_k(\overline{x}_s)$ for each sensor $1 \leq s \leq n$
Compute $\overline{LOF}_k(\overline{X}) = \frac{\sum_{s=1}^{n} LOF_k(\overline{x}_s)}{n}$
for $s = 1 : n$ do
    if $LOF(\overline{x}_s) < \overline{LOF}_k(\overline{X})$ then
        $S' = S' + \{s\}$;
    end
end
```

On the input we have a dataset X with m observations for each of n sensors. Output of the algorithm is a vector S' that consists of indices of sensors that are found reliable. First, the algorithm calculates $k = \lceil n/2 \rceil$. Then the algorithm computes an average of all m readings for each sensor x_s, $\overline{x}_s$. This is repeated for all n sensors. Now we have a vector $\overline{X}[n]$ of n data points. Third, compute $LOF_k(\overline{x}_s)$ for each element of $\overline{X}$. Next, calculate the average $\overline{LOF}_k(\overline{X})$ of all $LOF_k(\overline{x}_s)$. Then, compare each $LOF_k(\overline{x}_s)$ value with the $\overline{LOF}_k(\overline{X})$. Those sensors whose $LOF_k(\overline{x}_s)$ values are below the average are considered as reliable.

5.1.2 Detection of Outliers Iteratively

A variation of the previous method is the method in which outliers are found iteratively. Unlike the previous method, only one outlier with the maximum LOF value is removed in every cycle.

Values of LOF parameter are recalculated considering the remaining data points after every iteration. A *threshold* and a *ratio* of the maximum LOF value to the minimum (see Equation (5.2)) are used for finding a loop exit-condition.

$$ratio = \frac{\max(LOF_k(X))}{\min(LOF_k(X))} \tag{5.2}$$

According to the [10] those elements that are deep inside a cluster have LOF values close to 1. On the other hand, all outliers have LOF values significantly greater than 1. This gives a basis to expect that if the *ratio* exceeds a certain *threshold*, then x_s is an outlier, where $LOF_k(x_s) = \max(LOF_k(X))$. Initially we set a *threshold* equal 2. Our empirical observations and educated guess show that a value of 2 is good enough to detect obvious outliers. However, keeping the same *threshold* for all iterations often leads to false exclusion of reliable data. On the other hand, if we initially set the *threshold* too high it causes accepting outliers as a reliable data. Again, our extensive empirical evaluations has shown that if we increase the *threshold* every iteration using Equation (5.3) performance of the algorithm is excellent.

$$threshold_{i+1} = \frac{size(X)}{size(X) - 1} * threshold_i, \tag{5.3}$$

where i is the current iteration number.

The Equation 5.3 changes the threshold value slowly, if the size of the set X large, but the threshold is increased faster if the size of X smaller. Thus, as we near the elimination of outliers, loop exit-condition is also close to be satisfied.

Example 4. *For the example considered in Section 5.1.1 four iterations are needed.*

Initialization:

threshold = 2;

X = {1.5, 2, 3, 4, 5, 6, 6.5, 15, 17, 18}

$k = \lceil size(X)/2 \rceil = 5$

Iteration 1:

$LOF_5(X) = \{1.41, 1.23, 0.87, 0.67, 0.87, 1.23, 1.41, 1.93, 2.36, 2.57\}$

$ratio = \frac{2.57}{0.67} = 3.8$

$(ratio > threshold) \implies X = X - \{x_{10}\} = \{1.5, 2, 3, 4, 5, 6, 6.5, 15, 17\}$

$threshold = \frac{9}{8} * 2 = 2.25$

Iteration 2:

$LOF_5(X) = \{1.41, 1.23, 0.87, 0.67, 0.87, 1.23, 1.41, 2.82, 3.37\}$

$ratio = \frac{3.37}{0.67} = 5$

$(ratio > threshold) \implies X = X - \{x_9\} = \{1.5, 2, 3, 4, 5, 6, 6.5, 15\}$

$threshold = \frac{8}{7} * 2.25 = 2.57$

Iteration 3:

$LOF_5(X) = \{1.41, 1.23, 0.87, 0.67, 0.87, 1.23, 1.41, 3.69\}$

$ratio = \frac{3.69}{0.67} = 5.5$

$(ratio > threshold) \implies X = X - \{x_8\} = \{1.5, 2, 3, 4, 5, 6, 6.5\}$

$threshold = \frac{7}{6} * 2.57 = 3$

Iteration 4:

$LOF_5(X) = \{1.41, 1.23, 0.87, 0.67, 0.87, 1.23, 1.41\}$

$ratio = \frac{1.41}{0.67} = 2.1$

$(ratio < threshold) \implies X' = X$

Initially we set a threshold *and calculate* k*, which we keep the same throughout whole the algorithm.*

In first iteration we calculate values of LOF_5 for each element of X. Then, find the ratio of the maximum value of LOF, which is $LOF_5(x_{10})$, to the minimum - $LOF_5(x_4)$. Next, we find that the value of the ratio is greater than threshold, which means that x_{10} is an outlier. Exclude x_{10} from the set X, recalculate a new value of threshold and continue iterations.

In every iteration we recalculate values of LOF_5 for remaining elements of the X set, find new values of the ratio *and* threshold *until the exit-condition is satisfied.*

After completing the last iteration on the input set X, the elements in the set X' are considered to be good data points.

□

Using the notations introduced in Table 3.1, the Algorithm 2 provides a pseudo code description of the Detection of Outliers Iteratively.

Algorithm 2: Outliers Detection Iteratively

```
Input  : $X[m,n], threshold$
Output: $X'$
Compute $\overline{x}_s = \frac{\sum_{t=1}^{m} x_s^{(t)}}{m}$ for each sensor $1 \le s \le n$
Compute k = $\lceil n/2 \rceil$
do
    Compute $LOF_k(\overline{x}_s)$ for each sensor $1 \le s \le n$
    Compute $ratio = \frac{\max(LOF_k(\overline{X}))}{\min(LOF_k(\overline{X}))}$
    if $ratio > threshold$ then
        $\overline{X} = \overline{X} - \overline{x}_s \mid LOF_k(\overline{x}_s) = \max(LOF_k(\overline{X}))$
        $n = n - 1$
    end
    $threshold = \frac{size(\overline{X})}{size(\overline{X})-1} * threshold$
while $ratio > threshold$;
$X' = X$
```

Note that the example 4 excludes computation of average values for each sensor.

5.1.3 Detection of Outliers using Row-Wise Votes

Two outlier detection algorithms presented in previous sections, compute an average of all readings from a sensor. Then outlier detection algorithm identify outliers from these averages. The Algorithm 3 presented in this section identifies outliers from the n readings that are reported at each time instance t. In other words, the Algorithm 3 processes raw data in a row-wise manner. The outlier detection algorithm in Section 5.1.1 is applied to each row of X for identification outliers, that is, $LOF_k(x_s^{(t)})$ values are calculated for every observation $x_s^{(t)}$ of sensor s. The average $\overline{LOF}(x^{(t)})$ value is calculated for every vector of observation $x^{(t)}$. For those $x_s^{(t)}$ whose LOF values indicate them as reliable corresponding elements of $votes[m,n]$ array are assigned 1, otherwise it is assigned 0. This gives a table of votes (see example in the Table 5.3) with average values of all votes for each sensor, which is referred to *reliability*. Sensors that have *reliability* greater than a certain threshold are considered as reliable. After completing the algorithm the vector S' contains numbers of sensors that are found reliable.

Table 5.3: Votes table

Sensors					
instant	s_1	s_2	s_3	s_4	s_5
$t = 1$	1	0	1	1	1
$t = 2$	1	0	1	1	0
$t = 3$	1	0	1	0	0
$t = 4$	1	1	1	1	1
$t = 5$	1	0	1	1	1
reliability	1	0.2	1	0.8	0.6

The advantage of this method is that it allows to set a threshold for excluding outliers; A higher threshold will eliminate readings from a larger number of sensors, but the readings from remaining sensors is expected to be highly reliable. The data shown in the Table 5.3 identify s_1, s_3, s_4, s_5 as reliable, if threshold is set to 0.5. But for same data sensors s_1, s_3, s_4 are found to be reliable data, if the threshold is set to 0.8.

5.1.4 Detection of Outliers using K-means Clustering

The main challenge in using LOF based Outlier Detection algorithms is selection of an optimal value of k. As will be shown in Chaper 7, using Equation (4.4) it is possible to obtain good results, but it is not very good if the number of colluded sensors is much less

Algorithm 3: Outliers Detection Row-Wise Algorithm

```
Input  : $X[m, n], threshold$
Output: $S'$
for $t = 1 : m$ do
    Compute $LOF_k(x_s^{(t)})$ for current observation, where $1 \leq s \leq n$
    Compute $\overline{LOF}_k(x^{(t)}) = \frac{\sum_{s=1}^{n} LOF_k(x_s^{(t)})}{n}$
    for $s = 1 : n$ do
        if $LOF_k(x_s^{(t)}) < \overline{LOF}_k(x^{(t)})$ then
            $votes_s^{(t)} = 1;$
        end
        else
            $votes_s^{(t)} = 0;$
        end
    end
end
for $s = 1 : n$ do
    if $\frac{\sum_{t=1}^{m} votes_s^{(t)}}{m} > threshold$ then
        $S' = S' + \{s\};$
    end
end
```

than $n/2$ (see Figures 4.3 and 4.4). As we have already mentioned, the optimal value of k is $(n_g - 1)$, where n_g is the number of reliable objects.

For that task k-means clustering method has potential to produce better results [4]. The clustering algorithm splits data points into a given number of disjoint sets or clusters in which each data point belongs to one of the clusters. A data point closest to the mean value of all the data in a cluster serves as the *prototype* of the cluster.

Example 5. *Let us consider data points from previous examples* $a(1.5)$, $b(2)$, $c(3)$, $d(4)$, $e(5)$, $f(6)$, $g(6.5)$, $h(15)$, $i(17)$, *and* $j(18)$ *(shown in Table 5.2). The number in the parenthesis represents the value of an observation. First, we set the number of clusters* k[2], *which is in our case* 2 (for reliable and colluded data objects).

[2] Since the symbol k is an inherent component of k-means method terminology, we do not use other symbols. Symbol k is used to denote the number of clusters only in this context. Further, we again use symbol k in the meaning related to *k-distance*.

Then select two data objects furthest apart a(1.5) *and* j(18). *Next, define the initial cluster means (centroids):* $Mean_1$ *and* $Mean_2$*, which are initially equal to the values of the single data objects in the clusters.*

The remaining data objects are now examined one at a time and assigned to the cluster whose centroid is closest to the data point. The cluster centroid of a cluster is recalculated every time a new data object is added to it. After nine steps the data objects are partitioned into two clusters (see Table 5.4).

Table 5.4: k-means clustering: splitting into clusters

	$Cluster_1$		$Cluster_2$	
step	data objects	$mean_1$	data objects	$mean_2$
1	a(1.5)	1.5	j(18)	18
2	a(1.5), b(2)	1.75	j(18)	18
3	a(1.5), b(2), c(3)	2.17	j(18)	18
4	a(1.5), b(2), c(3), d(4)	2.63	j(18)	18
5	a(1.5), b(2), c(3), d(4), e(5)	3.1	j(18)	18
6	a(1.5), b(2), c(3), d(4), e(5), f(6)	3.6	j(18)	18
7	a(1.5), b(2), c(3), d(4), e(5), f(6), g(6.5)	4	j(18)	18
8	a(1.5), b(2), c(3), d(4), e(5), f(6), g(6.5)	4	h(15), j(18)	16.5
9	a(1.5), b(2), c(3), d(4), e(5), f(6), g(6.5)	4	h(15), i(17), j(18)	16.7

Next we verify that each data point has been allocated to the right cluster. To do that, we compare each data point's distance to its current cluster centroid and to its distance the other cluster's centroid. The procedure is illustrated in the Table 5.5.

The entries in the table confirms that the set of data points has been partitioned into two clusters correctly. Otherwise, data points that do not pass verification test are relocated to the other cluster and the centroids are recalculated. This iterative relocation continues until no more data movements are required.

□

To guarantee that the algorithm stops its execution, it is necessary to provide additional conditions.

Table 5.5: k-means clustering: checking

data point	Distance to the $mean_1(4)$	Distance to the $mean_2(16.7)$
a(1.5)	2.5	15.2
b(2)	2	14.7
c(3)	1	13.7
d(4)	0	12.7
e(5)	1	11.7
f(6)	2	10.7
g(6.5)	2.5	10.2
h(15)	11	1.7
i(17)	13	0.3
j(18)	14	1.3

After partitioning the data into two clusters, we know the number of elements in each, and assuming that reliable sensors are more than the colluded sensors, we can select the optimal k using Equation (4.5), that is, $k = (n_g - 1)$, where n_g is the number of elements in the largest cluster.

An obvious question is why one would need to use LOF algorithm, since the data has already been partitioned into two clusters. The answer is simple — k-means clustering algorithm partitions any set of data into a given number of clusters irrespective of their inherent property. But LOF values on a dataset give a measure for each data point that are useful for verification of the partitioning process.

Example (Cont.). *After determined that the size of the largest cluster {1.5, 2, 3, 4, 5, 6, 6.5} is* 7 *we set* $k = 7 - 1 = 6$ *and apply Detection of Outliers in a Single Step Algorithm (see Table 5.6)*

Table 5.6: Data points and corresponding LOF values with optimal k

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
value of x_s	1.5	2	3	4	5	6	6.5	15	17	18
$LOF_6(x_s)$	1.35	1.2	0.89	0.59	0.89	1.2	1.35	2.3	2.75	2.98
$\overline{LOF}_6(X)$	1.55									
$LOF_6(x_s) < \overline{LOF}_6(X)$	true	true	true	true	true	true	true	false	false	false

After we applied Outlier Detection Algorithm in a Single Step we detected the same outliers as in Table 5.2, but now the values of outliers' LOF are greater than using $k = 5$*, which makes outliers to standout for detection.*

□

5.2 Combinations of Algorithms

After the first phase of the algorithm is completed, readings suspected as outliers have been removed. In the second phase of the algorithm a true value is estimated from the remaining sensor-readings for each time instance. At this phase we can choose from IF, RDAM, and statistical estimation method. There are 12 possible combinations of four outlier detection algorithms described in Section 5.1, and three true value estimation algorithms in Chapter 3. In the following section, we describe some of them.

5.2.1 Detection of Outliers and Iterative Filtering

A combination of four methods for detection of outliers presented in Section 5.1 with IF gives four algorithms. Below is an example of combination of *Detection of Outliers in a Single Step* with IF. We call this algorithm *LOF-single-IF*.

Example 6. *We use the same data from the Example 1 of Chapter 3 for this example. We have a dataset* X *with* $m = 3$ *observations from each of* $n = 8$ *sensors. Sensors* s_1 - s_5 *are giving reliable data, while* s_6 - s_8 *are reporting false data, where readings from* s_8 *are equal to the average of all sensors. Table 5.7 illustrates results of LOF-single-IF. The algorithm excludes the impact of skewed readings from* s_6 - s_8*. First phase: calculate average readings* $\overline{x}_s$ *for each sensor. Next, calculate* LOF_k *for each average reading using equation* (4.4) $k = \lceil n/2 \rceil = 8/2 = 4$*. Then, compute* $\overline{LOF}_4(\overline{X})$ *- average of all* $LOF_4(\overline{x}_s)$*. After comparing each* $LOF_4(\overline{x}_s)$ *to* $\overline{LOF}_4(\overline{X})$ *we detect that* s_6*,* s_7*, and* s_8 *are outliers and exclude them from further calculations. In the second phase we apply IF.*

Table 5.7: Detection of Outliers in a Single Step and IF (LOF-single-IF)

		Sensor readings								aggregate values		
	instant	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$			
	$t = 1$	10.73	9.61	9.77	10.20	10.42	15.45	15.13	11.6157			
	$t = 2$	10.75	9.63	9.73	10.21	10.43	15.75	15.93	11.7757			
	$t = 3$	10.74	9.69	9.92	10.32	10.9	15.98	15.39	11.8486			
1 phase	$\overline{X}$	10.7400	9.6433	9.8067	10.2433	10.5833	15.7267	15.4833	11.7467			
	$LOF_4(\overline{X})$	1.0865	1.3148	1.0816	0.6061	1.0912	3.5704	3.3897	2.3257			
	$\overline{LOF}_4(\overline{X})$	1.8083										
	$LOF_4(\overline{x}_s) < \overline{LOF}_4(\overline{X})$	true	true	true	true	true	false	false	false			
2 phase	round	Sensor weights								t=1	t=2	t=3
	1	1	1	1	1	1	-	-	-	10.1460	10.1500	10.3140
	2	3.3993	3.1677	6.3423	457.8755	6.0378	-	-	-	10.1969	10.2064	10.3208
	3	3.9715	2.7916	5.2642	1.3E+05	6.8935	-	-	-	10.2000	10.2100	10.3200
	4	4.0059	2.7742	5.2146	1.4E+10	6.9254	-	-	-	10.2000	10.2100	10.3200

5.2.2 Detection of Outliers and RDAM

We created four combinations of methods of detection of outliers described in Section 5.1 with RDAM. Below is an example of combination showing *Detection of Outliers in a Single Step* with RDAM. We call this algorithm *LOF-single-RDAM*.

Example 7. *Combination of Detection of Outliers in a Single Step and RDAM similarly detects and excludes outlier readings from sensors s_6 - s_8 using LOF technique at the first phase. After removing skewed data, RDAM method is applied in the second phase. Table 5.8 shows how the algorithm excludes the impact of readings from s_6 - s_8.*

Table 5.8: Detection of Outliers in a Single Step and RDAM (LOF-single-RDAM)

		Sensor readings								aggregate values		
	instant	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$			
	$t = 1$	10.73	9.61	9.77	10.20	10.42	15.45	15.13	11.6157			
	$t = 2$	10.75	9.63	9.73	10.21	10.43	15.75	15.93	11.7757			
	$t = 3$	10.74	9.69	9.92	10.32	10.9	15.98	15.39	11.8486			
1 phase	$\overline{X}$	10.7400	9.6433	9.8067	10.2433	10.5833	15.7267	15.4833	11.7467			
	$LOF_4(\overline{X})$	1.0865	1.3148	1.0816	0.6061	1.0912	3.5704	3.3897	2.3257			
	$\overline{LOF}_4(\overline{X})$	1.8083										
	$LOF_4(\overline{x}_s) < \overline{LOF}_4(\overline{X})$	true	true	true	true	true	false	false	false			
2 phase	round	Sensor weights								t=1	t=2	t=3
	1	0.0789	0.4362	0.081	0.3997	0.0041	-	-	-	9.9505	9.9616	10.0483
	2	1.7568	8.4680	29.2168	15.1686	2.5746	-	-	-	9.9191	9.9054	10.0614
	3	1.6380	9.6996	41.1109	1.2574E+01	2.4401	-	-	-	9.8740	9.8552	10.0168
	4	1.4588	13.2041	83.6730	9.2561E+00	2.1298	-	-	-	9.8124	9.7856	9.9560
	5	1.2570	22.0700	485.0123	6.4817E+00	1.7904	-	-	-	9.7731	9.7367	9.9206
	6	1.1477	32.9083	5.50E+04	5.3023E+00	1.6143	-	-	-	9.7700	9.7300	9.9199
	7	1.1388	33.9074	7.78E+08	5.2144E+00	1.6017	-	-	-	9.7700	9.7300	9.9199

5.2.3 Algorithm Based on Combination of Weights and Votes

This algorithm is obtained by combining row-wise votes method described in 5.1.3 and IF. In this combination, actually, there is no clear separation between detection of outliers phase and estimation of true values phase. Having a table of votes (see Table. 5.3) the algorithm uses its values as an indicator to use the corresponding reading in calculations or not to use (see Equation (5.4)).

$$\overline{r}^{(t)} = \sum_{s=1}^{n} w_s * x_s^{(t)} * votes_s^{(t)} \tag{5.4}$$

If an indicator $votes_s^{(t)}$ is 1 then the algorithm uses the corresponding reading in the same way as in IF. Otherwise the reading is simply ignored. Pseudo-code of the algorithm is shown in Algorithm 4.

Algorithm 4: Weights and Votes Algorithm

```
Input  : $X[m, n]$
Output: $\overline{r}$
for $t = 1 : m$ do
    Compute $LOF_k(x_s^{(t)})$ for current observation, where $1 \leq s \leq n$
    Compute $\overline{LOF}_k(x^{(t)}) = \frac{\sum_{s=1}^{n} LOF_k(x_s^{(t)})}{n}$
    for $s = 1 : n$ do
        if $LOF(x_s^{(t)}) < \overline{LOF}_k(x^{(t)})$ then
            $votes_s^{(t)} = 1$;
        end
        else
            $votes_s^{(t)} = 0$;
        end
    end
end
$W \leftarrow 1$;
while $\overline{r}$ notConverged do
    for $t = 1 : m$ do
        $\overline{r}^{(t)} = \sum_{s=1}^{n} w_s * x_s^{(t)} * votes_s^{(t)}$
    end
    Compute $W$;
end
```

5.2.4 Detection of Outliers and Statistical Estimation

For our studies, we combined *Detection of Outliers in a Single Step* and *Detection of Outliers using K-means clustering method*, described in Sections 5.1.1 and 5.1.4, with *Statistical Estimation*, described in Section 3.3. As presented in Chapter 7, these two combinations produce best results in our experiments. There we call these algorithms *LOF-single-weightedSum* and *k-means-LOF-single-weightedSum*, respectively.

Example 8. *Let us apply* LOF-single-weightedSum *for the same data set used in previous examples. At the first phase the algorithm similarly detects and excludes outlier readings from sensors s_6 - s_8. At the second phase Statistical estimation method is applied (see Table 5.9).*

Table 5.9: Detection of Outliers in a Single Step and Statistical estimation

		Sensor readings								aggregate values		
	instant	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$			
	$t = 1$	10.73	9.61	9.77	10.20	10.42	15.45	15.13	11.6157			
	$t = 2$	10.75	9.63	9.73	10.21	10.43	15.75	15.93	11.7757			
	$t = 3$	10.74	9.69	9.92	10.32	10.9	15.98	15.39	11.8486			
1 phase	$\overline{X}$	10.7400	9.6433	9.8067	10.2433	10.5833	15.7267	15.4833	11.7467			
	$LOF_4(\overline{X})$	1.0865	1.3148	1.0816	0.6061	1.0912	3.5704	3.3897	2.3257			
	$\overline{LOF}_4(\overline{X})$	1.8083										
	$LOF_4(\overline{x}_s) < \overline{LOF}_4(\overline{X})$	true	true	true	true	true	false	false	false			
2 phase										t=1	t=2	t=3
	υ_s	0.0001	0.0017	0.0100	0.0044	0.0752	-	-	-			
	weights	0.9161	0.0529	0.0091	0.0207	0.0012	-	-	-	10.6507	10.6699	10.6685

Chapter 6

Estimation of LOF for Simple Datasets

Let $G = \{g_1, g_2, \cdots, g_{n_g}\}$ be a set of readings from n_g 'good' sensors. In our study, a reading from a 'good' sensor means that the sensor, the embedded device, and communication from the embedded device to the destination are *normal*; there is no hardware, software, and network issues have changed or altered the reading. Similarly, let $B' = \{b_1, b_2, \cdots, b_{n_b+1}\}$ be a set of 'bad' readings from $b_b + 1$ sensors. A 'bad' reading may be due to several reasons, including but not limited to, a faulty sensor or a reading altered by an adversary or a colluder. If a collusion attack of the type described in [18] has occurred, then one of readings $b_i \in B'$ is the average of the remaining n_b from B' and all n_g readings from G. Without loss of generality, let us assume that b_{n_b+1} is the average reading introduced by the colluder. Let us define $B = B' \setminus \{b_{n_b+1}\}$, and $av = b_{n_b+1}$, where

$$av = \frac{\sum_{i=1}^{n_g} g_i + \sum_{j=1}^{n_b} b_j}{n_g + n_b}. \tag{6.1}$$

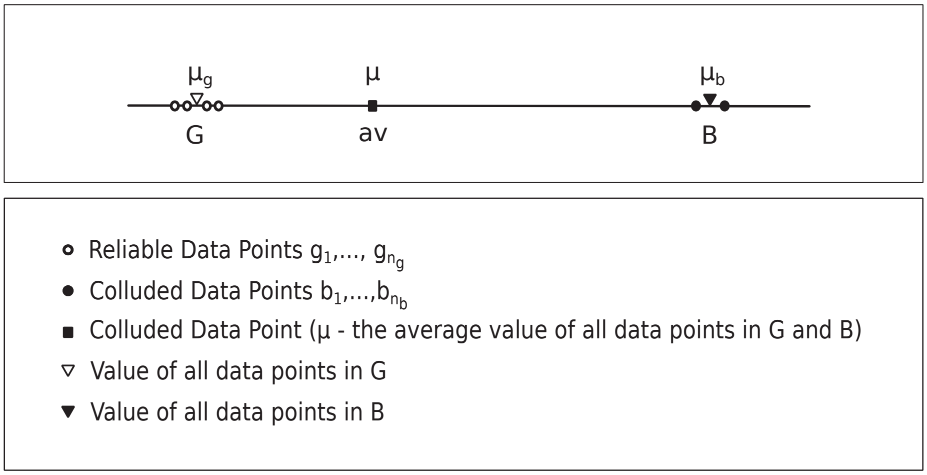

μg
μ
μb
G
av
B
○ Reliable Data Points g_1,..., g_{n_g}
● Colluded Data Points b_1,...,b_{n_b}
■ Colluded Data Point (μ - the average value of all data points in G and B)
▽ Value of all data points in G
▼ Value of all data points in B

Figure 6.1: Collusion Attack Model.

For simplifying presentation, initially we assume that all readings in G are same and equal to μ_g and similarly, we also assume that readings in B are equal to μ_b. Then the value of colluded reading $b_{n_b+1} = av$ is given by,

$$\mu = \frac{n_g \cdot \mu_g + n_b \cdot \mu_b}{n_g + n_b} \tag{6.2}$$

Later, we will discuss how this restriction can be eliminated. By these assumptions, $d(g_i, g_j) = 0$ for $1 \le i, j \le n_g$ and also, $d(b_i, b_j) = 0$ for $1 \le i, j \le n_b$. For real-world situations, there should be considerably more ‘good’ observations than ‘bad’, that is, $n_g > (n_b + 1)$.

In this situation, we want to theoretically determine the value of LOF_k for different range of values for k. For good sensors and av, we need to consider only one case, but for bad sensors, we have to consider three cases: $k < \min\{n_g, n_b\}$, $k = \min\{n_g, n_b\}$, and $\min\{n_g, n_b\} < k < \max\{n_g, n_b\}$.

Lemma 1. *For* $g_i, g_j \in G$*,* and $1 \le k < \max\{n_g, n_b\}$*,*

(1) $d_k(g_i) = 0$*,*

(2) $N_k(g_i) = G \setminus \{g_i\}$*,*

(3) $|N_k(g_i)| = n_g - 1$*,*

(4) $Rd_k(g_j \leftarrow g_i) = \epsilon$*,*

(5) $lrd_k(g_i) = \frac{1}{\epsilon}$*.*

□

Proof. The proofs of all items of the Lemma start from definitions. Since k is less than n_g, the number of objects in G, a k-nearest neighbor $d_k(g_i)$ of any object $g_i \in G$ is another object in G. Because of the assumption that all readings in G are equal and by Definition 3, we have $d_k(g_i) = 0$.

By Definition 4 all objects in G except the object g_i are in $N_k(g_i)$, the set of the k-nearest neighbors of g_i.

The observation just made leads us to the fact that there are $(n_g - 1)$ objects in $N_k(g_i)$. By Definition 5, and the fact that $d_k(g_i) = d(g_i, g_j)$, for all $g_i, g_j \in G$,

$$Rd_k(g_j \leftarrow g_i) = \max(d_k(g_i), d(g_i, g_j), \epsilon) = \epsilon.$$

By Definition 6,

$$lrd_k(g_i) = \frac{|N_k(g_i)|}{\sum_{g_j \in N_k(g_i)} Rd_k(g_j \leftarrow g_i)}.$$

Now using the results we have already proved,

$$lrd_k(g_i) = \frac{n_g - 1}{(n_g - 1)\epsilon} = \frac{1}{\epsilon}.$$

This completes the proofs. □

Theorem 1. *For all* $g_i \in G$, and $1 \leq k < \max\{n_g, n_b\}$, $LOF_k(g_i) = 1$.

□

Proof. By Definition 7, $LOF_k(g_i)$ for $g_i \in G$

$$\begin{align} LOF_k(g_i) &= \frac{\sum_{q \in N_k(g_i)} lrd_k(q)}{|N_k(g_i)| \cdot lrd_k(g_i)} \tag{6.3} \\ &= \frac{\sum_{q \in G \setminus \{g_i\}} lrd_k(q)}{(n_g - 1) \cdot lrd_k(g_i)} \qquad \text{(by Lemma 1)} \tag{6.4} \\ &= \frac{(n_g - 1) \cdot \frac{1}{\epsilon}}{(n_g - 1) \cdot \frac{1}{\epsilon}} = 1 \qquad \text{(by Lemma 1)} \tag{6.5} \end{align}$$

□

Next we present a lemma for the av.

Lemma 2. *For* $g_i \in G$*,* av*,* and $1 \leq k < \max\{n_g, n_b\}$,

(1) $d_k(av) = \frac{n_b \cdot |\mu_b - \mu_g|}{n_g + n_b}$,

(2) $N_k(av) = G$*,*

(3) $|N_k(av)| = n_g$*,*

(4) $Rd_k(g_i \leftarrow av) = \frac{n_b \cdot |\mu_b - \mu_g|}{n_g + n_b}$,

(5) $lrd_k(av) = \frac{n_g + n_b}{n_b \cdot |\mu_b - \mu_g|}$.

□

Proof. A k-nearest object of av can be either in G or in B, which are at distances $|\mu - \mu_g|$ and $|\mu_b - \mu|$, respectively. After substituting expression for μ from Equation (6.2) in $|\mu - \mu_g|$ and $|\mu_b - \mu|$, and then some simplifications we get,

$$|\mu - \mu_g| = \frac{n_b \cdot \left|\mu_b - \mu_g\right|}{n_g + n_b},$$

and

$$|\mu_b - \mu| = \frac{n_g \cdot \left|\mu_b - \mu_g\right|}{n_g + n_b}.$$

Since it was assumed that $n_g > (n_b + 1)$, we have $|\mu - \mu_g| < |\mu_b - \mu|$. Thus, all object in G are at k-distance from av, and

$$d_k(av) = \frac{n_b \cdot \left|\mu_b - \mu_g\right|}{n_g + n_b}.$$

Also, a k-nearest neighbor of av is in G and distance between any two objects in G is 0, by Definition 4 all n_g objects in G are in the set $N_k(av)$, that is, $N_k(av) = G$.

From above discussion, it is clear the value of $|N_k(av)|$ is n_g.

By Definition 5,

$$Rd_k(g_i \leftarrow av) = \max(d_k(av), d(av, g_i), \epsilon).$$

We have already proved that $d(av, g_i) = |\mu - \mu_g|$ and $d_k(av) = |\mu - \mu_g|$. Hence,

$$Rd_k(g_i \leftarrow av) = |\mu - \mu_g| = \frac{n_b \cdot \left|\mu_b - \mu_g\right|}{n_g + n_b}.$$

Finally, by Definition 6,

$$lrd_k(av) = \frac{|N_k(av)|}{\sum_{g_i \in N_k(av)} Rd_k(g_i \leftarrow av)}.$$

After substitution of values of $Rd_k(g_i \leftarrow av)$ in the equation above, we get

$$lrd_k(av) = \frac{n_g}{n_g \cdot |\mu - \mu_g|} = \frac{1}{|\mu - \mu_g|} = \frac{n_g + n_b}{n_b \cdot \left|\mu_b - \mu_g\right|}.$$

□

Theorem 2. *For* $g_i \in G$ and $1 \le k < \max\{n_g, n_b\}$, $LOF_k(av) = \frac{n_b \cdot |\mu_b - \mu_g|}{(n_g + n_b) \cdot \epsilon}$

□

Proof. By Definition 7,

$$\begin{aligned} LOF_k(av) &= \frac{\sum_{q \in N_k(av)} lrd_k(q)}{|N_k(av)| \cdot lrd_k(av)} && (6.6) \\ &= \frac{\sum_{q \in G} lrd_k(q)}{n_g \cdot lrd_k(av)} \quad \text{(by Lemma 2)} && (6.7) \\ &= \frac{n_b \cdot \left|\mu_b - \mu_g\right|}{(n_g + n_b) \cdot \epsilon} \quad \text{(by Lemmas 1 and 2)} && (6.8) \end{aligned}$$

□

Lemma 3. *For* $b_i, b_j \in B$, and $1 \leq k < \min\{n_g, n_b\}$,

(1) $d_k(b_i) = 0$,

(2) $N_k(b_i) = B \setminus \{b_i\}$,

(3) $|N_k(b_i)| = n_b - 1$,

(4) $Rd_k(b_j \leftarrow b_i) = \epsilon$,

(5) $lrd_k(b_i) = \frac{1}{\epsilon}$.

□

Proof is similar to the proof of Lemma 1, and omitted for avoiding duplicity and conserving space.

Theorem 3. *For* $b_i \in B$ and $1 \leq k < \min\{n_g, n_b\}$, $LOF_k(b_i) = 1$,

Proof. By Definition 7, $LOF_k(b_i)$ for $b_i \in B$

$$\begin{aligned} LOF_k(b_i) &= \frac{\sum_{q \in N_k(b_i)} lrd_k(q)}{|N_k(b_i)| \cdot lrd_k(b_i)} && (6.9) \\ &= \frac{\sum_{q \in B \setminus \{b_i\}} lrd_k(q)}{(n_b - 1) \cdot lrd_k(b_i)} \qquad \text{(by Lemma 3)} && (6.10) \\ &= \frac{(n_b - 1) \cdot \frac{1}{\epsilon}}{(n_b - 1) \cdot \frac{1}{\epsilon}} = 1 \qquad \text{(by Lemma 3)} && (6.11) \end{aligned}$$

□

Lemma 4. *For* b_i, $b_j \in B$, av, and $k = min(n_g, n_b)$,

(1) $d_k(b_i) = \left|\frac{n_g \cdot (\mu_b - \mu_g)}{n_g + n_b}\right|$,

(2) $N_k(b_i) = \{av\} \cup B \setminus \{b_i\}$,

(3) $|N_k(b_i)| = n_b$,

(4) $Rd_k(b_j \leftarrow b_i) = \left|\frac{n_g \cdot (\mu_b - \mu_g)}{n_g + n_b}\right|$,

$Rd_k(av \leftarrow b_i) = \left|\frac{n_g \cdot (\mu_b - \mu_g)}{n_g + n_b}\right|$,

(5) $lrd_k(b_i) = \left|\frac{n_g + n_b}{n_g \cdot (\mu_b - \mu_g)}\right|$.

Proof. Since k is equal to the number of objects in B, the k-nearest neighbor of any object in B must be an object that is not in B. Since the object av is closest to the objects in B, it is the k-nearest neighbor of all objects in B. The distance between an object $b_i \in B$ and av is $\frac{n_g \cdot |\mu_b - \mu_g|}{n_g + n_b}$, and equals to $d_k(b_i)$.

Since distance between any objects $b_i \in B$ is 0 and the k-nearest neighbor of b_i is av, then by Definition 4 object av and all objects in B except the object b_i are the k-nearest neighbors. Thus, $N_k(b_i) = \{av\} \cup B \setminus \{b_i\}$.

From the proof above, it is clear that $b_i \in B$, $|N_k(b_i)| = n_b$.

By Definition 5, $Rd_k(b_j \leftarrow b_i) = \max(d_k(b_i), d(b_i, b_j), \epsilon)$ and $Rd_k(av \leftarrow b_i) = \max(d_k(b_i), d(b_i, av), \epsilon)$. Already we have proved that $d_k(b_i) = \frac{n_g \cdot |\mu_b - \mu_g|}{n_g + n_b}$. Hence, $Rd_k(b_j \leftarrow b_i) = \frac{n_g \cdot |\mu_b - \mu_g|}{n_g + n_b}$ and $Rd_k(av \leftarrow b_i) = \frac{n_g \cdot |\mu_b - \mu_g|}{n_g + n_b}$.

By Definition 6,

$$lrd_k(b_i) = \frac{|N_k(b_i)|}{\sum_{p \in N_k(b_i)} Rd_k(p \leftarrow b_i)} \tag{6.12}$$

$$= \frac{|N_k(b_i)|}{\sum_{p \in N_k(b_i) \setminus \{av\}} Rd_k(p \leftarrow b_i) + Rd_k(av \leftarrow b_i)}. \tag{6.13}$$

Substitution of all values of $Rd_k(b_i)$ and simplification leads to the desired result,

$$lrd_k(b_i) = \frac{n_g + n_b}{n_g \cdot \left|\mu_b - \mu_g\right|}. \tag{6.14}$$

□

Theorem 4. *For $g_i \in G$, $b_i \in B$, av* and $k = \min\{n_g, n_b\}$

$LOF_k(b_i) = 1 + \frac{n_g - n_b}{(n_b)^2}$.

Proof of Theorem 4. By definition of $LOF_k(b_i)$ we have,

$$\begin{aligned}
LOF_k(b_i) &= \frac{\sum_{q \in N_k(b_i)} lrd_k(q)}{|N_k(b_i)| \cdot lrd_k(b_i)} \qquad &(6.15)\\
&= \frac{lrd_k(av) + \sum_{q \in B \setminus \{b_i\}} lrd_k(q)}{n_b \cdot lrd_k(b_i)} \text{ (by Lemma 4)} \qquad &(6.16)\\
&= \frac{lrd_k(av)+}{n_b \cdot lrd_k(b_i)} + \frac{\sum_{q \in B \setminus \{b_i\}} lrd_k(q)}{n_b \cdot lrd_k(b_i)} \qquad &(6.17)\\
&= \frac{lrd_k(av)+}{n_b \cdot lrd_k(b_i)} + \frac{(n_b - 1) \cdot lrd_k(b_i)}{n_b \cdot lrd_k(b_i)} \qquad &(6.18)
\end{aligned}$$

Now using results from Lemmas 2 and 4, and doing some simplifications we get,

$$LOF_k(b_i) = 1 + \frac{n_g - n_b}{(n_b)^2}$$

□

Lemma 5. *For b_i, $b_j \in B$, av,* and, $\min\{n_g, n_b\} < k < \max\{n_g, n_b\}$

(1) $d_k(b_i) = |\mu_b - \mu_g|$,

(2) $N_k(b_i) = G \cup \{av\} \cup B \setminus \{b_i\}$,

(3) $|N_k(b_i)| = n_g + n_b$,

(4) $Rd_k(b_j \leftarrow b_i) = |\mu_b - \mu_g|$,

$Rd_k(av \leftarrow b_i) = |\mu_b - \mu_g|$,

$Rd_k(g_i \leftarrow b_i) = |\mu_b - \mu_g|$,

(5) $lrd_k(b_i) = \frac{1}{|\mu_b - \mu_g|}$.

□

Proof. Since k is greater than the number of objects in B, a k-nearest neighbor of any object in B is an object in G, which gives k-distance, $d_k(b_i) = |\mu_b - \mu_g|$.

Since distance between any two objects in B is 0, a k-nearest neighbor is in G, and distance between any objects in G is also 0, then by Definition 4 all objects in G, object av, and all objects in B except objects b_i are in the k-nearest objects set denoted as $N_k(b_i)$.

Since all objects in G, object av, and all objects in B except the subject b_i are in $N_k(b_i)$, the number of members of $N_k(b_i)$ is calculated as $n_g + 1 + (n_b - 1) = n_g + n_b$.

By Definition 5,

$$Rd_k(b_j \leftarrow b_i) = \max(d_k(b_i), d(b_i, b_j), \epsilon), \tag{6.19}$$

$$Rd_k(av \leftarrow b_i) = \max(d_k(b_i), d(b_i, av), \epsilon), \tag{6.20}$$

$$Rd_k(g_i \leftarrow b_i) = \max(d_k(b_i), d(b_i, g_i), \epsilon). \tag{6.21}$$

We have already proved that $d_k(b_i) = |\mu_b - \mu_g|$. Hence, $Rd_k(b_j \leftarrow b_i) = |\mu_b - \mu_g|$, $Rd_k(av \leftarrow b_i) = |\mu_b - \mu_g|$, and $Rd_k(g_i \leftarrow b_i) = |\mu_b - \mu_g|$.

By Definition 6,

$$\begin{aligned} lrd_k(b_i) &= \frac{|N_k(b_i)|}{\sum_{p \in N_k(b_i)} Rd_k(p \leftarrow b_i)} && (6.22) \\ &= \frac{|N_k(b_i)|}{\sum_{p \in B \setminus b_i} Rd_k(p \leftarrow b_i) + Rd_k(av \leftarrow b_i) + \sum_{q \in G} Rd_k(q \leftarrow b_i)}. && (6.23) \end{aligned}$$

Now substitution of expressions we have already found for $Rd_k(b_j \leftarrow b_i)$, $Rd_k(av \leftarrow b_i)$, and $Rd_k(g_i \leftarrow b_i)$ in the expression above, we get the desired result:

$$lrd_k(b_i) = \frac{1}{|\mu_b - \mu_g|}. \tag{6.24}$$

□

Theorem 5. *For* $k > \min\{n_g, n_b\}$, $g_i \in G$, $b_j \in B$, *and* av
$LOF_k(b_i) = \frac{n_g \cdot |\mu_b - \mu_g|}{(n_g + n_b) \cdot \epsilon} + \frac{1}{n_b} + \frac{n_b - 1}{n_g + n_b}$.

Proof of Theorem 5. By definition of $LOF_k(b_i)$ we have,

$$\begin{aligned}
LOF_k(b_i) &= \frac{\sum_{q\in N_k(b_i)} lrd_k(q)}{|N_k(b_i)| \cdot lrd_k(b_i)} && (6.25)\\
&= \frac{\sum_{q\in G} lrd_k(q) + lrd_k(av) + \sum_{q\in B\setminus\{b_i\}} lrd_k(q)}{(n_g+n_b)\cdot lrd_k(b_i)} \quad \text{(by Lemma 5)} && (6.26)\\
&= \frac{\sum_{q\in G} lrd_k(q)}{(n_g+n_b)\cdot lrd_k(b_i)} + \frac{lrd_k(av)}{(n_g+n_b)\cdot lrd_k(b_i)} && (6.27)\\
&+ \frac{\sum_{q\in B\setminus\{b_i\}} lrd_k(q)}{(n_g+n_b)\cdot lrd_k(b_i)} \quad \text{(by Lemmas 1, 2, and 5)} && (6.28)\\
&= \frac{n_g \cdot |\mu_b - \mu_g|}{(n_g+n_b)\cdot \epsilon} + \frac{1}{n_b} + \frac{n_b - 1}{n_g + n_b} && (6.29)
\end{aligned}$$

□

Chapter 7

Experimental Results

The experimental evaluation of the proposed algorithms aims to verify their reliability and efficiency in the presence of a collusion attack. Since the RDAM performs best in the presence of collusion attack [18], we use this algorithm's performance as a benchmark for measuring that of the proposed algorithms. In order to evaluate the accuracy of the investigated algorithms in the presence of collusion attack, we assume that an attacker compromises c sensor nodes out of n nodes in a cluster and $c < n/2$.

7.1 Experimental Settings

Unless otherwise stated, we reconstruct the same conditions that are used in [18] for generating datasets for evaluation of the proposed algorithms. We used Matlab 9.0 (R2015b) as our programing platform. We model a cluster of $n = 20$ sensor nodes and assume that these 20 nodes reports their readings to a cluster head or data aggregator. Each sensor gathers $m = 400$ readings before sending them to the cluster head. For consistency and fair comparisons, the original signal is generated using the Equation (7.1), which has been used in [18].

$$r^{(t)} = 5 \cdot n \cdot sin\left(\frac{\sqrt{2} \cdot t}{2 \cdot \pi}\right), \tag{7.1}$$

where $t \in \{1..m\}$.

To generate synthetic data sets X, we add noise $e_s^{(t)}$ to the original signal $r^{(t)}$ as shown by Equation (7.2)

$$x_s^{(t)} = r^{(t)} + e_s^{(t)}. \tag{7.2}$$

We use Gaussian distribution for both stochastic noise for readings from reliable sensors and correlated noise for readings from colluded sensors. In our experiments for the sensors that are considered reliable we add the same biased error as in [18] and is defined by Equations (7.3) and (7.4).

$$b_s \sim \mathcal{N}(0, \sigma_b^2) \tag{7.3}$$

$$e_s^{(t)} \sim \mathcal{N}(b_s, \sigma^2) \tag{7.4}$$

Moreover, we conduct experiments with another type of additive noise signals for the reliable sensors. Equation (7.5) describes this type of noise signal. Note that the baseline variance of the noise is increased by a multiplicative factor of $\sqrt{s}$ for the sensor with id s.[3]

$$e_s^{(t)} \sim \mathcal{N}(b_s, \sqrt{s} \cdot \sigma^2) \tag{7.5}$$

For errors of colluded readings we use the same models, which are described by Equations (7.6) and (7.7) as in [18]. For noise we also use a modified model described by Equation (7.8). In this modified model the multiplicative factor $\sqrt{s}$ is removed (or set to 1).

$$b'_s \sim \mathcal{N}(0, 3 \cdot \sigma_b^2), \tag{7.6}$$

$$e_s^{(t)} \sim \mathcal{N}(b'_s, 5 \cdot \sqrt{s} \cdot \sigma^2) \tag{7.7}$$

$$e_s^{(t)} \sim \mathcal{N}(b'_s, 5 \cdot \sigma^2) \tag{7.8}$$

Parameters used in our experiments:

- each experiment is repeated 200 times and results are averaged;
- for all experiments $\sigma_b^2 = 4$;
- value of standard deviation σ in each experiment is varied from 1 to 5;
- number of compromised sensors c in each experiment is varied from 3 to 8.

[3] We noted in Chapter 2 that sensor networks are often used in hostile environment. So it could happen that every next sensor is exposed to greater external influence and therefore noise increases accordingly.

We use only one discriminant function in our experiments $g(d) = d^{-1}$ (see details in [18]) for iterative filtering phase of RDAM algorithm. The selection of this discriminant functions is compelled by the fact that out of four discriminant functions used for evaluations in [18], this function had shown the best performance.

For fair comparisons performance of the RDAM and our proposed algorithms, we used same sets of generated data for both methods.

7.2 Performance Evaluation Metric

To measure performance of the proposed algorithms we use two metrics: *Root Mean Squared error* (RMSE) and the *Maximum Error* (ME) defined by Equations (7.9) and (7.10), respectively. The RDAM algorithm used RMSE for evaluating performance of the algorithm against existing algorithms. We added ME to the metric for capturing worst-case performance.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{m}(r^{(t)} - \overline{r}^{(t)})^2}{m}}, \tag{7.9}$$

where $r^{(t)}$ and $\overline{r}^{(t)}$ are true values and estimate of true values of the signal at time t, respectively.

$$ME = \max_{\forall t}(|r^{(t)} - \overline{r}^{(t)}|), \tag{7.10}$$

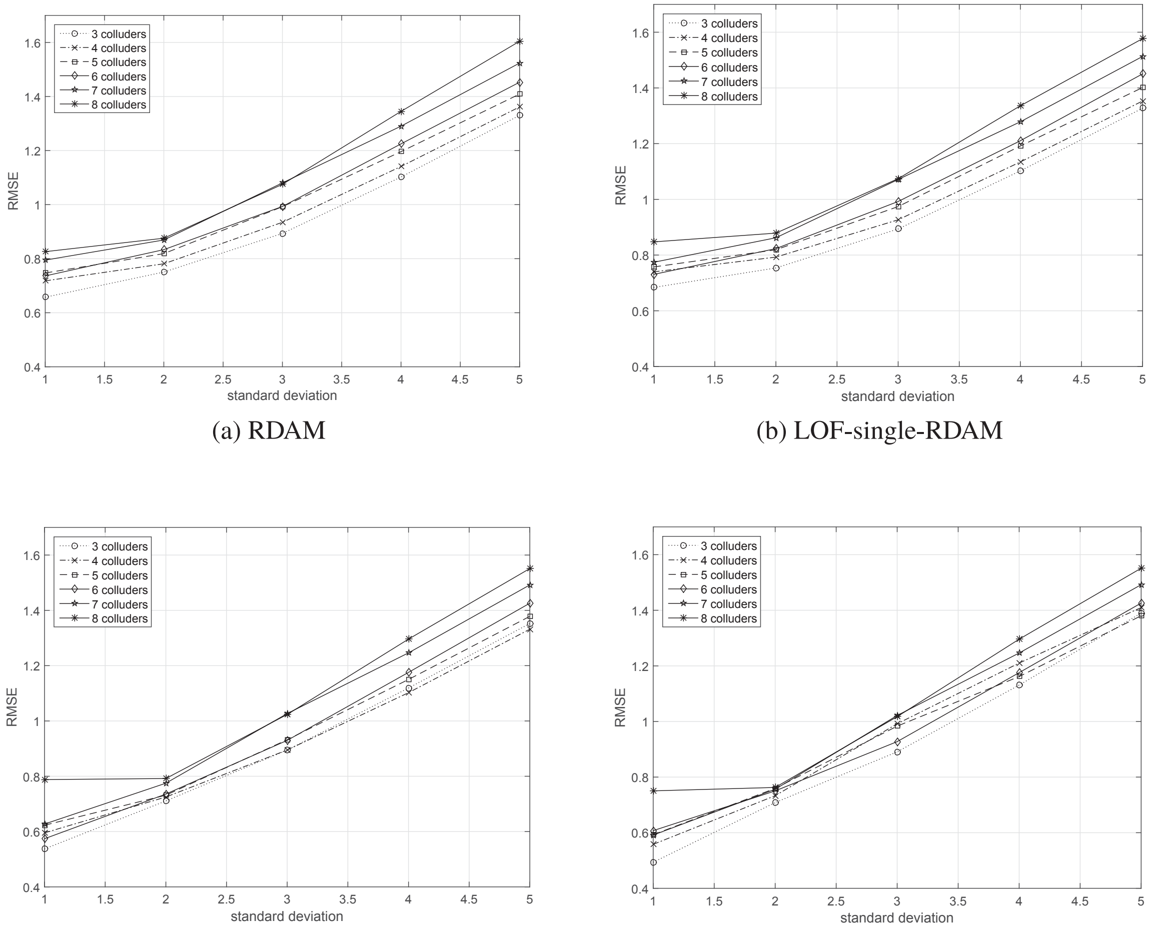
where max is the function that returns maximum value for all t and that $|.|$ is absolute value function. The efficiency is evaluated by the number of iterations needed for the IF algorithm to converge.

7.3 Results

We present some of our typical observations from our extensive simulation results. For ease of comparison we illustrate our results as line plots. Moreover, we included corresponding data in the table format for those who want to examine the results more closely and critically.

We evaluated all the proposed algorithms, but we report results only for those algorithms that have produced best results. It should be noted that with rare exceptions, all our algorithms performed better than the RDAM algorithm.

The first set of results is for datasets that are (statistically) identical to those used for experiments in [18]. For visual display of the results, Figure 7.1 show plots RSME vs standard deviation for our proposed algorithms and RDAM algorithms. For detailed comparisons data is shown in Table 7.1. It is clear from the table and the plots that performance of our algorithm is as good or better than that of RDAM. For the case with 8 colluders the RMSE for our methods are 3-13% lower than that of the RDAM algorithm.



(a) RDAM

(b) LOF-single-RDAM

(c) LOF-single-weightedSum

(d) k-means-LOF-single-weightedSum

Figure 7.1: RMSE with original conditions of collusion attack

Table 7.1: Evaluation parameters for algorithms with original conditions of collusion attack[4]

	$\sigma = 1$				$\sigma = 2$				$\sigma = 3$				$\sigma = 4$				$\sigma = 5$			
	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated
colluders = 3																				
RDAM	0.6574	1.4508	30.1	-	0.7505	2.0504	18.2	-	0.8934	2.6717	13.2	-	1.1022	3.3903	11.6	-	1.3304	4.1212	10.5	-
LOF-single-RDAM	0.6838	1.4883	29.4	3.6	0.7538	2.0507	18.1	2.3	0.8938	2.6747	13.1	2.1	1.1022	3.3940	11.1	2.3	1.3273	4.1203	9.7	2.5
LOF-single-weightedSum	0.5372	1.2022	-	3.6	0.7108	1.9284	-	2.3	0.8947	2.6533	-	2.1	1.1192	3.3929	-	2.3	1.3524	4.1712	-	2.5
k-means-LOF-single-weightedSum	0.4936	1.1485	-	3.5	0.7083	1.9174	-	2.2	0.8908	2.6435	-	2	1.1309	3.4292	-	2	1.3899	4.2508	-	2
colluders = 8																				
RDAM	0.8259	1.9498	33.5	-	0.8757	2.4223	20	-	1.0750	3.2259	15.4	-	1.3449	4.0963	13.6	-	1.6044	4.9986	12.2	-
LOF-single-RDAM	0.8473	1.9845	30.5	9.3	0.8788	2.4518	18.9	8.2	1.0742	3.2310	13.3	8	1.3359	4.0855	11.6	8	1.5771	4.9631	10.1	8
LOF-single-weightedSum	0.7880	1.8029	-	9.3	0.7918	2.2175	-	8.2	1.0235	3.1051	-	8	1.2960	3.9414	-	8	1.5513	4.8549	-	8
k-means-LOF-single-weightedSum	0.7505	1.6621	-	8.7	0.7628	2.1627	-	8	1.0173	3.0911	-	8	1.2960	3.9414	-	8	1.5513	4.8549	-	8

Figures 7.2, 7.3, 7.4 and Table 7.2 report results when sensor readings for reliable and compromised sensors is generated by noise models described by Equations (7.5) and (7.8), respectively. It is not difficult to see from the plots that the proposed algorithms reduce RMSE values significantly. The accuracy of the proposed algorithms is especially noticeable for larger number of colluders and higher standard deviations. As can be seen from Figure 7.6, for the case with 8 colluders the RMSE for our methods is 18% to 53% lower when standard deviation is varied from 1 to 5. The RMSE improvement linearly grows from about 18% to about 50% as a standard deviation increases from 1 to 3. For standard deviation 3 to 5 the improvement rate slows down, but improvement continues.

Table 7.2: Evaluation parameters for algorithms with changed conditions of collusion attack[4]

	$\sigma = 1$				$\sigma = 2$				$\sigma = 3$				$\sigma = 4$				$\sigma = 5$			
	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated	rmse	maximum error	iterations	colluders estimated
colluders = 3																				
RDAM	1.0059	2.7799	4.6	-	1.7168	5.1185	4.4	-	2.5316	7.6470	4.5	-	3.3142	10.0966	4.4	-	4.0603	12.6340	4.5	-
LOF-single-RDAM	1.0014	2.7872	5.2	6.2	1.7168	5.1185	4.7	5.7	2.5316	7.6470	4.8	4.9	3.3142	10.0966	4.8	3.9	4.0603	12.6340	4.9	3.5
LOF-single-weightedSum	0.9692	2.8874	-	6.2	1.7075	5.3249	-	5.7	2.4209	7.6926	-	4.9	3.0739	9.8150	-	3.9	3.7481	11.9457	-	3.5
k-means-LOF-single-weightedSum	0.9573	2.8315	-	5.6	1.6414	5.1075	-	4.6	2.3447	7.3742	-	4	3.0361	9.6776	-	3.6	3.7237	11.9046	-	3.3
colluders = 8																				
RDAM	2.4005	4.4884	7.5	-	3.6886	7.5213	4.9	-	5.1274	10.7595	4.2	-	6.6793	14.1918	4	-	8.1766	17.3392	3.9	-
LOF-single-RDAM	2.2845	4.3970	10	6.8	2.5574	6.9895	21.3	9.1	3.2860	10.1231	20.6	9.6	4.1614	13.1627	18.6	9.4	5.0825	16.0328	19.6	8.9
LOF-single-weightedSum	1.9412	4.1053	-	6.8	2.5341	6.7288	-	9.1	2.6409	8.2082	-	9.6	3.1995	10.2005	-	9.4	3.8542	12.3122	-	8.9
k-means-LOF-single-weightedSum	1.9694	4.1640	-	6.7	2.3306	6.2936	-	8.6	2.5331	7.9323	-	8.9	3.1282	10.0773	-	8.7	3.7671	12.0540	-	8.4

Assessing performances of the algorithms in terms of the maximum error (defined by (7.10)), it can be concluded that all our proposed methods give approximately the

[4] For conserving space we provide data only for cases with 3 and 8 colluders.

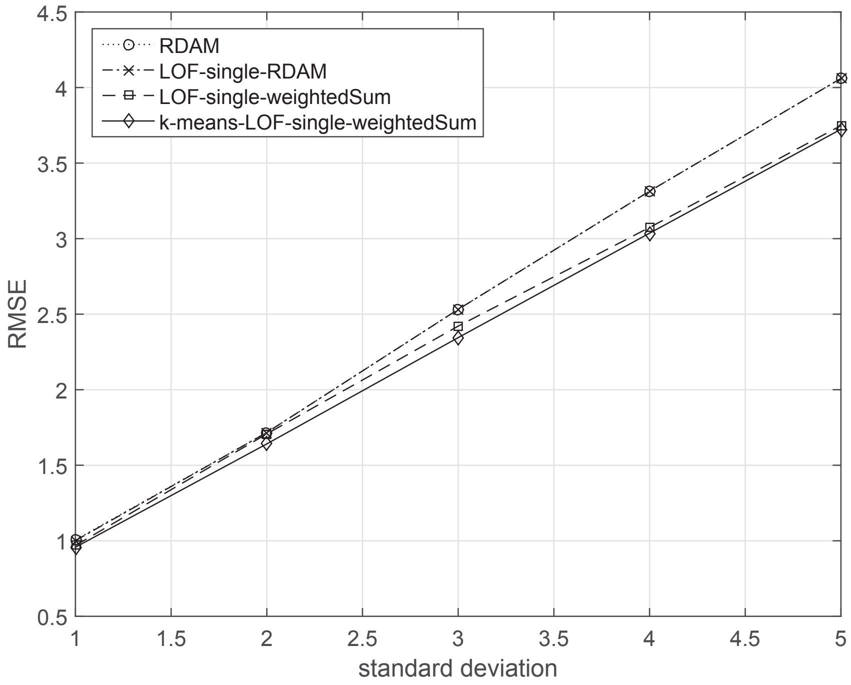


Figure 7.2: RMSE with changed conditions of collusion attack with 3 colluders

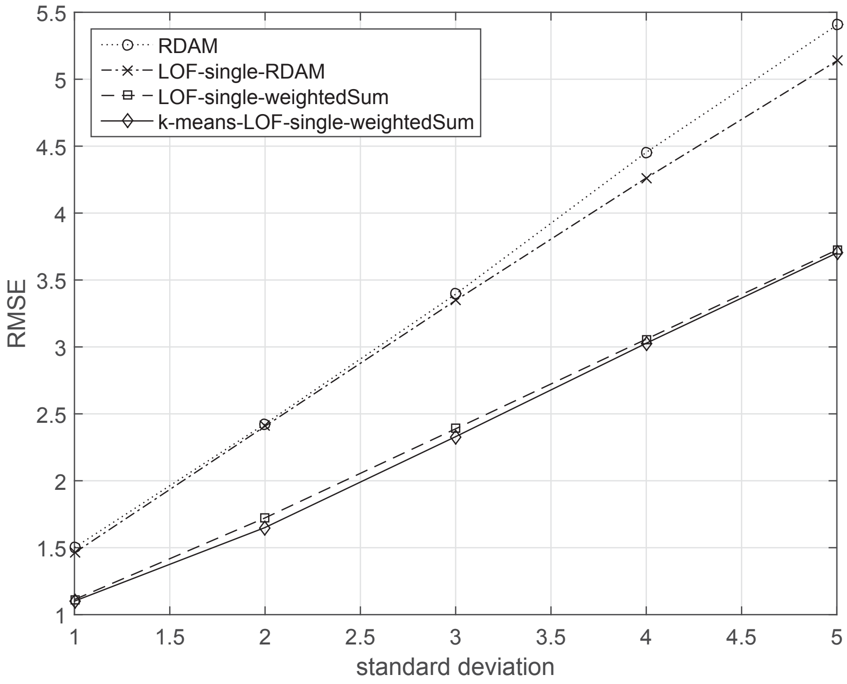


Figure 7.3: RMSE with changed conditions of collusion attack with 5 colluders

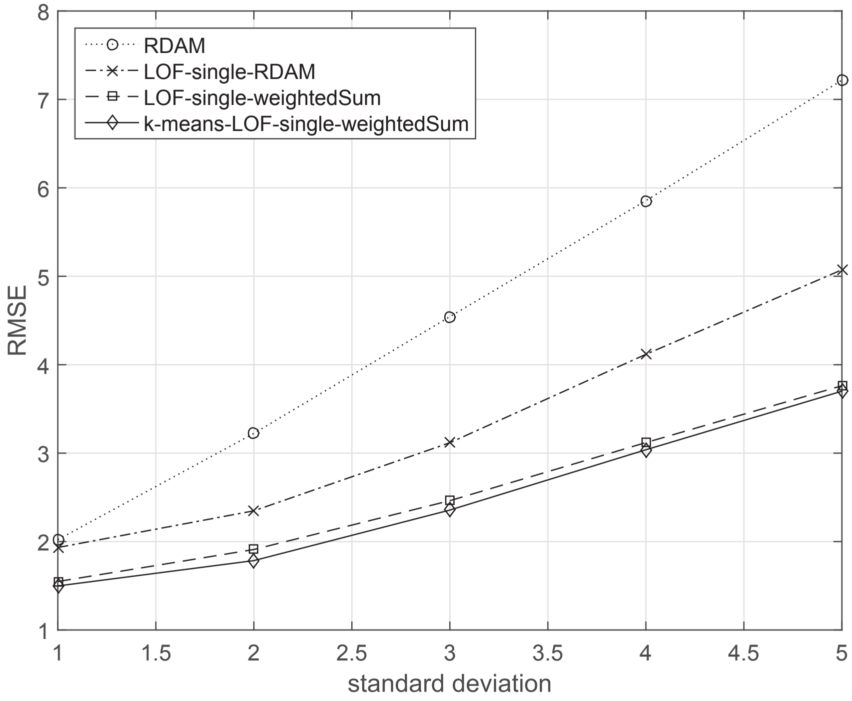


Figure 7.4: RMSE with changed conditions of collusion attack with 7 colluders

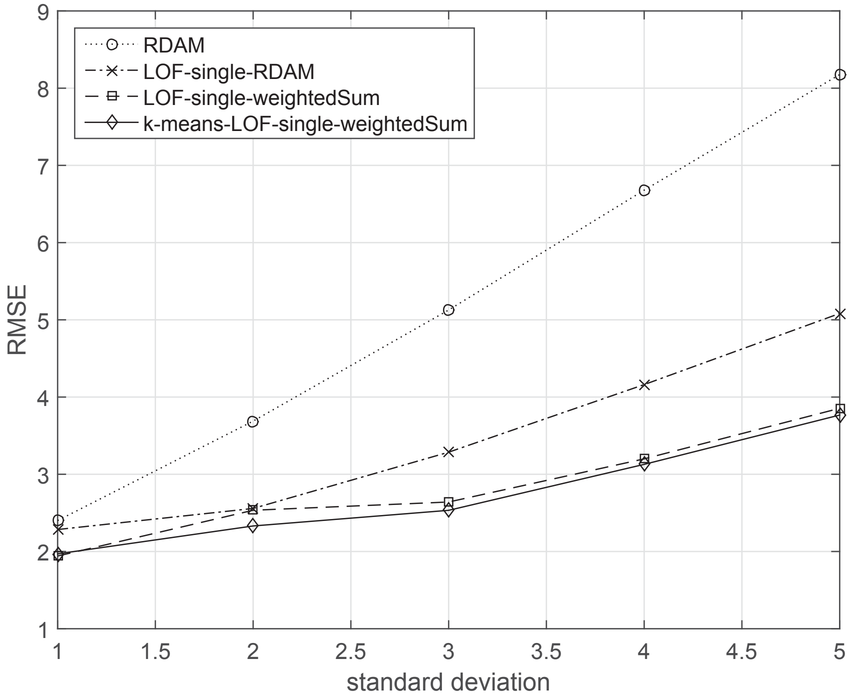


Figure 7.5: RMSE with changed conditions of collusion attack with 8 colluders

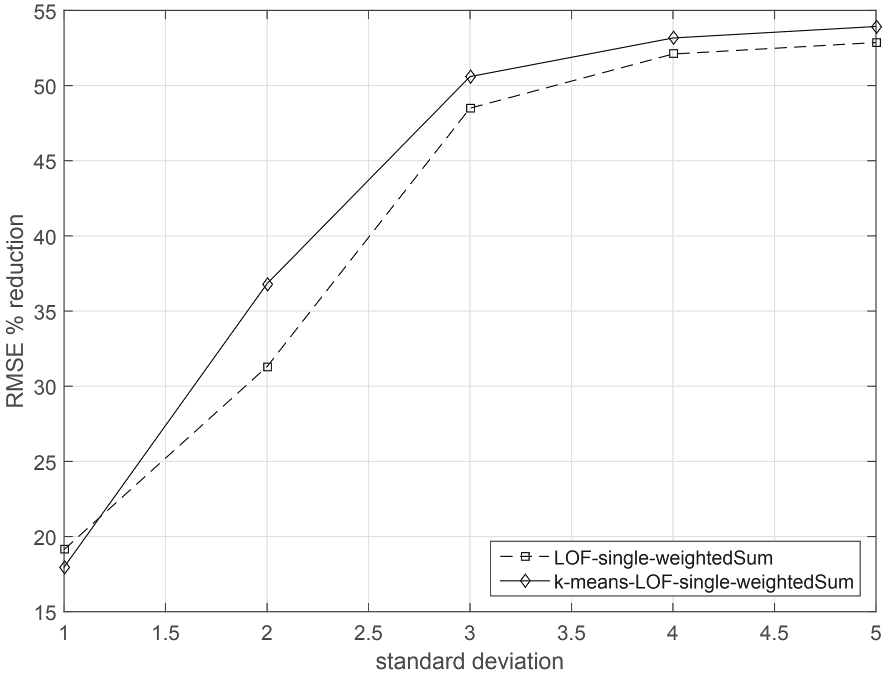
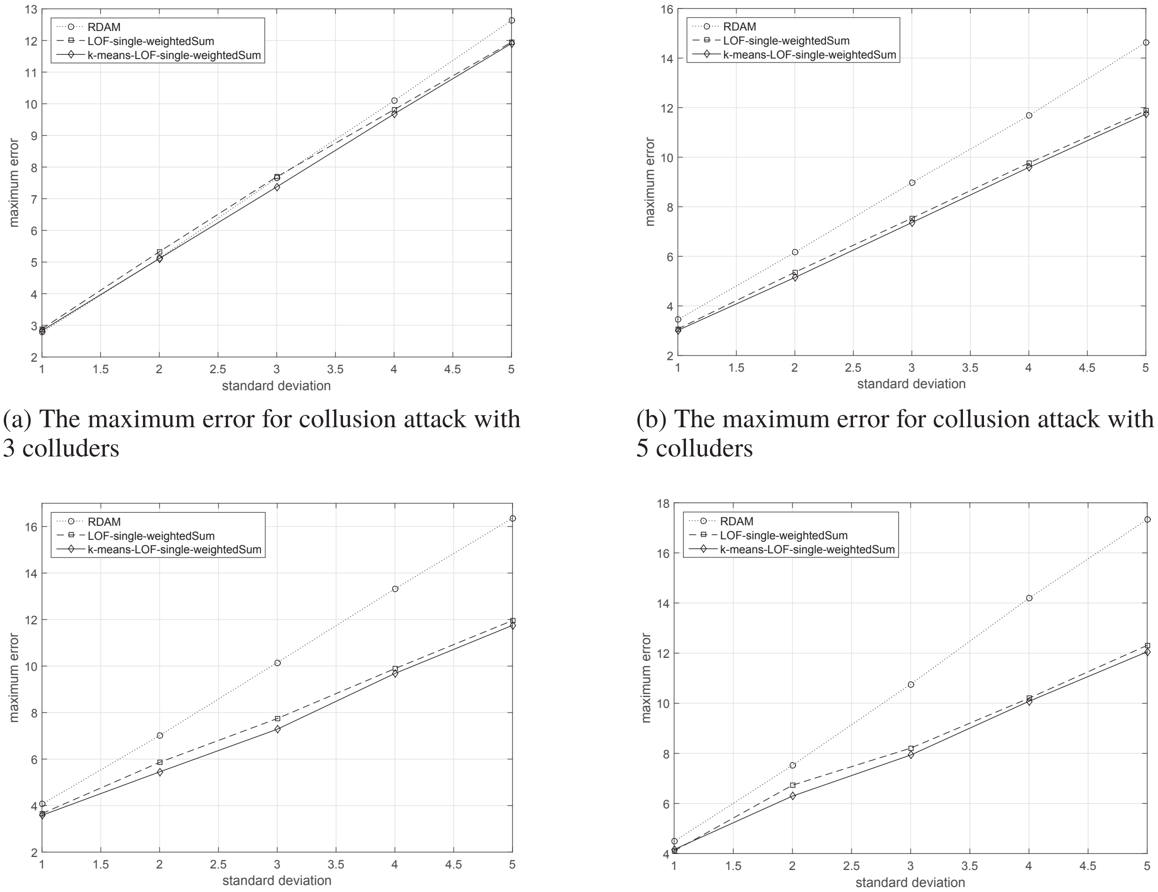


Figure 7.6: RMSE reduction with changed conditions of collusion attack with 8 colluders comparing to RDAM

same level of maximum error, except *LOF-single-weightedSum* and *k-means-LOF-single-weightedSum*, which show significantly lower ME as can be seen in Figure 7.7 and Table 7.2.

Figures 7.2, 7.3, 7.4, 7.5, and 7.7 also show that two of our methods — *LOF-single-weightedSum* and *k-means-LOF-single-weightedSum* — outperforms all others. For a given level of added noise level and number of colluders, their RMSE and ME are almost identical. The reason for their success is their ability to identify colluded readings with fairly high degree of accuracy (see Tables 7.1 and 7.2). For original conditions of collusion attack for $\sigma = \{3, 4, 5\}$ and 8 colluders the estimated number of colluders is exactly 8.

Recall that algorithms *LOF-single-weightedSum* and *k-means-LOF-single-weightedSum* do not use iterative filtering. Thus, they reduce computation time and do not suffer from weaknesses of the iterative filtering algorithms.



(a) The maximum error for collusion attack with 3 colluders

(b) The maximum error for collusion attack with 5 colluders

(c) The maximum error for collusion attack with 7 colluders

(d) The maximum error for collusion attack with 8 colluders

Figure 7.7: The maximum error with changed conditions of collusion attack

Chapter 8

Conclusion

In this work, we concentrated on the development of outlier detection methods. We described the conception of local outlier factor, which plays a key role in our methods. Then we proposed several two phase outlier detection algorithms.

The main feature of our algorithms is detection and removal of outliers before estimation true value. First, this increases the accuracy by removing the influence of outliers in aggregated result. Second, having only reliable data true values can be estimated non-iteratively, which decreases computational cost.

We used the RDAM [18] as the benchmark for comparison since it has shown best results against original collusion attack. We tested RDAM and our algorithms against collusion attacks considered in [18]. Moreover, we created examples of collusion attack scenario that were not considered in [18]. Under these novel attack scenario the proposed algorithms performed much better than the RDAM.

We have presented experimental results that show that (1) the estimates have higher accuracy than RDAM and (2) the algorithms have better efficiency than that of the RDAM. We observed that detecting local outliers for sensor readings using LOF is efficient.

Since computation of local outlier factor is not a low computational-cost method, we have found expressions that are useful for calculating outlier factors for simple data. While conducting this research we changed some original definitions and discovered promising opportunities that can lead to new clustering methods, which will be a scope for future work.

References

[1] Mahmoud Abou-Nasr. *Real world data mining applications*. Springer Publishing Company, Incorporated, 2014.

[2] Erman Ayday and Faramarz Fekri. Iterative trust and reputation management using belief propagation. *IEEE Transactions on Dependable and Secure Computing*, 9(3):375–386, 2012.

[3] V. Barnett and T. Lewis. *Outliers in statistical data*. John Wiley, 1994.

[4] Christopher M. Bishop. *Pattern recognition and machine learning (Information science and statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[5] Joel Branch, Boleslaw Szymanski, Chris Giannella, Ran Wolff, and Hillol Kargupta. In-network outlier detection in wireless sensor networks. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, ICDCS '06, pages 51–, Washington, DC, USA, 2006. IEEE Computer Society.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. *LOF: Identifying density-based local outliers*. ACM, Dalles, Texas, USA, 2000.

[7] V. Chatzigiannakis, S. Papavassiliou, M. Grammatikou, and B. Maglaris. Hierarchical anomaly detection in distributed large-scale sensor networks. In *Proceedings of the 11th IEEE Symposium on Computers and Communications*, ISCC '06, pages 761–767, Washington, DC, USA, 2006. IEEE Computer Society.

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

[9] John A. Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition, 1975.

[10] D. Hawkins. *Identification of outliers*. Chapman and Hall, London, 1980.

[11] David J Hill, Barbara S Minsker, and Eyal Amir. Real-time Bayesian anomaly detection for environmental sensor data. In *Proceedings of the Congress-International Association for Hydraulic Research*, volume 32, page 503. Citeseer, 2007.

[12] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[13] Raja Jurdak, X.Rosalind Wang, Oliver Obst, and Philip Valencia. Wireless sensor network anomalies: Diagnosis and detection strategies. In Andreas Tolk and Lakhmi C. Jain, editors, *Intelligence-Based Systems Engineering*, volume 10 of *Intelligent Systems Reference Library*, pages 309–325. Springer Berlin Heidelberg, 2011.

[14] David C LeBlanc. *Statistics*. Jones and Bartlett, 2004.

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

[16] Stefano Mizzaro. Quality control in scholarly publishing: A new proposal. *Journal of the American Society for Information Science and Technology*, 54(11):989–1005, 2003.

[17] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. Sensor network data fault types. *ACM Trans. Sen. Netw.*, 5(3):25:1–25:29, June 2009.

[18] M. Rezvani, A. Ignatovich, E. Bertino, and S. Jha. *Secure data aggregation technique for wireless sensor networks in the presence of collusion attacks*, volume 12. IEEE Transactions on Dependable and Secure Computing, January/February 2015.

[19] Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Trans. Sen. Netw.*, 6(3):23:1–23:39, June 2010.

[20] 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.

[21] X. Rosalind Wang, Joseph T. Lizier, Oliver Obst, Mikhail Prokopenko, and Peter Wang. Spatiotemporal anomaly detection in gas monitoring sensor networks. In *Proceedings of the 5th European Conference on Wireless Sensor Networks*, EWSN'08, pages 90–105, Berlin, Heidelberg, 2008. Springer-Verlag.

[22] Weili Wu, Xiuzhen Cheng, Min Ding, Kai Xing, Fang Liu, and Ping Deng. Localized outlying and boundary data detection in sensor networks. *IEEE Trans. on Knowl. and Data Eng.*, 19(8):1145–1157, August 2007.

[23] Yi-Kuo Yu, Yi-Cheng Zhang, Paolo Laureti, and Lionel Moret. Decoding information from noisy, redundant, and intentionally distorted sources. *Physica A: Statistical Mechanics and its Applications*, 371(2):732–744, 2006.

[24] Y. Zhang, N. A. S. Hamm, N. Meratnia, A. Stein, M. van de Voort, and P. J. M. Havinga. Statistics-based outlier detection for wireless sensor networks. *Int. J. Geogr. Inf. Sci.*, 26(8):1373–1392, August 2012.

[25] Yang Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *Commun. Surveys Tuts.*, 12(2):159–170, April 2010.

[26] Yan-Bo Zhou, Ting Lei, and Tao Zhou. A robust ranking algorithm to spamming. *EPL*, 94(4):48002, 2011.