# Resilient Aggregation: Statistical Approaches

Levente Buttyán Péter Schaffer[1] István Vajda
Budapest University of Technology and Economics
Laboratory of Cryptography and Systems Security (CrySyS)
{buttyan, schaffer, vajda}@crysys.hu

**Abstract**

In typical sensor network applications, the sensors are left unattended for a long period of time. In addition, due to cost reasons, sensor nodes are usually not tamper resistant. Consequently, sensors can be easily captured and compromised by an adversary. Once compromised, a sensor can send authentique messages to other nodes and to the base station, but those messages may contain arbitrary data created by the adversray (e.g., bogus measurments). A similar effect can be achieved by manipulating the physical environment of uncompromised sensors so that they measure false values. Bogus data introduced by the adversary may considerably distort the output of the aggregation function at the base station, and may lead to wrong decisions. The goal of resilient aggregation is to perform the aggregation correctly despite the possibility of the above mentioned attacks.

In this paper, we give an overview of the state-of-the-art in resilient aggregation in sensor networks, and briefly summarize the relevant techniques in the field of mathematical statistics. In addition, we introduce a particular approach for resilient aggregation in more details. This approach is based on RANSAC (RAndom SAmple Consensus), which we adopted for our purposes. We also present some initial simulation results showing that our RANSAC based approach can tolerate a high percentage of compromised nodes.

## 1 Introduction

Sensor networks are distributed systems, consisting of hundreds or thousands of tiny, low-cost, low-power sensor nodes and one (or a few) more powerful base station(s). These networks are designed to interact with the physical environment. Typically, sensors measure some physical phenomena (e.g., temperature and humidity) and send their measurements to the base station using wireless communications. The base station performs data processing functions and provides gateway services to other networks (e.g., the Internet).

Sensor nodes are able to transmit messages only within a short communication range, therefore, it is envisioned that the sensors form a multi-hop network in which the nodes forward messages on behalf of other nodes toward the base station. The typical toplogy of a sensor network is a tree with the base station at the root (see Figure x.1 for illustration). In order to reduce the total number of messages sent by the sensors, in-network processing may be employed, whereby some sensor nodes perform data aggregation functions. Aggregator nodes collect data from surrounding sensors, process the collected data locally, and transmit only a single, aggregated

[1] Corresponding author.

message toward the base station. Finally, the base station computes a single aggregated value (e.g., average, minimum, or maximum) from the data received from the network.

After deployment, sensors are typically left unattended for a long period of time. In order to keep their cost acceptable, common sensor nodes are not tamper resistant. This means that they can be captured and compromised at a reasonable cost. Therefore, we cannot assume that common sensors attacked by a determined adversary are able to protect any secret cryptographique elements (e.g., secret keys). Once a sensor is compromised, it can send authentique messages to other nodes and to the base station, but those messages may contain arbitrary data created by the adversray (e.g., bogus measurments).

Note that even if we assumed that the adversary is less powerful or that the nodes are tamper resistant, the adversary can still perform *input based attacks*, meaning that it can directly manipulate the physical environment monitored by some of the sensors, and in this way, it can distort their measurements and the output of the aggregation mechanism at the base station.

In this paper, we will focus on this problem. More precisely, we assume that a small fraction of the sensor nodes are compromised (or their input is directly manipulated) by an adversary. In general, we do not make assumptions about how the compromised nodes behave (i.e., Byzantine fault model is assumed). We do assume, however, that the objective of the attacker is to maximally distort the aggregated value computed by the base station, while at the same time, the adversary does not want to be discovered (stealthy attack).

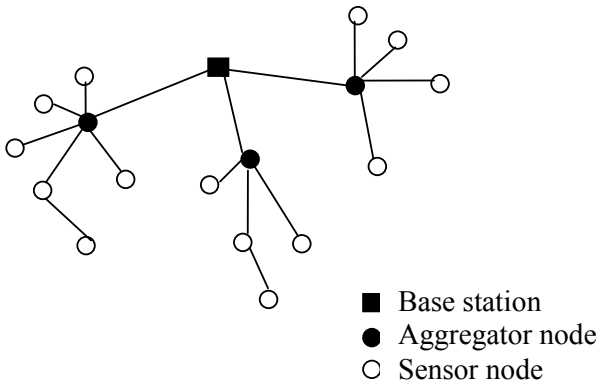


Fig. x.1. Sensor network architecture

The adversary may not limit its attack to simple sensor nodes, but it may try to compromise aggregator nodes too. Aggregator nodes are, obviously, more valuable targets for attack. The reason is that via gaining control over such nodes, the adversary might manipulate the aggregated data collected from a larger set of common sensors. Therefore, we cannot assume that aggregation points are trustworthy. We will assume, however, that the base stations cannot be compromised, and they always behave correctly, since there are only a few of them, they can be placed in a protected environment, and they can be monitored continuously.

The use of wireless communication channels means additional vulnerabilities. For instance, data can be easily eavesdropped or disrupted by jamming. Jamming may lead to temporary link failures, which may be viewed as a kind of Denial of Service (DoS) attack. We note, however, that in general, the adversary can be assumed to be more or less aware of the data measured in the network. Consequently, its primary goal may not be eavesdropping, but instead, it wants to modify the data delivered to the base station. Similarly, we can assume that it is more valuable for an adversary to manipulate the data collected from a given part of the network, than completely eliminate the corresponding subset of data via applying DoS attacks. On the other hand, if degradation of the communication graph endangers the fulfillment of the mission of the network, then techniques ensuring reliable communications (e.g., redundant paths, diversification techniques) must also be applied.

While cryptographique techniques (e.g., message integrity protection) and reliability methods (e.g., path diversification) are also important defense measures for resilient aggregation, in this paper, we put our emphasis on *statistical approaches* to detect deliberate attacks stemming from the insertion of bogus data by compromised sensors or by the manipulation of the physical environment of some uncompromised sensors. Note that such attacks simply cannot be detected by cryptographique techniques, that is why we need statistical methods, such as *extreme value detection* and *robust statistics*. These approaches help to detect and eliminate unusual, strange data values from the pool of data before the aggregation is performed, or they can considerably suppress the effect of bogus data on the aggregated value. The strength of these methods depends on the accuracy of the statistical model assumed for the data as statistical sample, and the accuracy of our assumptions about the capabilities of the adversary (e.g., the percentage of the compromised sensors and/or aggregators).

The rest of the paper is organized as follows: In Section 2, we give an overview of the state-of-the-art in resilient aggregation in sensor networks. In Section 3, we summarize the main results of outlier tests and robust statistics; these tools can be used to design data aggregation schemes that are resilient to attacks stemming from the insertion of bogus data by compromised sensors or by the manipulation of the physical environment of some uncompromised sensors. In Section 4, we give a detailed example of a statistical approach to resilient data aggregation. This example is based on an approach called RAndom SAmple Consensus (RANSAC). We also present some simulation results that show the efficiency of the RANSAC approach.

## 2 State-of-the-art

Resilient aggregation is a recent topic in the world of sensor networks; that is why the list of related papers is relatively short. In this section, we give an overview of these related papers. We begin, in Subsection 2.1, with some proposed data aggregation schemes that are resilient to node and/or link failures. As we mentioned before, such failures can be the result of jamming attacks, therefore, these schemes can be viewed as data aggregation algorithms that would work in the presence of a jamming adversary. Then, in Subsection 2.2, we give an overview of those papers that assume an adversary that can compromise aggregator nodes in the network and modify the data that they pass on toward the base station. As we will see, such attacks can be detected by appropriate cryptographique techniques. Finally, in Subsection 2.3, we deal with papers that address the problem of input based attacks and bogus data introduced by compromised sensors. The countermesaures for this latter class of attack are based on statistical approaches.

*2.1 Data aggregation in case of node or link failures*

Shrivastava *et al.* present a new approach for computing complicated statistics like the histogram of the sensor readings (Shrivastava et al. 2004). The topology of the sensor networks here is assumed to be a typical tree structure, where each intermediary node performs aggregation (i.e., compression). No attacker is assumed and even no link failures are considered.

Each sensor maintains a q-digest which comprises the summary of data measured by it. A q-digest is a binary tree built on the value space in the following way. The leaf nodes represent buckets corresponding to optionally overlapping intervals of the value space and store the count of readings that came from the corresponding interval. At the beginning, the intermediary nodes do not store any counts, they are needed for later processing. Note, that here the term 'node' is not corresponding to a sensor, but reflects to an egde in the graph of the q-digest.

To construct a memory-effective q-digest we need to hierarchically merge and reduce the number of buckets. This means going through all nodes bottom-up and checking if any node violates the digest property, which declares that none of the nodes should have a high count (this is needed to prove error bounds on the q-digest) and no parents-children triple should have low count. (The upper and the lower bounds for this count are determined by a compression parameter *k*.) This latter property is responsible for the compression, because if two sibling buckets have low counts then we do not want to have two separate counters for them. They will be merged into its parent and the empty buckets will be deleted. Doing this recursively bottom-up in the q-digest, we archieve a degree of compression. This emphasizes a key feature of q-digest: detailed information concerning data values which occur frequently are preserved in the digest, while less frequently occuring values are lumped into larger buckets resulting in compression, but also in information loss. This technique will always produce only an approximate result, but it is aware of the limited memory of the sensors.

Q-digests can be constructed in a distributed fashion too, that makes them available for calculating aggregates in sensor networks. If two sensors send their q-digests to their parent sensor (i.e., parent in the rooting tree), it can merge the received q-digests together and add its own q-digest to produce a common q-digest. The idea in merging of two q-digests is to take the union of the two q-digests and then run the above described compression algorithm on the union q-digest.

Q-digests are best deployable at such queries, where the output of the query can be represented with a single node - the error derived by the compression becomes minimal in this case. Such queries are the quantiles or specially the median. By a given memory *m*, the error of a quantile query can be upper bounded by $3\log(\sigma)/m$, where $\sigma$ stands for the size of the value space which is assumed to be $[1\ldots\sigma]$. However, other queries can be executed with the help of q-digests too (like inverse quantile, range query, consensus query or histogram), but their included error will possibly grow above this limit.

Chen *et al.* propose a method to calculate a statistic in a distributed fashion (Chen et al. 2005). The sensor network is abstracted as a connected undirected graph, where the sensor nodes are the

vertices and the bi-directional wireless communication links are the edges. This underlying graph can be of any kind depending on the placement of the sensor nodes. The paper does not consider attackers, but handles link failures.

The method proposed in (Chen et al. 2005) is called Distributed Random Grouping (DRG), and it can be used to calculate the average or the min/max of the sensor readings in a distributed fashion. The main idea of DRG is to randomly generate groups of nodes which locally synchronize their aggregate values to the common aggregate value of the group. Sooner or later this leads to a global consensus about the aggregate value in the sensor network. The detailed algorithm can be described as consecutive steps in a round: In the first step, each node in idle mode independently originates to form a group and become the group leader with probability $p$. A node $i$ which decides to be the group leader enters the leader mode and broadcasts a group call message containing its id (i.e., $i$). After this, it waits for the response of its neighbours. In the second step, a neighbouring node $j$, at the idle mode that successfully received the group call message, responds to the group leader $i$ with a joining acknowledgement containing its reading $v_j$. Node $j$ then enters to member mode and waits for the group assignment message from group leader $i$. In the third step of the algorithm the group leader $i$ gathers the received joining acknowledgements and computes the number of the group members and the desired statistic on the readings (e.g., the average). Then it broadcasts the group assignment message comprising the calculated statistic and returns to idle mode. The neighbouring nodes at member mode of group $i$ which receive the assignment message, update their value to the received one and return to idle mode.

Although this algorithm is natural, it is not straightforward to give bounds on the running time (i.e., the number of rounds needed until the whole network owns roughly the same value for the desired statistic). The authors assumed that the nodes run DRG in synchronous rounds to evaluate the performance of the algorithm. Based on theoretical calculations, they succeed to upper bound the number of rounds needed as a function of the properties of the graph, the grouping probability, the desired accuracy and the grand variance of the initial value distribution. An other way of evaluation was the simulation of the algorithm on hyphotetical sensor networks deployed according to Poisson random geometric graphs. The authors also compared DRG with known distributed localized algorithms, like the Flooding algorithm and the Uniform Gossip algorithm. In each comparison scenario, the DRG outperformed these two algorithms in the total number of transmissions, which is closely related to the energy consumption of the sensor network.

### *2.2 Attacking the aggregators, cryptographique countermeasures*

In (Anand et al. 2005), Anand *et al.* consider an eavesdropping attacker who wants to eavesdrop the network-wide aggregatum. The topology is the same as in (Shrivastava et al. 2004), each sensor senses and aggregates (except the leaf nodes, they have no aggregation task).

The adversary has the capability only for eavesdropping the communication between some sensor nodes. The goal of the adversary is to determine the aggregatum as precisely as possible. Two types of eavesdropping attacks are distinguished in the paper. There are passive eavesdroppers (who only listens to the broadcast medium) and active eavesdroppers (who have the power to send queries to the sensor network).

The authors of (Anand et al. 2005) show a way how the probability of a meaningful eavesdrop can be calculated, where meaningful means that the information obtained by the eavesdropper helps him to calculate a good estimate of the real aggregatum. The function that measures this probability is called eavesdropping vulnerability and it depends on the set of eavesdropped nodes, on the adversary's error tolerance and on the aggregation function used to calculate the aggregatum. Eavesdropping vulnerability can be calculated for a group containing only one aggregation point or for hierarchical groups, here the goal is to consider how close the adversary gets to the aggregatum higher in the tree when he eavesdrops on data in the lower level comprising that group (this latter kind of analysis is called eavesdropping vulnerability over a hierarchy).

In (Hu et al. 2003), Hu and Evans consider large sensor networks where the sensor nodes organize themselves into a tree for the purpose of routing data packets to a single base station represented by the root of the tree. Intermediate sensor nodes (i.e., nodes that are not leaves) perform aggregation on the data received from their children before forwarding the data to their parent. The details of the aggregation are not given, presumably it can be the computation of the minimum, the maximum, and the sum (for computing the average at the base station).

Two types of adversaries are considered: adversaries that deploy intruder nodes into the network, and adversaries that can compromise a *single* node in the network. Intruder nodes try to defeat the system by introducing bogus data into the network or altering the contents of the packets sent by the legitimate nodes. However, they do not have access to any key material. On the other hand, adversaries that can compromise a node are assumed to gain access to all the secrets of the compromised node. These two types of adversaries are considered separately, meaning that the key material of the compromised node is not distributed to the intruder nodes.

The goal of the adversary is to distort the final aggregated value by either modifying the contents of the data packets using the intruder nodes or forwarding a bogus aggregated value to the parent of a single compromised node.

The authors make the following system assumptions. The base station is powerful enough to send packets to all sensor nodes directly (i.e., in a single hop). On the other hand, sensor nodes can only communicate with other nodes in their vicinity. Message delivery between two neighboring nodes is reliable. The size of the network is large enough so that most of the nodes are several hops away from the base station (and the other nodes). However, the network is dense enough so that every node has several neighbors. A shared secret is established between the base station and each node in a safe environment before the deployment of the network.

The proposed solution is based on two mechanisms: delayed aggregation and delayed authentication. Delayed aggregation means that instead of performing the aggregation at the parent node, messages are forwarded unchanged to the grandparent and aggregation is performed there. This increases the overall transmission cost but it allows the detection of a compromised parent node if the grandparent is not compromised (and by assumption it cannot be compromised if the parent is compromised). Delayed authentication does, in fact, refer to the $\mu$TESLA protocol which allows broadcast authentication with using purely symmetric cryptography.

During the operation phase, leaf nodes send sensor readings to their parents. The integrity of these messages is protected by a MAC. Parent nodes compute the aggregated value of the data received from their children, but do not forward this aggregated value. Instead a parent node forwards the messages (together with their MACs) received from its children to its own parent (i.e., the grandparent of its children). In addition, the parent also sends a MAC which is computed over the aggregated value. Thus, intermediate nodes receive messages that contain values authenticated by their grandchildren, and a MAC on the aggregated value computed by their child. An intermediate node verifies the MACs of its grandchildren, computes the aggregated value (which should be the same as the one computed by its child) and then verifies the MAC of its child. If any of these verification fails, then the node raises an alarm. The idea is that since there is only a single compromised node in the network, if the child is compromised then the grandchildren are honest. Thus, if the child sends a MAC on a bogus aggregated value, then the node will detect this because it also receives the correct input values from the honest grandchildren.

MACs are computed with TESLA keys that are shared by the sender of the MAC and the base station but not yet known to the other nodes at the time of sending. These keys are disclosed after some time delay (specified by the TESLA protocol) by the base station. Thus messages and their MACs are stored and verified later, when the corresponding keys are revealed. The base station authenticates the disclosed keys by its own current TESLA key, which is also disclosed in a delayed manner.

The authors use informal arguments to illustrate that the scheme is resistant to the type of adversaries considered. In addition some cost analysis is performed too, which focuses on the communication costs of the scheme as sending and receiving messages are the most energy consuming operations. Finally some scalability issues are discussed.

In (Przydatek et al. 2003), Przydatek *et al.* present cryptography based countermeasures against the attacker, who wants to distort the aggregatum. The topology assumed is the following. A distant home server collects statistics of the measured data, which are the median, the average and the minimum/maximum of the sample. Intermediary nodes, aggregators with enhanced computation and communication power are used to optimize the communication need by calculating partial statistics on spot over subset of sensors (leaf-nodes in this tree structured topology). The communication complexity between the home server and the aggregator is a central issue for the optimization of the approach, because it is assumed that in real application this is an expensive long-distance communication link. The case of a single aggregator is considered in (Przydatek et al. 2003). Each sensor shares a separate secret key with the home server and the aggregator. The data sent from the nodes to the aggregator is authenticated and encrypted (using both keys).

The adversary can corrupt a small fraction of the sensors, and by using the compromised key(s) it is able to change the measured values arbitrarily. No model is assumed about the statistical properties (like probability distribution) of the measured data. Consequently the approach does not use outlier tests to detect and filter out corrupted data. While the median is the most common and simplest robust statistics, the other two statistics calculated (the average and the min/max) are not robust at all, so the protection in this paper is basically not statistical, and the countermeasures deployed at the level of the aggregator node(s) are based on cryptographic

approaches. The main attack is launched against the aggregator (stealthy attack), by corrupting it and modifying the aggregated value.

The secure aggregation approach proposed in this paper is based on cryptographic commitment and interactive proof techniques. The aggregator sends the aggregate statistics to the home server together with a Merkle hash tree based commitment. In the proof step the home server asks the aggregator for a randomly selected sub-sample. In this step the aggregator sends the wanted sub-sample in the form protected by the keys shared between the nodes and the home server. The home server checks elements of this sub-sample against the commitment by interacting with the aggregator. If this check is successful, i.e., the home server is convinced that the sub-sample is really comes from the sample used for the calculation of the commitment and sent by the corresponding sensors, it calculates the actual statistics for this sub-sample and compares the result to the value sent previously by the aggregator and calculated for the whole sample. If the distance between these two values are slight enough, the home server accepts the statistics authentic.

The quality of the approach is measured by a pair $(\varepsilon, \delta)$, where $\delta$ is an upper bound on the probability of not detecting a cheating aggregator, where the cheating means that the reported value sent by the aggregator is not within $\varepsilon$ bounds.

*2.3 Input based attacks, statistical countermeasures*

One of the most important papers in the field of resilient aggregation in sensor networks is from Wagner (Wagner 2004). Wagner assumes an abstract sensor network topology, where the inessential underlying physical structures are abstracted away. All the sensor nodes are connected with a single base station each over its own separate link. The base station is assumed to be trusted and the links are secure and independent, meaning that capturing one sensor node might compromise the contents of that node's channel but it has no influence on the other node's channel. The sensors send their packets directly to the base station via their channel, every packet contains a measurement value. The base station collects the measurements from all the sensor nodes and then it computes the desired aggregatum with the aggregation function *f*. The overall question is: Which aggregation functions can be securely and meaningfully computed in the presence of a few compromised nodes?

The main threat considered is that of malicious data. The adversary is able to inject malicious data to the list of sensor measurements by capturing a few nodes and modify their readings, e.g., by altering the environment around the sensor or by reverse-engineering the sensor. The modification of the sensor's environment can be considered for example as lighting a lighter near to a temperature sensor or as flashing with a flashlight to a photometer sensor. The adversary is only able to compromise a small percent of the sensor nodes (at least fewer than the half of the network), but he can inject arbitrary values in place of the original readings. Thus, the adversary is modelled by the Byzantine fault model. The goal of the adversary is to skew the computed aggregate as much as possible.

The term *resilient aggregation* has been coined in this paper and it refers to those aggregation function that are reliable even in the case if some of the sensor nodes are compromised. The

resiliency of an aggregation function *f* is measured by the root-mean-square (r.m.s.) error value of the function for the best possible *k*-node attack, where the best possible *k*-node attack is that one which skewes the aggregatum at most. The mathematical framework beyond the root-mean-square error is based on independent and identically distributed sample elements and on the knowledge of the parametrized distribution. An aggregation function is declared to be resilient if its r.m.s. error for the best *k*-node attack is beyond the critical value corresponding to $\alpha$ times the root-mean-square error in the uncompromised state, concisely expressed if it is $(k,\alpha)$-resilient for some $\alpha$ that is not too large. The analysis of the paper shows that the commonly used aggregation functions are inherently insecure. The min/max, the sum and the average are totally insecure, because only one compromised reading can mislead the base station (or generally the aggregator) without an upper bound – this means that $\alpha$ becomes infinity. The aggregation functions that are secure are the median and the count – these have acceptable $\alpha$ values. The tools proposed in (Wagner 2004) to archieve better resiliency is truncation (i.e., if we know that valid sensor readings will usually be in the interval [*l,u*], then we can truncate every input to be within this range) and trimming (i.e., throwing away the highest and the lowest part of the input).

The paper is based on strong statistical background and contains a lot of simple proofs which help to understand the theory of the root-mean-square error calculation. There were no simulations made to demonstrate the viability of the approach and no real-life experience are shown. However, this type of investigation does not need any simulation because it is purely mathematical.

The spectrum of the potencial application of resilient aggregation is sensor networks is very wide. Resilient aggregation is best-suited to settings where the data is highly redundant, so that one can cross-check sensor readings for consistency. None the less there are many possible applications of such robust aggregation functions, the paper does not mention any of them. But there are scenarios where aggregation should not be applied, for instance in fire-detection or in any systems searching for a needle in the haystack.

Wagner's idea has been developed further by Buttyán *et al.* in (Buttyán et al. 2006). In (Buttyán et al. 2006), the abstract topology is the same as in the previous paper. The sensor nodes perform some measurements and send their readings to the base station. The base station runs the aggregation procedure (after an analysis phase) and it is assumed to be reliable. The communication between the nodes and the base station is assumed to be secure – there are effective cryptographic techniques to protect it and resilient aggregation is not concerned with this problem. Nonetheless, some percentage of the sensor readings can be compromised by artificially altering the measured phenomena.

The adversary is allowed to modify the sensor readings before they are submitted to the aggregation function. The affected sensors are chosen randomly without knowing their readings (myopic attacker). The adversary's goal is to maximize the distortion of the aggregation function, defined as the expected value of the difference between the real parameter and the computed one. In addition, the adversary does not want to be detected, or more precisely, he wants to keep the probability of successful detection of an attack under a given small threshold value. This criterion upper bounds the possibilities of the adversary, even for aggregation functions that were considered to be insecure earlier.

The novel data aggregation model in (Buttyán et al. 2006) consists of an aggregator function and of a detection algorithm. The detection algorithm analyzes the input data before the aggregation function is called and tries to detect unexpected deviations in the received sensor readings. In fact, trimming (proposed by Wagner in (Wagner 2004)) is a special case of this more general idea. The detection algorithm uses the technique of sample halving, i.e., it first halves the sample and computes the sum of the halves separately. Then it subtracts the two sums from each other and indicates attack if the result is above a limit. The concrete value of this limit can be calculated from the desired false positive probability (i.e., when there is no attack but the detection algorithm indicates attack). As it can be seen, (Buttyán et al. 2006) uses a natural statistical tool to analyze the skewness of the distribution of the sensor readings and to filter out unusual samples. However, to filter out something that is unusual, we need some a priori knowledge about what is usual. Here the assumption is that the sensor readings are independent and identically distributed and that the variance of the sample is 1.

The efficiency of this novel data model and the sample halving technique is measured by the probability of successful detecting an attack. This probability is shown as a function of the distortion archieved by the adversary for different number of compromised sensors. Thanks to the detection algorithm, the probability of an attack detection grows steeply even for small ammount of compromised sensor readings and thus the archievable distortion is strictly upper bounded.

## 3 Outliers and Robust Statistics

By modifying the readings of the compromised sensors, or by manipultaing the sensors' physical environment (in order to influence their measurements), the adversary contaminates the samples received by the aggregation function with bogus data. Methods for eliminating the effects of bogus data have been extensively studied in statistics (although not with a determined adversary in mind), and we find it useful to summarize the main achievements in this field here. Thus, after a brief introduction, we give an overview of *outlier tests* and *robust statistics* in this section.

An outlier is an observation, which is not consistent with the data, meaning that we have some assumptions about the statistical characteristics of the data (e.g., the type of distribution) and some data points do not fit this assumption.

If an outlier is really an error in the measurement, it will distort the interpretation of the data, having undue influence on many summary statistics. Many statistical techniques are sensitive to the presence of outliers. For example, simple calculations of the mean and standard deviation may be distorted by a single grossly false data point. The most frequently cited common example is the estimation of linear regression from sample contaminated with outliers: Because of the way in which the regression line is determined (especially the fact that it is based on minimizing not the sum of simple distances but the sum of squares of distances of data points from the line), outliers have a dangerous influence on the slope of the regression line and consequently on the value of the correlation coefficient. A single outlier is capable of considerably changing the slope of the regression line and, consequently, the value of the correlation. Therefore it is tempting to remove atypical values automatically from a data set. However, we have to be very careful: if an "outlier" is a genuine result (a genuine extreme value), such a data point is important because it might indicate an extreme behavior of the process under study. No matter how extreme a value is

in a set of data, the suspect value could nonetheless be a correct piece of information. Only with experience or the identification of a particular cause can data be declared 'wrong' and removed.

Accordingly outlier tests and elimination of detected outliers are appropriate if we are confident about the distribution of the data set. If we are not sure in that, then robust statistics and/or non-parametric (distribution independent) tests can be applied to the data.

If we know extreme values represent a certain segment of the population, then we must decide between biasing the results (by removing them) or using a nonparametric test that can deal with them. In statistics, classical least squares regression relies on model assumptions (the Gauss-Markov hypothesis) which are often not met in practice. 'Nonparametric' tests make few or no assumptions about the distributions, and do not rely on distribution parameters. Their chief advantage is improved reliability when the distribution is unknown. However, non-parametric models give very imprecise results, compared to their parametric counterparts. Therefore, a compromise between parametric and non-parametric methods was created: robust statistics.

The aim of robust statistics is to create statistical methods which are resistant to departure from model assumptions, i.e., outliers. If there is no reason to believe that the outlying point is an error, it should not be deleted without careful consideration. However, the use of more robust techniques may be warranted. Robust techniques will often downweight the effect of outlying points without deleting them. Robust statistics include methods that are largely unaffected by the presence of extreme values. Therefore the three approaches handling data seemingly "contaminated" with "atypical" points are the following:

- Outlier tests,
- Robust estimates,
- Non-parametric methods.

Below the main approaches in outlier testing and robust statistics are summarized.

*3.1 Outlier tests*

There are outlier tests which perform quite good across a wide array of distributions, like the Box Plot which is a traditional graphical method, however can also be automated. We mark the largest data point that is less than or equal to the value that is 1.5 times the interquartile range (*IQR*) above the 3rd quartile. Similarly we mark the smallest data point that is less than or equal to the value that is 1.5 times the interquartile range (*IQR*) below the 1st quartile. Data point above or below this upper and lower marks (so called whiskers) is considered an outlier. Why 1.5 *IQR*s? John Tukey, the inventor of Box Plot, commented: "1 is too small and 2 is too large." Really, in practice this outlier rule is quite good across a wide array of distributions.

For instance, consider the following ranked data set of daily temperatures in winter season:

$$-15, -7, -4, -3, -2, +1, +2, +4, +5$$

Here is an example, using the first set of numbers above using Tukey's method of determining *Q1* and *Q3*. From $Q1 = -4$, $Q3 = +2$, we get *IQR* = 6. Now 1.5 times 6 equals 9. Subtract 9 from the first quartile: $-4 - 9 = -13$. Note that $-15$ is an outlier, and the whisker should be drawn to $-7$,

which is the smallest value that is not an outlier. Add 9 to the third quartile: $+2+9=11$. Any value larger than 11 is an outlier, so in this side we have no outlier. Draw the whisker to the largest value in the dataset that is not an outlier, in this case $+5$. Since this value is the 3rd quartile, we draw no whisker at all. Mark $-15$ as outlier.

Tests are more sharp if we know something about the underlying distribution of data. Assuming the knowledge of the mean and the standard deviation some researchers use simple quantitative technique to exclude outliers. They simply exclude observations that are outside the range of ±2 standard deviations (or even ±1.5 sd's) around the mean. Refining this method we can draw confidence intervals also for the mean, which give us a range of values around the mean where we expect the "true" (population) mean is located (with a given level of certainty). For example, if the (sample) mean is 12, and the lower and upper limits of the $p = 0.05$ confidence interval are 8 and 16 respectively, then we can conclude that there is a 95 percent probability that the population mean is greater than 8 and lower than 16. The width of the confidence interval depends on the sample size and on the variation of data values. The larger the sample size, the more reliable its mean. The larger the variation, the less reliable the mean. The standard calculation of confidence intervals is based on the assumption that the variable is normally distributed in the population. The estimate may not be valid if normality assumption is not met, unless the sample size is large, say $n = 100$ or more, which is not unusual in case of sensor networks.

There are outlier tests which are explicitly based on the assumption of normality. A typical such test is the Grubbs' test, which is also known as the maximum normed residual test. Grubbs' test detects one outlier at a time. This outlier is expunged from the set of data and the test is iterated until no outliers are detected. More formally, Grubbs' test is defined for the hypothesis:

$H_0$: There are no outliers in the data set.

$H_1$: There is at least one outlier in the data set.

The Grubbs' test statistic is defined as:

$$G_1 = \max_i \left|\bar{x} - x\right| / s$$

with $x$ and $s$ denoting the sample mean and standard deviation, respectively. In words, the Grubbs' test statistic is the largest absolute deviation from the sample mean in units of the sample standard deviation. This is the two-sided version of the test. The Grubbs' test can also be defined as one of the following one-sided tests: we can test if the minimum value is an outlier

$$G_2 = \left|\bar{x} - x_{\min}\right| / s$$

with $x_{min}$ denoting the minimum value. Similarly we can test if the maximum value is an outlier

$$G_3 = \left|\bar{x} - x_{\max}\right| / s$$

with $x_{max}$ denoting the maximum value. The critical values are calculated according to Student t-distribution with appropriate degree of freedom (Grubbs 1969).

There are approaches, like the Random Sample Consensus approach (RANSAC) (Fischler et al. 1981), which rely on random sampling selection to search for the best fit. The model parameters are computed for each randomly selected subset of points. Then the points within some error tolerance are called the consensus set of the model, and if the cardinality of this set exceeds a pre-specified threshold, the model is accepted and its parameters are recomputed based on the whole

consensus set. Otherwise, the random sampling and validation is repeated as in the above. Hence, RANSAC can be considered to seek the best model that maximizes the number of inliers. The problem with this approach is that it requires the prior specification of a tolerance threshold limit which is actually related to the inlier bound. In Section 4, we will present a possible application of the RANSAC approach.

The Minimum Probability of Randomness (MINPRAN) (Stewart 1995) is a similar approach, however it relies on the assumption that the noise comes from a well known distribution. As in RANSAC, this approach uses random sampling to search for the fit and the inliers to this fit that are least likely to come from the known noise distribution.

*3.2 Robust statistics*

As it was mentioned above, robust techniques will often downweight the effect of outlying points without deleting them. Several questions arise: How many outliers can a given algorithm tolerate? How can we describe the influence of outliers on the algorithm? What are the properties desirable for robust statistical procedures? Accordingly, three basic tools are used in robust statistics to describe robustness:

1. The breakdown point,
2. The influence function,
3. The sensitivity curve.

Intuitively, the breakdown point of an estimator is the maximum amount of outliers it can handle. The higher the breakdown point of an estimator, the more robust it is. The finite sample breakdown point $\varepsilon_n^*$ of an estimator $T_n$ at the sample $(x_1,\ldots,x_n)$ is given by

$$\varepsilon_n^*(T_n) = \frac{1}{n}\max_{m\geq 0}\left\{ m : \max_{i_1,\ldots,i_m} \sup_{y_1,\ldots,y_m} \left|T_n(z_1,\ldots,z_n)\right| < +\infty \right\}$$

where $(z_1,\ldots,z_n)$ is obtained by replacing the $m$ data points $(x_{i_1},\ldots,x_{i_m})$ by arbitrary values $(y_1,\ldots,y_n)$. The maximum breakdown point is 0.5 and there are estimators which achieve such a breakdown point. For example, the most commonly used robust statistics, the median, has a breakdown point of 0.5.

As for the influence function we distinguish the empirical influence function and the (theoretical) influence function (Hampel et al. 1986). The empirical influence function gives us an idea of how an estimator behaves when we change one point in the sample and no model assumptions is made. The definition of the empirical influence function $EIF_i$ at observation $i$ is defined by replacing the $i$-th value $x_i$ in the sample by an arbitrary value $x$ and looking at the output of the estimator (i.e., considering it as a function in variable $x$): $T_n(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots,x_n)$.

The influence function tells us what happens to an estimator when we change the distribution of the data slightly. It describes the effect of an infinitesimal "contamination" at the point $x$ on the estimate. Let $\Delta_x$ be the probability measure which gives probability 1 to $x$. The influence function is then defined by:

$$IF(x,T,F)=\lim_{t\to 0}\frac{T(t\Delta_x+(1-t)F)-T(F)}{t}$$

A good influence function is qualified by the following properties: finite rejection point, small gross-error sensitivity and small local-shift sensitivity.

The rejection point is defined as the point beyond which function *IF* becomes zero (Goodall 1983), formally

$$\rho^*=\inf_{r>0}\{r:IF(x,T,F)=0,|x|>r\}$$

Observations beyond the rejection point have zero influence. Hence they make no contribution to the final estimate. However, a finite rejection point may result in the underestimation of scale. This is because when the samples near the tails of a distribution are ignored, too little of the samples may remain for the estimation process (Goodall 1983).

The Gross Error Sensitivity expresses asymptotically the maximum effect a contaminated observation can have on the estimator. It is the maximum absolute value of the *IF* when *x* is varied.

The Local Shift Sensitivity (l.s.s.) measures the effect of the removal of a mass at *x* and its reintroduction at *y*. For a continuous and differentiable *IF*, l.s.s. is given by the maximum absolute value of the slope of *IF* at any point:

$$\lambda^*(T,F)=\sup_{x\neq y}\left|\frac{IF(y,T,F)-IF(x,T,F)}{y-x}\right|$$

General constructions for robust estimators are the M-estimators (Rey 1983). A motivation behind M-estimators can be a generalization of maximum likelihood estimators (MLE). MLE are therefore a special case of M-estimators (hence the name: "generalized Maximum likelihood estimators"). The M-estimate, $T(x_1,\ldots,x_n)$ for the function $\rho$ and the sample $x_1,\ldots,x_n$ is the value that minimizes the following objective function

$$\min_t\sum_{j=1}^{n}\rho(x_j;t)$$

i.e., the estimate *T* of the parameter is determined by solving

$$\sum_{j=1}^{n}\psi(x_j;t)=0,$$

where

$$\psi(x_j;t)=\frac{\partial\rho(x_j;t)}{\partial t}.$$

One of the main reasons for studying M-estimators in robust statistics is that their influence function is proportional to $\psi: IF(x,T,F)=b\psi(x;T(F))$. When the M-estimator is equivariant, i.e., $T(x_1+a,\ldots,x_n+a)=T(x_1,\ldots,x_n)+a$, for any real constant *a* then we can write functions $\psi$ and $\rho$ in terms of residuals $x-t$. If additionally scale estimate *S* is used we obtain the, so called, scaled residuals $r=(x-t)/S$, and we can write $\psi(r)=\psi((x-t)/S)$ and $\rho(r)=\rho((x-t)/S)$.

For the purpose of scale estimate most M-estimators use the MAD (Median Absolute Deviation). For example $\rho(r) = r^2/2$ (least-squares) estimators are not robust because their influence function is not bounded; if no scale estimate (i.e., formally, $S = 1$) is used we get back the mean as estimate. $\rho(r) = |r|$ estimators reduce the influence of large errors, but they still have an influence because the influence function has no cut off point; using this estimator we get the median. For more details about M-estimators and their robustness we refer the reader to (Huber 1981) and (Rey 1983).

The W-estimators (Goodall 1983) represent an alternative form of M-estimators. The L-estimators are also known as trimmed means for the case of location estimation (Koenker et al. 1978). All the above estimators are either obliged to perform an exhaustive search or assume a known value for the amount of noise present in the data set (contamination rate). When faced with more noise than assumed, these estimators will lack robustness. And when the amount of noise is less than the assumed level, they will lack efficiency, i.e., the parameter estimates suffer in terms of accuracy, since not all the good data points are taken into account.

Now that we have got an inside view of robust statistics, in Section 4, we will present an example how to use the above mentioned RANSAC paradigm to build a filtering and model fitting tool that can be applied in resilient aggregation in sensor networks.

## 4 An example of the RANSAC approach

Usually it is impossible to protect the sensor nodes from malicious mishandling, therefore, we need resilient aggregation techniques to treat the situation of receiving some amount of bogus data. The RANSAC paradigm is capable of handling data containing a significant percentage of gross errors by using random sampling. That makes it suitable for environments such as sensor networks where the sensors can be affected by compromising their measurements and that is why it can be convenient as a building block in robust statistical tools.

RANSAC is the abbreviation of Random Sample Consensus and it defines a principle how non-consistent data can be filtered from a sample, with other words, how a model can be fitted to experimental data (smoothing). The principle is the opposite to that of conventional smoothing techniques: Rather than using as much of the data as possible to obtain an initial solution and then attempting to eliminate the non-consistent data elements, RANSAC uses as few of the data as feasible to determine a possible model and then tries to enlarge the initial datum set with the consistent data. Algorithm 1 shows how the RANSAC principle can work as the heart of an algorithm.

**Algorithm 1** RANSAC Pseudo-Algorithm

```
1: while No. of trials ≤ Max trials do
2:    Randomly select s data elements (S)
3:    Instantiate the model M
4:    Depute all data elements within some error tol-
      erance of M (S*)
5:    if #(S*) > threshold then
6:       Instantiate the model M* based on S*
7:       Quit the loop
8:    end if
9: end while
10: if #(S*) < threshold then
11:    Compute M* on the largest S* or terminate in
       failure
12: end if
```

For example, if the task is to fit a circle to a set of points with two-dimensional coordinates, then the above algorithm would randomly choose three points for the initial set *S* (since three points are required to determine a circle), and would fit a circle to this three points (this circle would be model *M*). With this the algorithm would enlarge the initial set with all the points that are not too far from the arc of the circle (this would be *S**, called also the consensus set of *S*). If the size of the consensus set is above a threshold, then the algorithm would finish by fitting a final circle by considering all the points within the consensus set. If the consensus set is too small, then the algorithm would drop *S* and would retry to establish a suitable *S** by picking another three points and running the algorithm again. If after some number of trials the algorithm would not find a suitable consensus set, it would finish with the best possible fit (that would include more errors than desired) or would return with an error message.

### *4.1 The applied RANSAC algorithm*

To show how to use the RANSAC paradigm in resilient aggregation, we have implemented a filtering function based on this paradigm that filters out outlier measurements supposing that all the samples are independent and normally distributed (with the same parameters) in the uncompromised case. For sake of simplicity we have called it the RANSAC algorithm. Notice, that we assumed nothing about the expected value or the standard deviation of the distribution.

Our RANSAC algorithm is applied against an attacker who can distort some sensor measurements. The assumptions we have made on the attacker are the following. The attacker has limited power resources, but he has full control over some part of the sensor network, so he knows the concrete values measured by the compromised nodes and he can arbitrarily modify the values sent to the base station. The attacker knows the applied RANSAC algorithm in detail, but he cannot horn in it since it runs on the base station which is assumed to be secure. The attacker also knows that the sample is normally distributed. The sensors communicate with the base station by cryptographically protected messages so the attacker cannot modify the messages after they were encrypted. However, there is no need for it, because the attacker can arbitrary modify the messages of the compromised nodes just before the encryption. The attackers object is to

cause as high distortion in the aggregation result as possible while remaining undetected (stealthy attack).

We investigated a naive attacker who simply modifies all the compromised sensor measurement values to one common value. For example, if the attacker wants the cooling system to turn on, then he lights a lighter just beside one (or more) temperature sensors. That implies a peak in the histogram of the sample received by the base station. The attacker is able to set this peak to an arbitrary place in the histogram, with other words, he can modify the measured values to an arbitrary common one.

The working of the RANSAC algorithms is as follows (see Figure x.1). The base station receives the sample compromised previously by the attacker. The sample is the input of the RANSAC algorithm along with an upper estimation of the percentage of compromised nodes κ, and the required confidence level α. At first, a bag $S$ of minimum size will be randomly chosen to establish a preliminary model. Here we use bag instead of set because the sample may contain repetitive elements. The size of bag $S$ is $s$, the model $M$ is the theoretical histogram of the empirical Gaussian distribution with the expected value of

$$\hat{\theta} = \frac{1}{s}\sum_{i=1}^{s} S_i$$

and with the standard deviation of

$$\hat{\sigma}^2 = \frac{1}{s-1}\sum_{i=1}^{s}(S_i - \overline{S})^2 = \frac{1}{s-1}\sum_{i=1}^{s}(S_i - \hat{\theta})^2$$

with the restriction that $\hat{\sigma}^2$ cannot be 0.

After the histogram is imagined, the algorithm checks whether the sample is consistent with the histogram or not. The algorithm collects those elements from the sample that can be fit to the theoretical histogram and temporally drops the other elements. This phase is denoted by *Depute S** on Figure x.2 and can be described as

$$y_x = \min(c_x, n\phi(x \mid \hat{\theta}, \hat{\sigma})) = \min\left(c_x, \frac{n}{\sqrt{2\pi}\hat{\sigma}} e^{-\frac{(x-\hat{\theta})^2}{2\hat{\sigma}^2}}\right),$$

where $y_x$ is the number of elements with a value of $x$ that should not be dropped, $n$ is the size of the sample, $c_x$ is the number of elements in the sample with a value of $x$ and $\phi(x \mid \hat{\theta}, \hat{\sigma})$ is the probability density function of the Gaussian distribution. The bag

$$S^* = \bigcup_i y_i$$

is the consensus bag of $S$. If the size of $S^*$ is smaller than a required size $t$ then the algorithm starts again from the beginning, otherwise $S^*$ will be forwarded to the aggregator. There is an upper bound on the maximum number of retrials denoted by $f$. If there were more iterations than $f$, the algorithm ends with failure. The aggregator can be of any kind, here we used the average to estimate the expected value of the distribution of the sample. The value produced by the aggregator is $M^*$.

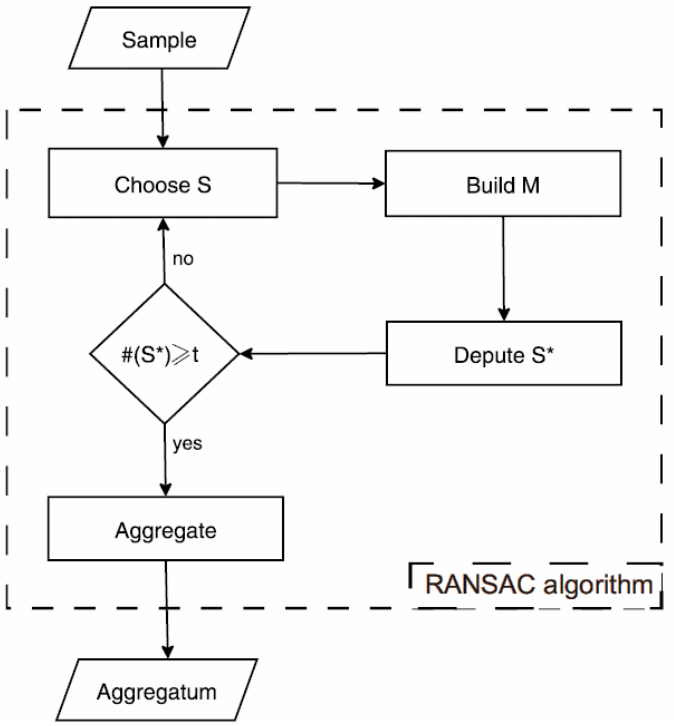

Fig. x.2. The RANSAC algorithm

The RANSAC algorithm consists of three parameters that have not been defined yet. These are the size $s$ of the initial bag $S$, the required size $t$ of the consensus bag $S^*$ and the maximum permitted number $f$ of iterations.

The size $s$ of the initial bag is desired to be as small as possible according to the RANSAC paradigm. For the RANSAC algorithm, we need to establish a theoretical histogram according to a Gaussian distribution. The Gaussian distribution has two parameters, the expected value and the standard deviation. The expected value can come from only one element, but for the standard deviation we need at least two elements. That was the motivation by the choice of $s = 2$.

The required size $t$ of the consensus bag is the most important parameter in the algorithm, however, the RANSAC paradigm does not gives us any hint about the correct choice of its value. If $t$ is small, then the algorithm has a higher probability to succeed, but the aggregatum at the end will contain a high level of error caused by the attacker. If $t$ is too big, the algorithm cannot work because of the high expectations on the number of elements in the final bag. Nevertheless, we required the algorithm to be as effective as possible by filtering all the attacked elements:

$$t = (1 - \kappa)n$$

where $\kappa$ is the upper estimation for the number of attacked sensor nodes and $n$ is the total number of sensor nodes in the network.

The maximum number $f$ of iterations can be figured on probabilistic analysis. Assuming that we need only one $S$ bag that satisfies the criterion that it does not contain compromised elements, we can try to select $s$ elements for $S$ as long as the probability of finding a correct $S$ is beyond a given value $1 - \alpha$, where $\alpha$ is the confidence level. Since the elements are selected without replacement, the probability of taking a good $S$ is

$$P_S = \frac{\binom{n-\kappa n}{s}}{\binom{n}{s}} = \prod_{i=0}^{\kappa n-1} \frac{n-s-i}{n-i}$$

The probability of finding at least one correct $S$ out of $f$ trials is

$$P_S^f = 1-(1-P_S)^f = 1-\left(1-\frac{\binom{n-\kappa n}{s}}{\binom{n}{s}}\right)^f$$

and we require

$$P_S^f \geq 1-\alpha$$

This leads to

$$f \geq \log_{1-P_S} \alpha$$

We have chosen

$$f = \log_{1-P_S} \alpha + 1,$$

because there is no need for a higher perturbation. The value of $f$ is related to the punctuality of the algorithm as well as the confidence level $\alpha$. Since the confidence level is an input of the algorithm, the value $f$ can be controlled through the proper choice of $\alpha$.

*4.2 Results of the simulations*

To validate the RANSAC algorithm, we have implemented it in a simulation environment. The parameters for the simulation are included in Table 1.

Table 1. Simulation Parameters

| **Parameter name** | **Value** |
|---|---|
| $n$ (sample size) | 100 |
| Sample distribution | $N(\mu,\sigma)$ |
| $\mu$ (expected value, unknown to the algorithm) | 0 |
| $\sigma$ (standard deviation, unknown to the algorithm) | 1 |
| $s$ (size of random set) | 2 |
| $\alpha$ (confidence level) | 0.01 |

The sample for the simulations has been generated using the randn function of Matlab. After the attacker has compromised $\kappa n$ nodes (i.e., he set the value of $\kappa n$ nodes to a fixed value), the RANSAC algorithm analyzes the modified sample by trying to establish a model that fits at least $t$ readings. If the algorithm can not make it in $f$ iterations, it stops in failure. In this case either $\alpha$ was too low or $\kappa$ was too high and so the algorithm could not estimate under the given criteria.

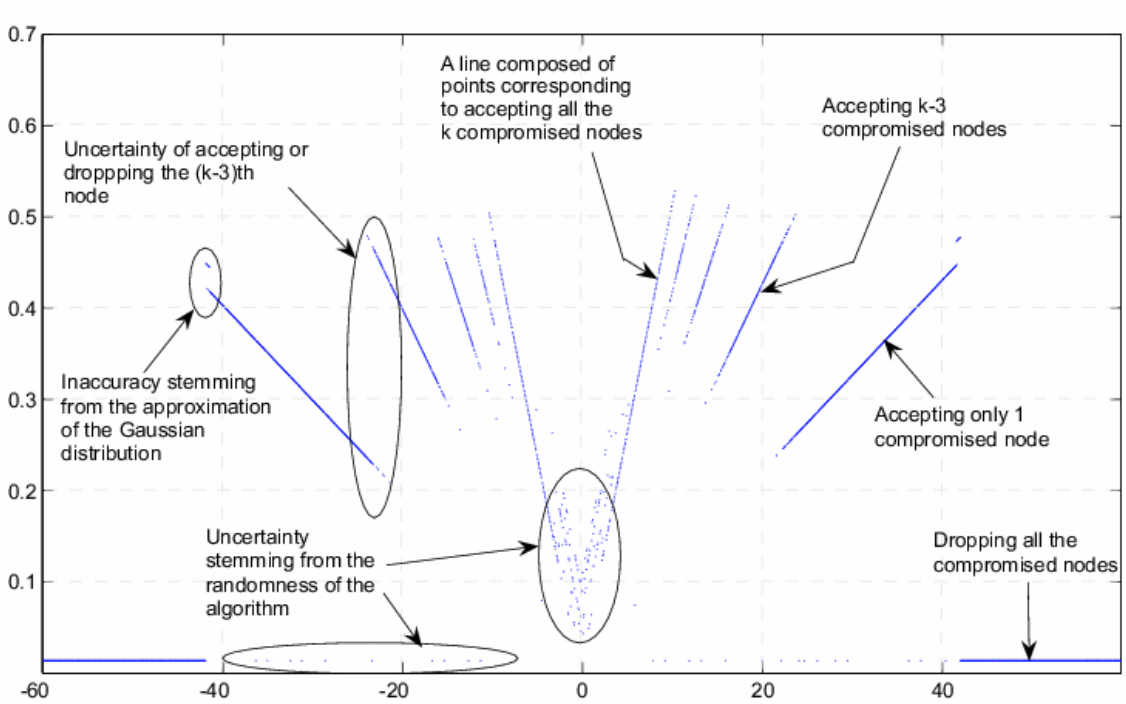

Fig. x.3. A typical simulation result and its explanation

Figure x.3 shows a typical results for the maximum distortion achieved by the attacker as a function of the place of the peak (i.e., the value to which it sets the readings of the compromised sensors), where the maximum is taken over 50 simulation runs, and $\kappa = 0.05$ (i.e., the proportion of compromised nodes is 0.05 which means that 5 nodes were compromised). The distortion is defined as the absolute value of the difference between the real average and the estimated average produced by the RANSAC algorithm. Notice, that the expected value of the test data is 0, therefore Figure x.3 is nearly symmetrical to $x = 0$. The slight asymmetry comes from the randomness of the input data.

As it can be seen on the figure, the adversary can never achieve a distorion more than 0.6. This means that the estimated average will be always in the range $(-0.6, 0.6)$ surrounding the real average given that 5 nodes out of 100 are compromised. In other words, the adversary can perturb the compromised readings with an arbitrary value, but this will not significantly change the estimate. Moreover, this $(-0.6, 0.6)$ bound is independent of the expected value of the sample, thus for a higher expected value, the relative distortion (i.e., distortion divived by the expected value) can be arbitrarily small.

Figure x.3 tells us even more about the algorithm. The steeply ascending lines composed of points correspond to accepting some elements from the peak after the filtering phase. For some $x$ values, it looks like there would be two distinct values for the same $x$ value. For example at $x = -22$, the two lines composed of points appear as if they both had a value for this value of $x$. As a matter of fact, there is only one $y$ value for every $x$, but in some cases the algorithm is uncertain to accept a compromised reading. In a small range surrounding $x = -22$, the algorithm accepts a compromised reading in some cases and drops it in another cases. This implies points that look like if they belonged to the same $x$. Some additional bursts on the figure stem from the inaccuracy of the approximation of the Gaussian distribution and from the random property of the

algorithm. As it can be seen, it is meaningless for the adversary to push the peak too far from the average because the highest distortion occurs when the peak is about 10 units far from the real average. Above the distance of approximately 42 units the peak will have no influence on the estimated average (i.e., it will be always filtered out by the algorithm). The unit can be of any kind, for example degree in case of a thermometer sensor or lumen in case of a photometer sensor.

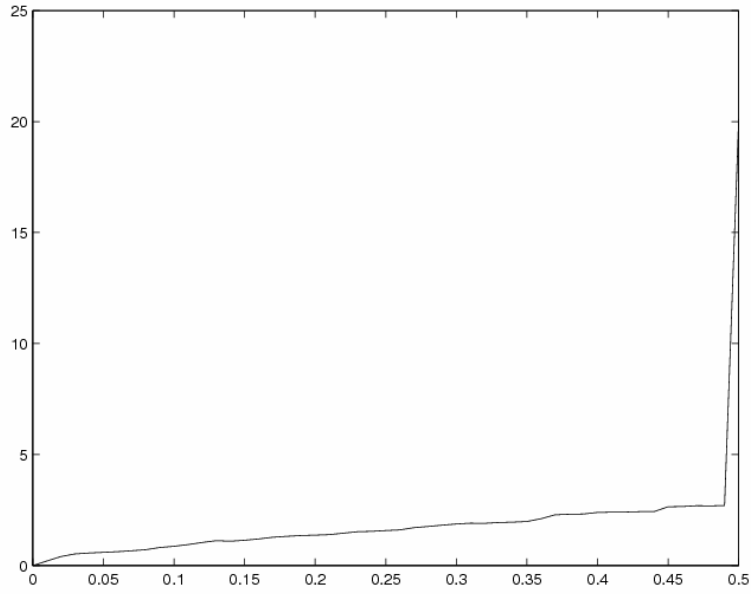

Fig. x.4. Demonstration of the breakdown point

Another demonstration on the performance of the RANSAC algorithm is Figure x.4. This shows the highest reacheable distortion (*y* axis) as a function of different values of $\kappa$ (*x* axis). We made 50 simulation runs with the same parameter set for each values of $\kappa$ and plotted out the one with the maximum distortion. As one can see, the RANSAC algorithm reaches the theoretical maximum of the breakdown point, since the distortion never goes to infinity for $\kappa \leq 0.5$ (which means that at most 50% of the nodes are compromised). Notice, that the theoretical maximum for the breakdown point is 0.5. Another important issue is that the RANSAC algorithm strictly upper bounds the reacheable distortion even for high values of $\kappa$. For example, for a node compromisation ratio of 45% ($\kappa = 0.45$) the error is upper bounded in 3 units. This means, that even if almost the half of the nodes send bogus information to the base station, the RANSAC algorithm is capable to produce a result with at most 3 units of perturbation.

## 5 Conclusions

In typical sensor network applications, the sensors are left unattended for a long period of time once they have been deployed. In addition, due to cost reasons, sensor nodes are usually not tamper resistant. Consequently, sensors can be easily captured and compromised by an adversary. Moreover, when a sensor is compromised, it can send authentique messages to other nodes and to the base station, but those messages may contain arbitrary data created by the adversray (e.g., bogus measurments). Even if we assumed that the adversary is less powerful or that the nodes are tamper resistant, the adversary can still perform *input based attacks*, meaning that it can directly

manipulate the physical environment monitored by some of the sensors, and in this way, it can distort their measurements and the output of the aggregation mechanism at the base station. The task of resisilient aggregation is to perform the aggregation despite the possibility of the above mentioned attacks, essentially by supressing their effect.

Resilient aggregation cannot be based on pure cryptographic countermeasures (e.g., message authentication) and/or reliability mechanisms (e.g., diversification techniques). Statistical methods, such as extreme value detection and robust statistics, are relevant and inevitable additional elements in protecting data aggregation mechanisms in sensor networks. However, their efficiency basically depends on the accuracy of the model describing the characteristics of the data as a statistical sample. This means, for instance, that we have some a priori information about the probability distribution of data, the normal range of time-variability of the main statistical characteristics, the statistical dependence between measurements of certain groups of sensors etc. Many statistical techniques are available, and the task of the designer of the resilient aggregation scheme is to find the appropriate and valid adversary model, distortion measure, and description of the sample, as well as to appropriately apply the toolkit provided by statisticians.

Besides giving an overview of the state-of-the-art in resilient aggregation in sensor networks, and a brief summary of the relevant techniques in the field of mathematical statistics, we also introduced a particular approach for resilient aggregation in somewhat more details. This approach is based on RANSAC (RAndom SAmple Consensus), which we adopted for our purposes. We presented some initial simulation results about the performance of our resilient data aggregation scheme, where we assumed a priori knowledge of the type of the distribution of the clean sample, and a particular type of adversary. These results show that the our RANSAC based approach is extremely efficient in this particular case, as it can cope with a very high fraction of compromised nodes. In our future work, we intend to further develop the RANSAC approach for resilient aggregation by also considering correlated samples and fewer assumptions about the distribution.

## 6 Acknowledgements

The work presented in this paper has been partially supported by the UbiSec&Sens EU Project (contract number 026820) and the Hungarian Scientific Research Fund (contract number T046664). The first author has been further supported by the Hungarian Ministry of Education (BÖ 2003/70). The second author has been further supported by the HSN Lab.

## References

Anand M, Ives Z, Lee I (2005) Quantifying Eavesdropping Vulnerability in Sensor Networks. In: Proceedings of the Second VLDB Workshop on Data Management for Sensor Networks (DMSN), pp 3-9

Buttyán L, Schaffer P, Vajda I (2006) Resilient Aggregation with Attack Detection in Sensor Networks. In: Proceedings of the Second IEEE Workshop on Sensor Networks and Systems for Pervasive Computing (PerSeNS), accepted for publication

Bychkovskiy V, Megerian S, Estrin D, Potkonjak M (2003) A collaborative approach to in-place sensor calibration. In: Proceedings of the Second International Workshop on Information Processing in Sensor Networks (IPSN), pp 301-316

Chen J-Y, Pandurangan G, Xu D (2005) Robust Computation of Aggregates in Wireless Sensor Networks: Distributed Randomized Algorithms and Analysis. In: Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks (IPSN), pp 348-355

Chum O, Matas J (2002) Randomized RANSAC with $T_{d,d}$ test. In: Image and Vision Computing, 22: 837-843

Fischler MA, Bolles RC (1981) Random sample consensus for model fitting with applications to image analysis and automated cartography. In: Communications of the ACM 24: 381-395

Goodall C (1983) M-estimators of location: An outline of the theory. In: Hoaglin DC, Mosteller F, Tukey JW (eds) Understanding Robust and Exploratory Data Analysis, Wiley, New York, pp 339-403

Grubbs F (1969) Procedures for Detecting Outlying Observations in Samples. In: Technometrics 11: 1-21

Hampel FR, Ronchetti EM, Rousseeuw PJ, Stahel WA (1986) Robust Statistics: The Approach Based on Influence Functions. Wiley, New York

Hu L, Evans D (2003) Secure Aggregation for Wireless Networks. In: Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT), pp 384-394

Huber PJ (1981) Robust Statistics. Wiley, New York

Koenker R, Basset Jr G (1978) Regression quantiles. In: Econometrica 36: 33-50

Lacey AJ, Pinitkarn N, Thacker NA (2000) An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration. In: Proceedings of the British Machine Vision Conference (BMVC)

Li Z, Trappe W, Zhang Y, Nath B (2005) Robust statistical methods for securing wireless localization in sensor networks. In: Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks (IPSN)

Przydatek B, Song D, Perrig A (2003) SIA: Secure Information Aggregation in Sensor Networks. In: Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys), ACM Press, New York, pp 255-265

Rey WJJ (1983) Introduction to Robust and Quasi-Robust Statistical Methods. Springer, Berlin, Heidelberg

Shrivastava N, Buragohain C, Agrawal D, Suri S (2004) Medians and Beyond: New Aggregation Techniques for Sensor Networks. In: Proceedings of theACM Conference on Embedded Networked Sensor Systems (SenSys), ACM Press, Baltimore, pp 239-249

Stewart CV (1995) MINPRAN: A new robust estimator for computer vision. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, 17: 925-938

Wagner D (2004) Resilient Aggregation in Sensor Networks. In: Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN), ACM Press, pp 78-87

Yan J, Pollefeys M (2005) Articulated Motion Segmentation Using RANSAC With Priors. In: Proceedings of the ICCV Workshop on Dynamical Vision

Zuliani M, Kenney CS, Manjunath BS (2005) The MultiRANSAC Algorithm and its Application to Detect Planar Homographies. In: Proceedings of the IEEE International Conference on Image Processing, pp 153-156