RANBAR: RANSAC-Based Resilient Aggregation in Sensor Networks

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

ABSTRACT

We present a novel outlier elimination technique designed for sensor networks. This technique is called RANBAR and it is based on the RANSAC (RANdom SAmple Consensus) paradigm, which is well-known in computer vision and in automated cartography. The RANSAC paradigm gives us a hint on how to instantiate a model if there are a lot of compromised data elements. However, the paradigm does not specify an algorithm and it uses a guess for the number of compromised elements, which is not known in general in real life environments. We developed the RANBAR algorithm following this paradigm and we eliminated the need for the guess. Our RANBAR algorithm is therefore capable to handle a high percent of outlier measurement data by leaning on only one preassumption, namely that the sample is i.i.d. in the unattacked case. We implemented the algorithm in a simulation environment and we used it to filter out outlier elements from a sample before an aggregation procedure. The aggregation function that we used was the average. We show that the algorithm guarantees a small distortion on the output of the aggregator even if almost half of the sample is compromised. Compared to other resilient aggregation algorithms, like the trimmed average and the median, our RANBAR algorithm results in smaller distortion, especially for high attack strengths.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: [Security and Protection]

General Terms

Algorithms, Design, Security

Keywords

Sensor networks, Resilient aggregation, Outlier elimination, Random Sample Consensus

1. INTRODUCTION

Sensor networks will be the near future's most powerful monitoring applications. These networks are autonomous networks consisting of a large number of sensor nodes and some base stations. The nodes are applied to perform measurements of some physical phenomena and report them to the base station. Then, it is the base station's task to aggregate the received sample elements. However, if there are some sensor nodes that were compromised by an attacker then a part of the received sample may be incorrect. If the base station aggregated the sample in this original form then an attacker could cause a significant distortion in the aggregation result.

There are two ways in which the sample can be compromised. Firstly, the messages that carry the data from the sensors to the place of the aggregation (the base station) can be modified in transit. This can be detected by cryptographic techniques. Secondly, the attacker may compromise some sensors in the network and affect their readings (e.g., it can increase the temperature around a temperature sensor). This latter kind of attack cannot be prevented, neither detected, by cryptographic mechanisms. Resilient aggregation is concerned with this problem. The objective of resilient aggregation is to perform secure data aggregation in the presence of an adversary who can modify the input of the aggregation function.

One of the first research paper on resilient data aggregation was written by David Wagner and it was published at SASN 2004 [1]. The author investigated the resilience of commonly known aggregation functions and found that the average, the minimum and the maximum calculations are not resilient even against only one compromised node. However, the median was found to be resilient, the count to be acceptable and the trimmed average to be employable in some cases, namely when an upper bound on the percentage of compromised nodes is known. But what shall we do if we want to calculate the average instead of the median and we do not known an upper bound on the percentage of compromised nodes?

In this paper we propose a novel technique for resilient sensor data aggregation to handle this problem using a common principle: the RANSAC paradigm [3]. The RANSAC paradigm aims at finding data elements in the sample that correspond to a specified data model. This paradigm is described in several papers that deal with camera calibration, hyphotesis testing, geological research, robotics and computer graphics, but to the best of our knowledge, nobody has applied it to achieve resilient aggregation. With this tech-

nique, we can notably reduce the power of an attack even if the attacker has compromised a large percent of the sample and knows the aggregation procedure in detail. Thus, the RANSAC paradigm can be especially useful in sensor networks, where the nodes are unattended and lack physical protection, and therefore they can be compromised in large quantities.

The rest of this paper is organized as follows. In Section 2, we present the related papers. In Section 3, we introduce the RANSAC paradigm in general. In Section 4, we specify the attacker model considered by the design of the RANBAR algorithm presented in Section 5. In Section 6, we study the simulation results. Finally, in Section 7, we conclude the paper.

2. RELATED WORK

If an attacker can cause an arbitrary altering on a part of the sample then the average as an aggregation function is not secure, because every modified sample element may have an unduly large influence on the aggregation result. What shall we do, if we want to calculate the average in a secure way? We shall use resilient aggregation techniques or in a wider sense robust statistics.

The term 'resilient aggregation' was coined by Wagner in his SASN 2004 paper [1] and it has reference to such aggregation methods that are also secure in case of an attack, i.e., it is hard for an attacker to produce a significant distortion in the aggregation result. The main threat considered by Wagner was that of malicious data. The adversary is able to insert malicious data in to the list of sensor measurements by capturing a few nodes and modifying their readings. 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 insert arbitrary values in place of the original readings. This means, that the adversary is modelled by the Byzantine fault model. The goal of the adversary is to skew the computed aggregate as much as possible. Wagner investigates this problem for the case when the aggregation function is, for example, the average, and finds that instead of the average we should use the median. But there is a problem with the median. In real life deployments, we usually do not want to know the median, but the average. In some cases, the median can be significantly far from the average, especially for high attack strengths.

In [2], the authors propose a resilient aggregation scheme with attack detection, where the aggregator analyzes the received sensor readings before the aggregation function is called. In this model the attacker wants to remain undetected besides causing as high distortion in the output of the aggregation function as possible. The novel data aggregation model in this paper consists of an aggregator function and 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 [1]) 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 threshold. The concrete value of this threshold can be calculated from the desired false positive probability (i.e., when there is no attack but the detection algorithm indicates attack). The

advantage of this approach is that in order to remain undetected, the adversary cannot distort the output arbitrarily, but rather the distortion is upper bounded, even for aggregation functions that were considered to be insecure, such as the average. Another drawback of the solution in [2] is that if the detection algorithm indicates attack, the aggregation function is not called. The approach of RANBAR improves on this, because here, an output is produced most of the time even when an attack is suspected.

The RANSAC (RANdom SAmple Consensus) paradigm was suggested by A. Fischler and R. C. Bolles in 1981 [3]. The RANSAC approach relies on random sampling selection to search for the best fit. The model parameters are computed for every 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 prespecified threshold, the model is accepted and its parameters are recomputed based on the whole consensus set. Otherwise, the random sampling and validation is repeated. 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 which is actually related to the inlier bound. The authors proposed this technique for fitting a model to experimental data, and applied it to the Location Determination Problem: given a set of control points in the image with known locations, determine the point in space from which the image was obtained. The proposed principle worked well, however, it is just a principle without defined parameters. We have adopted it for the purposes of resilient aggregation in sensor networks.

A large number of papers are concerned with securing some aggregation related functions with the help of cryptography. In [12] the adversary wants to estimate the network-wide aggregate as accurately as possible. To achieve this, the adversary can eavesdrop the communication between some of the sensors. The authors 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 aggregate. 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 aggregate.

Hu et al. [13] 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. The authors present countermeasures against intruder nodes deployed by the adversary and against a single node compromise. Their approach is based on 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. Delayed authentication refers to the method when intermediate nodes receive messages that contain values authenticated by their grandchildren and a MAC (Message Authentication Code) on the aggregated value computed by their child.

Przydatek et al. present cryptography based countermeasures against the attacker who wants to distort the aggregate in [14]. The secure aggregation approach proposed in that 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 subsample. In this step the aggregator sends the wanted subsample in the form protected by the keys shared between the nodes and the home server. The home server checks elements of this subsample against the commitment by interacting with the aggregator. If this check is successful, i.e., the home server is convinced that the subsample 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 subsample 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 small enough, the home server accepts the statistics as authentic.

Browsing through the literature, we can also find several papers on applications of robust techniques for securing measurement data without citing the problem of resilient aggregation. For example, in [4] and [5] statistical tools are deployed for securing the localization in sensor networks as well as allowing in-place sensor calibration. The applied statistical tools are the median and the method of cutting outlier measurements to secure these functions. The problem with the median is mentioned above, namely that it can be significantly far from the average, especially for a large number of compromised nodes. The solution for the in-place sensor calibration problem in [5] is built on time series correlation analysis and pairwise relations between sensor nodes. Experiments show that the results of the algorithm are promising but the algorithm is still in an early development phase.

There are several publications about applications of RANSAC. The main orientation of these papers is computer vision and image processing. In [6], the authors compare a RANSAC-based algorithm with traditional optimization schemes and find that in case of stereo camera calibration the use of RANSAC seems unjustified, while in motion estimation it is beneficial. In [7], RANSAC was improved by randomising its hyphotesis evaluation step and it was shown that under some conditions this approach is significantly more efficient than the standard RANSAC method. In [8], the authors use RANSAC for articulated motion segmentation, and test their algorithm with both synthetic and real data. They have found that their algorithm shows both the properties of efficiency and robustness. The authors of [9] extended RANSAC to handle the problem of simultaneous parameter estimation of multiple models in data sets with a high percentage of outliers. Experimental results on synthetic data seem to support the robustness of their approach.

3. THE RANSAC PARADIGM

The RANSAC paradigm is capable of handling data containing a significant percentage of gross errors by using random sampling. That makes it convenient to be a building block in robust statistical tools.

RANSAC is the abbreviation of RANdom SAmple Consensus and it defines a principle for filtering non-consistent data from a sample, or in other words, fitting a model to experimental data. The principle of RANSAC 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 data set with the consistent data.

Algorithm 1 shows how the RANSAC principle works in general. In the first step, RANSAC establishes an inital set S of minimum size s by randomly choosing sample elements. Then, it builds a model with the help of set S. This will be model M. In the next step, all elements that can be approximately modelled by model M (i.e., that are within some error tolerance from M) will be collected into set S^*, the consensus set of S. If the size of set S^* is satisfying, then the algorithm calculates the final estimation M^* (based on set S^*) for the missing parameter and it ends. If set S^* is too small, then the algorithm drops its intermediate results, and starts again by establishing a new random set S. In the case when the algorithm is unable to find a suitable consensus set S^* in some upper bounded number of trials, it can either end in failure or it can give an unprecise estimation based on the largest consensus set found during the operation.

Algorithm 1 RANSAC Pseudo-Algorithm

1: **while** *No. of trials* $\leq$ *Max trials* **do**
2:    Randomly select s data elements (S)
3:    Instantiate the model M based on S
4:    Select all data elements within some error tolerance from M (S^*)
5:    **if** $\#(S^*) > threshold$ **then**
6:       Instantiate the model M^* based on S^*
7:       Return
8:    **end if**
9: **end while**
10: Compute M^* on the largest S^* or terminate in failure

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 it would fit a circle to this three points (this circle would be model M). Then, 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 limit, then the algorithm would finish by fitting a final circle on 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 did 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.

The RANSAC paradigm is used in many applications, e.g., geography or computer vision (see Section 2). However, to the best of our knowledge nobody has applied it to resilient aggregation so far. 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 invalid data. RANSAC is able to handle data that contains a significant percentage of invalid elements, and that makes it suitable for environments

such as sensor networks where the sensors can be affected by compromising their measurements.

Up to this point, we have only presented a general paradigm for handling invalid data content in a sample. In Section 4 we introduce our attacker model and then we show how the paradigm can be applied against this attacker.

4. THE ATTACKER MODEL

Our RANBAR algorithm is applied against an attacker who can distort some sensor measurements. The attacker has limited power resources, but he has control over some part of the sensor network, thus he knows the concrete values measured by the compromised nodes and he can arbitrarily modify the values that will be sent to the base station by the compromised nodes. The attacker knows the RANBAR algorithm in detail, as well as the aggregation function, but he cannot control them since they run 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 have been encrypted. Thus, the attacker cannot modify the readings of the non-compromised sensors. The attacker's objective is to cause as high a distortion in the aggregation result as possible by altering the measurement of the compromised sensors.

However, it is not straightforward to determine what the best possible attack is. The attacker may attempt to achieve a high distortion in the aggregation result by introducing highly extreme values in the list of measurements, but then he risks that those values will be filtered out by the RANBAR algorithm. In other words, introducing highly exteme values may not be the best strategy. On the other hand, a small modification of the sensor readings, while probably being unfiltered, likely results in a small distortion. Due to the complexity and the probabilistic nature of the RANBAR algorithm, finding the best attack that achieves the highest possible distortion seems to be a very difficult problem. We postpone the investigation of this problem for the future.

As a first step, in this paper, we investigate the simplest attacker, whom we call the Peak Attacker. The Peak Attacker is a naïve attacker who simply modifies the measurement values of all the compromised sensor nodes to one common value. That implies a peak in the histogram of the sample received by the base station (hence the name). The Peak Attacker is able to set this peak to an arbitrary place in the histogram, in other words, he can modify the measured values to an arbitrary value. The intuition behind the choice of this attacker model is that it models well the behaviour of a real-world attacker who simply goes to the temperature sensors and lights a lighter near to them.

The Peak Attacker is characterized by the proportion of sensor nodes he is able to compromise (this proportion is denoted by κ). This means that their measured values are replaced with a common value that is best suitable for the attacker. κ is theoretically upper bounded by 0.5, since for a higher κ no defence is possible. The attacker wants to distort the aggregate as much as possible, therefore he chooses the common replacing value that maximizes the distortion. Since the details of the RANBAR algorithm are known, the attacker can compute this common value. This is the motivation behind assuming always the maximum attack strength in the simulations in Section 6 (worst case simulations).

5. THE RANBAR ALGORITHM

The RANBAR algorithm filters out outlier measurements from a sample. To do this, we need to know something about the sample to have the possibility to distinguish between outlier and outlier elements. For this, we assumed that the unattacked sample follows the empirical Gaussian distribution, but we assumed nothing about the expected value or the standard deviation of this distribution. Nevertheless, the RANBAR algorithm is not restricted to the i.i.d. case. We are currently working on a correlated data model, see Section 7.

The operation of the RANBAR algorithm is as follows (see Algorithm 1). The base station receives the sample compromised previously by the attacker. The sample is the input of the RANBAR algorithm. First, a set S of minimum size will be randomly chosen to establish a preliminary model. The size of set S is s, and the model M is the theoretical histogram of the empirical Gaussian distribution with the expected value of

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

and with the variance of

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

with the restriction that $\hat{\sigma}^2$ cannot be 0. S_i denotes the ith element of set S. We note that the RANBAR algorithm could be applied to any type of parametrized[1] distributions, but the initial estimation for the expected value and for the standard deviation would have another form. Generally speaking, our RANBAR algorithm uses the Maximum Likelihood Estimation (MLE) to obtain the initial parameters $\hat{\theta}$ and $\hat{\sigma}^2$.

After the probability density function (PDF) of the model M is calculated, the algorithm checks whether the histogram of the sample is consistent with the PDF of the model M or not. The bins of the histogram are determined by dividing the elements into 10 equiwidth buckets after cutting the upper and the lower 0.5% of the sample. The consistency check is done by repeated steps the following way.

1. First, the algorithm normalizes the histogram of the elements, i.e., it ensures that the area under the histogram is equal to one.

2. Then, it calculates the distance between the PDF of model M (denoted by $p(x)$) and the histogram of the sample (denoted by $h(x)$), where the distance is defined by

$$d = \int |p(x) - h(x)|_+, \qquad (3)$$

---

[1]In case of non-parametrized distributions the RANBAR algorithm cannot work without prior analysis. Considering such a situation, we first have to approximate the underlying non-parametrized distribution with a parametrized one, and this has to be done when there is no attack. After this initializing step, the RANBAR algorithm can be applied to that parametrized distribution.

where

$$|x|_+ = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} \qquad (4)$$

3. Finally, it drops one element from the bin of the histogram corresponding to the maximum $|p(x) - h(x)|_+$.

If the distance becomes smaller than δ, this repetitive phase ends.

The remaining sample elements constitute the set S^*, called also the consensus set of S. If the size of S^* is smaller than a required size t then the algorithm starts again from the first step, 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 estimator can be of any kind; here we use the average to estimate the expected value of the distribution of the sample. The value produced by the aggregator is M^*.

The RANBAR algorithm has four parameters that have not been defined yet. These are the size s of the initial set S, the required size t of the consensus set S^*, the maximum permitted number f of iterations and the error tolerance δ.

The size s of the initial set is desired to be as small as possible according to the RANSAC paradigm. For the RANBAR algorithm, we need to establish the theoretical histogram of a Gaussian distribution. The Gaussian distribution has two parameters, the expected value θ and the standard deviation σ. A rough estimate for the expected value can be based on a single element from the sample. Similarly, for an estimate on the standard deviation we need at least two elements. This was the motivation for the choice

$$s = 2 \qquad (5)$$

The required size t of the consensus set is the most important parameter of the algorithm. However, the RANSAC paradigm does not give 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 aggregate at the end will contain a high level of error caused by an attacker. If t is too big, the algorithm cannot work because of the high demand on the number of elements in the final set. In general, we have no information about the percentage of compromised nodes, but we require the algorithm to work even in extreme situations, e.g., when only half of the network is uncompromised. That is why we have chosen

$$t = \frac{n}{2} \qquad (6)$$

where n is the total number of sensor nodes in the network.

The maximum number f of iterations can be determined analytically in the following way.

$$f \cdot P(good\ fit) = 1, \qquad (7)$$

since we require to have one good fit in f trials. Here $good\ fit$ is defined as

$$\begin{aligned} \{good\ fit\} &= \{\theta - \epsilon_1 \leq \hat{\theta} \leq \theta + \epsilon_1\ ;\ \sigma^2 - \epsilon_2 \leq \hat{\sigma}^2 \leq \sigma^2 + \epsilon_2\} \\ &= \{\hat{\theta} \cong \theta\ ;\ \hat{\sigma}^2 \cong \sigma^2\}, \qquad (8) \end{aligned}$$

where ϵ_1 and ϵ_2 are small. Thus

$$\begin{aligned} P(good\ fit) &= P(p(x) \cong h(x)) &(9) \\ &= P(\hat{\theta} \cong \theta, \hat{\sigma}^2 \cong \sigma^2) &(10) \\ &= P(\hat{\theta} \cong \theta) \cdot P(\hat{\sigma}^2 \cong \sigma^2) &(11) \end{aligned}$$

since the sample is i.i.d. and therefore the Fisher-Bartlett theorem is applicable in the last step. The distribution of $\hat{\theta}$ is $\mathcal{N}(\theta, \frac{\sigma}{\sqrt{s}})$, therefore

$$\begin{aligned} P(\hat{\theta} \cong \theta) &= \frac{1}{\sqrt{\pi}\sigma} \int_{\theta-\epsilon_1}^{\theta+\epsilon_1} e^{-\frac{(x-\theta)^2}{\sigma^2}} dx &(12) \\ &= \frac{1}{\sqrt{\pi}\sigma} \int_{-\epsilon_1}^{\epsilon_1} e^{-(\frac{x}{\sigma})^2} dx &(13) \end{aligned}$$

As a consequence of the Fisher-Bartlett theorem, $\frac{s-1}{\sigma^2}\hat{\sigma}^2$ follows the Chi Square distribution with $s-1$ degrees of freedom, thus the probability density function of $\hat{\sigma}^2$ is $\frac{s-1}{\sigma^2}\chi^2_{s-1}(\frac{s-1}{\sigma^2}x)$, therefore

$$\begin{aligned} P(\hat{\sigma}^2 \cong \sigma^2) &= \frac{s-1}{\sigma^2} \int_{\sigma^2-\epsilon_2}^{\sigma^2+\epsilon_2} \chi^2_{s-1}(\frac{s-1}{\sigma^2}x)dx \Big|_{s=2} &(14) \\ &= \frac{1}{\sigma^2} \int_{\sigma^2-\epsilon_2}^{\sigma^2+\epsilon_2} \frac{(\frac{x}{\sigma^2})^{-\frac{1}{2}} e^{-\frac{x}{2\sigma^2}}}{\Gamma(\frac{1}{2})2^{\frac{1}{2}}} dx &(15) \\ &= \frac{1}{\sigma\sqrt{2\pi}} \int_{\sigma^2-\epsilon_2}^{\sigma^2+\epsilon_2} \frac{1}{\sqrt{x}} e^{-\frac{x}{2\sigma^2}} dx &(16) \end{aligned}$$

As it can be seen from (13) and (16), the value of f depends on σ. However we would like to keep σ out of the list of incoming parameters. Thus, we have chosen to determine the value of f with the help of empirical analysis. We have tested the algorithm with a couple of f values and we have found that the choice of f does not really affect the distortion of the final aggregate, but there is a trade-off between f and the probability of finding a good consensus set S^*. If f is small then there is a great risk that the algorithm will not find a suitable model, and by increasing f this probability decreases, however, the running time increases. Based on empirical analysis, the choice of

$$f = 15 \qquad (17)$$

seems to be convenient. We note that in [3] the authors propose a probabilistic calculation method for f. In our case it is not applicable, because we do not know the parameter σ.

The error tolerance δ is defined as the stopping criterion of the algorithm. When d becomes smaller than δ, the repetitive phase of the algorithm ends. Thus, δ can be considered as an accuracy level requirement. If δ is too big, the output of the algorithm can be far from the real expected value of the uncompromised sample. If δ is too small, the algorithm may can not shape $h(x)$ to be enough close to $p(x)$, thus it ends in failure. A suitable error tolerance level δ can be obtained by testing this metric in the unattacked case and for different proportions of compromised nodes. In the test cases, we have tested all the reasonable δ values for some typical attack strengths (denoted by κ), and we have prefered those δ values by which both the average and the variance of the distortion was small. The suitable δ values for different proportions of compromised nodes are listed in Table 1. A value compatible with all choices of κ is

$$\delta = 0.3 \qquad (18)$$

is a proper choice for all cases.

Hereby we have defined all the parameters of the RANBAR algorithm. In Section 6 we present our simulation results.

Table 1: The suitable δ values for different κ values

κ	δ
0	0.25 - 0.35
0.05	0.3
0.1	0.25 - 0.3
0.15	0.2 - 0.3
0.25	0.2 - 0.3
0.35	0.25 - 0.3
0.45	0.2 - 0.3

Table 2: Simulation parameters
μ and σ are unknown to the algorithm

Parameter	Value	Parameter	Value
sample distr.	$\mathcal{N}(\mu, \sigma)$	s	2
n	100 and 1000	t	$\frac{n}{2}$
μ	0	f	15
σ	1 and 5	δ	0.3

6. RESULTS OF THE SIMULATIONS

To prove the effectiveness of the RANBAR algorithm we have tested it in a simulation environment with the help of Matlab. The sample was generated by the *randn* function. The Peak Attacker was simulated by a function which replaces those sample elements to a common value, which correspond to the proportion determined by κ. This replacement was done in the wide surroundings of the real expected value of the sample with fine granularity. This can be considered as building a peak in the histogram of the sample, and moving it slowly from one side to the other. In each of the positions during this movement the distortion is obtained. The distortion is defined as the difference between the real average and the average calculated by the algorithm.

To obtain the maximum distortion reachable by the Peak Attacker, we have made 50 simulation runs for different values of κ (i.e., the proportion of compromised nodes). Figure 1 shows our simulation results. The x axis represents the different values of κ, the y axis corresponds to the distortion. The points in the figure represent the maximum distortion in the average of the numerous simulation runs. The maximum distortion of a simulation run is obtained by taking the maximum among the distortions given by the different peak positions. The parameter values of the simulations are listed in Table 2.
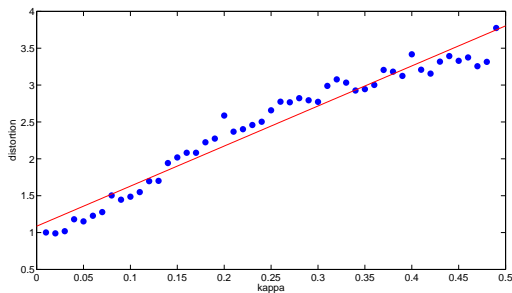

Figure 1: RANBAR simulation result ($\sigma = 5$)

As one can see, the distortion is always upper bounded for $\kappa < 0.5$, in other words, the breakdown point of the RANBAR algorithm is 0.5^2. In addition, the distortion never goes above σ, which means that the attacker has very limited attack strength even if he controls about half of the network. The straight line in Figure 1 is a regression line on the points which is applied to emphasize that the algorithm degrades very slowly while κ goes to 0.5. The explanations of the linear growth of the distortion is the following. With increasing κ, the attacker alters increasing number of sensor readings. If the RANBAR algorithm operates well, these compromised elements will be filtered out, thus the number of elements in the final average calculation decreases. This implies growing inaccuracy, i.e., growing distortion.

In Figure 2, we have compared the resilient aggregation methods suggested by Wagner in [1](i.e., trimming and median) with RANBAR in case of the Peak Attacker. The horizontal axis corresponds to different attack strengths, the vertical axis corresponds to distortion of the algorithms. The stipple line shows how the 5% trimming method performs. The idea of trimming is that if there are some compromised readings then we should drop the upper and the lower 5-5 percent of the readings (5% trimming) and calculate the average of the remaining sample as the estimation of the real average. It is a relatively straightforward method and it works well until the proportion of compromised readings stays below the prespecified trimming level, but if this condition does not hold, the distortion of trimming goes to infinity. In other words, the 5% trimming has a breakdown point of 0.05. Of course, the trimming level can be lifted up to 50%, but with this the accuracy of the method declines. Thus, there is a need to precisely foretell the proportion of compromised readings the method has to encounter, but in a real life situation this information is usually not available.

The same problem occurs with the truncated average too, which is not shown in Figure 2. The truncated average is another naïve approach to handle outlier elements by performing the average calculation only after a mapping phase. The mapping of element x to the $[l, u]$ interval is done by the following rule. Let $trunc_{[l,u]}(x)$ be x if $l \leq x \leq u$, l if $x < l$, and u if $x > u$. The truncated average is the average of the truncated elements (i.e., $trunc_{[l,u]}(x_i)$). The mentioned problem with this method is that it needs l and u as input parameters. In general, the $[l, u]$ interval has to contain the expected value of the distribution of the sample, otherwise the method gives a misleading result. However, one of our preconditions was that we do not know the expected value of this distribution, thus we cannot make appropriate guesses to l and u. This means that the truncated average method is not applicable in our model. Moreover, the truncated average method was found to be worse than the trimmed average method from resiliency point of view in [1], therefore its application has no advantages.

As a consequence, the median is the only rival of RANBAR. The results of the median calculations for different values of κ are represented with the dotted line. The median is defined as the middle element(s) of the sorted sample. The median has a breakdown point of 0.5 similarly to RANBAR. This means that both algorithms can tolerate up to 50 percent of compromised nodes. The lesson of Figure 2

---

[2]The breakdown point is defined as the fraction of data that can be compromised while the error of the estimator remains bounded.
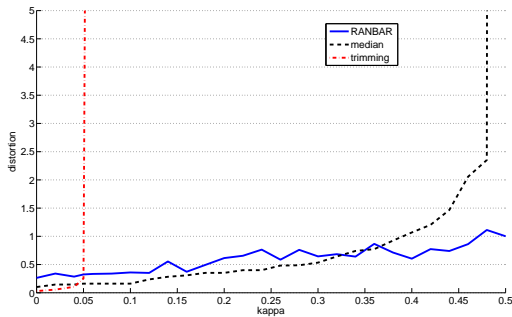
Figure 2: Comparison of RANBAR, median and the trimming calculation ($\sigma = 1$)

is that the median calculation and the RANBAR algorithm produce similarly distorted estimates for $\kappa < \frac{1}{3}$, but for higher κ values the results of the median calculation rapidly decline while the results of RANBAR are still close to the real average of the original sample. The explanation for this is that using the median method one compromised sample element can alter the result by changing it to its neighbour in the row of sorted elements. For more compromised elements the result can be as many indices far from the real median as many compromised readings are presented in the sample. Supposing a normally distributed sample by which the majority of elements are located closely to the real median, the attacker has to compromise about one third of the sample to reach a small distortion, but after this, each compromised node can spoil the median significantly. In contrast to this, the result of RANBAR does not diverge notably from the real average even for high values of κ. Recall that for example $\kappa = \frac{1}{3}$ does not mean that the attacker controls everything in $\frac{1}{3}$ of the network (e.g., he cannot disrupt the communication protocols), but he is able to alter the readings of $\frac{1}{3}$ of the sensors.

In conclusion, we can summarize the result of the comparison by noting that while trimming is a natural way to defend against outliers, it is not easy to determine the suitable level of trimming. The median calculation incurs only a small computation overhead and still produces precise estimates for small proportion of compromised nodes, while the RANBAR algorithm is applicable in all conceivable attack strength cases with more computational overhead. However, RANBAR always produces reliable estimates even for a higher proportion of compromised nodes.

7. CONCLUSION AND FUTURE WORK

We have developed an algorithm for resilient aggregation in sensor networks which can handle a large proportion of compromised sensor readings. We have tested our algorithm in several cases with different parameter values and we have found that it works well for arbitrary attacker power in a simple attacker model. We have compared our algorithm with known robust estimators and we have found that our RANBAR algorithm outperforms the median and the trimmed average for a higher proportion of compromised nodes.

The work presented in this paper is in progress. We intend to further study the behavior of our model in other attacker models. We also intend to examine scenarios where in-network processing is applied. Another interesting future direction that we intend to explore is to consider redundant or highly correlated sensor readings. We believe that assuming correlated measurements will further limit the capabilities of the adversary, as attack detection becomes easier, especially if the adversary is not aware of which sensors are correlated.

Acknowledgement

8. REFERENCES

[1] D. Wagner. Resilient aggregation in sensor networks. In *Proceedings of the ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN)*, October 2004.

[2] L. Buttyan, P. Schaffer, I. Vajda. Resilient Aggregation with Attack Detection in Sensor Networks. In *Proceedings of the Fourth IEEE International Conference on Pervasive Computing and Communications*, March 2006.

[3] M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Communications of the ACM*, Vol. 24, No. 6, pp. 381-395., June 1981.

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

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

[6] A. J. Lacey, N. Pinitkarn, N. A. Thacker. An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2000.

[7] O. Chum, J. Matas. Randomized RANSAC with $T_{d,d}$ test. In *Proceedings of the 13th British Machine Vision Conference (BMVC)*, September 2002.

[8] J. Yan, M. Pollefeys. Articulated Motion Segmentation Using RANSAC With Priors. In *Proceedings of the ICCV Workshop on Dynamical Vision*, 2005.

[9] M. Zuliani, C. S. Kenney, B. S. Manjunath. The MultiRANSAC Algorithm and its Application to Detect Planar Homographies. In *Proceedings of the IEEE International Conference on Image Processing*, September 2005.

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

[11] J.-Y. Chen, G. Pandurangan, D. Xu. 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, 2005.

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

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

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