Distributed Reputation System for Tracking Applications in Sensor Networks

Tanya Roosta, Marci Meingast, Shankar Sastry
{roosta,marci,sastry}@eecs.berkeley.edu
EECS Department, University of California, Berkeley

Abstract*— Ad-hoc sensor networks are becoming more common, yet security of these networks is still an issue. Node misbehavior due to malicious attacks can impair the overall functioning of the system. Existing approaches mainly rely on cryptography to ensure data authentication and integrity. These approaches only address part of the problem of security in sensor networks. However, cryptography is not sufficient to prevent the attacks in which some of the nodes are overtaken and compromised by a malicious user. Recently, the use of reputation systems has shown positive results as a self-policing mechanism in ad-hoc networks. This scheme can aid in decreasing vulnerabilities which are not solved by cryptography. We look at how a distributed reputation scheme can benefit the object tracking application in sensor networks. Tracking multiple objects is one of the most important applications of the sensor network. In our setup, nodes detect misbehavior locally from observations, and assign a reputation to each of their neighbors. These reputations are used to weight node readings appropriately when performing object tracking. Over time, data from malicious nodes will not be included in the track formation process. We evaluate the reputation system experimentally and demonstrate how it improves object tracking in the presence of malicious nodes.

I. Introduction

Sensor networks are wireless networks that lack any pre-defined structure, and consist of many small, wireless devices called motes. These motes (sensors) gather information from the environment and communicate the data back to a central unit called the base station. These networks have demonstrated their usefulness in many applications including health monitoring, data acquisition in hazardous environments, airborne plume detection and surveillance. However, sensor networks are more vulnerable to security attacks than other ad-hoc networks due to their unattended nature. For example, it is conceivable that an adversary can physically capture a subset of nodes in the network and use them to inject faulty data into the system. Cryptographic methods, such as TinySec [1], solve some of the security issues, but they are not effective once the adversary has taken over a node and has access to the cryptographic keys. The adversary can then use this compromised node to engage in communication with the network without being detected.

This work was supported in part by TRUST (The Team for Research in Ubiquitous Secure Technology), which receives support from the National Science Foundation (NSF award number CCF-0424422) and the following organizations: Cisco, ESCHER, HP, IBM, Intel, Microsoft, ORNL, Qualcomm, Pirelli, Sun and Symantec.

The concept of reputation has been used in many fields such as, economy, sociology, and computer science. Reputation systems have proven useful as a self-policing mechanism to address the threat of compromised entities. They operate by identifying selfish nodes, and isolating them from the network. Centralized reputations systems were popularized by the internet [4], [14], [15]. An example of this system is Ebay's rating system [14]. Decentralized methods were then created for use in ad hoc networks [10]. The *CORE* reputation system [12] and the CONFIDANT protocol [22] use a watchdog module at each node to monitor the forwarding rate of the neighbors of the node. If the node does not forward the message, its reputation is decreases, and this information is propagated throughout the network. Each node also uses the second hand information from other nodes to find the overall reputation of a node. Over time the bad behaving nodes are less trusted and will not be used in forming reliable paths for routing purposes. These two protocols differ in how they use the second hand information, how to punish bad behavior and how to instill trust for the node which misbehave temporarily. A formal model for trust in dynamic networks is introduced by Carbone et. al [17]. The most relevant work to ours is presented in [23]. The authors suggest a high level reputation system frame work for sensor networks. The main difference between our work and [23] is that the authors only suggest a 'watchdog' mechanism to determine the reputation of each node. They state that there is no unifying way in which reputations can be assigned, i.e. the mechanism to assign reputation has to be context dependent and varies based on the application at hand. Our contribution in this paper is to develop the assignment mechanism for the specific case of multi-object tracking application in sensor networks.
Multiple-object tracking is a canonical application of sensor networks. It demonstrates multiple aspects of sensor networks, for example, event detection, sensor data aggregation, multi-hop communication, and decision making process. The task of tracking multiple objects in a sensor

network is challenging due to a number of constraints on sensor nodes, such as short-range sensing and communication, and limited amount of memory, computational power, and energy resources. Finally, since a sensor network surveillance system operates autonomously without human operators, it requires a tracking algorithm that is capable of tracking an unknown number of targets. However, the tracking algorithms that have been developed for sensor networks, such as [5], [24], do not explicitly consider the possibility of attacks on sensor networks, and the effects of faulty sensor observation on the performance of their tracking algorithm. In this paper, we focus on the problem of secure tracking in sensor networks, and develop a reputation system to aid mitigate the effects of contaminated sensor observations. We take the multi-object tracking algorithm that was proposed by authors in [5] as the baseline algorithm and we develop our reputation system for this tracking algorithm. It is worth mentioning that different tracking algorithms in sensor networks follow similar procedures to perform object tracking. As a result, the algorithm in [5] is a representative multi-object tracking algorithm in sensor networks.

The rest of the paper is organized as follows. In section 2, we describe the problem as well as the threat model. In section III summarize the basic operations of the tracking algorithm in sensor networks. In section IV, we describe the reputation assignment mechanism that we have developed for the tracking application. In section V, we present the results of testing our method in simulation.

II. Problem Statement and Threat Model

In this work, we consider the problem of secure and robust multi-object tracking in sensor networks. We assume that the nodes are equipped with cryptographic keys for secure communication [1].

Our threat model is as follows: the adversary has physically captured a subset of the nodes, and therefore has access to the network keys and can participate in communication with other nodes without being detected. The objective of the adversary is to use the compromised nodes to inject faulty data into the network, where the data refers to the signal strength that each sensor detects(observation of the sensors). We look at the effect of this type of attack on a multi-object tracking algorithm that has specifically been developed for sensor networks [5], [6].

In this paper we consider 'decentralized trust' as opposed to 'centralized trust' [25], meaning there is no global entity that assigns the reputation values, and all the nodes contact this central unit to find the reputations of the other nodes. In contrast, the 'decentralized trust' refers to the situation where each node is responsible for determining the reputations itself. In addition, we deal with 'reactive' reputation computation [25], meaning the nodes compute the reputation only when they become leaders, as will be explained in the later sections.

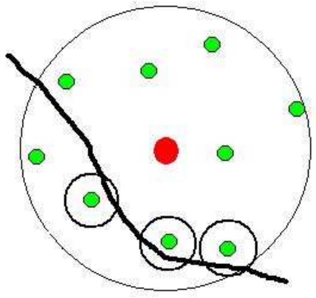


Fig. 1. The red node in the middle of the figure is a supernode which can communicate with all the regular node within its communication range (the large circle). When an object passes through the sensor field, the regular nodes close to the track of the moving object pick up a signal. The sensing range of the regular nodes is shown with the small circles.

III. Basic Operation of Multi-object Tracking

In this paper, we base the design of our reputation system on the multi-object tracking application. Therefore, we briefly describe the basic operations of the multi-object tracking algorithm [5], [6]. We refer the interested reader to [5], [6] for more details of the tracking algorithm.

The sensor network is comprised of N_s nodes. A few of these nodes are called supernodes since they have more communication and computation power. We refer to the rest of the nodes as regular nodes. These sensors are distributed throughout the region $R \subset \mathbb{R}^2$ A single supernode, denoted by S_i, governs a given area $A_i \subset R$ and can communicate with all regular nodes in A_i as well as the neighboring supernodes. Regular nodes, denoted by s_{ij} can communicate with other regular nodes within a range of $2R_s$,where R_s is the sensing range of a node, Figure 1.

When a moving object crosses the sensor field, the sensor nodes that are close to the object track get triggered, i.e. each sensor node records a signal strength which is called the observation of that node. For example, if a node is equipped with passive infrared sensor and an object gets close to the node, the object blocks the infrared port, and as a result the node records a signal strength, or 'senses the object'.

The observations from different nodes are combined to form object tracks using a distributed tracking algorithm. The way the tracking algorithm works is to keep a state variable for each object it is tracking. The track for the object is formed by finding the most likely path given the

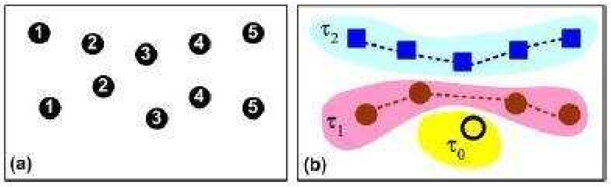


Fig. 2. Partitioning the observations of the sensors over time into object tracks. This figure is taken from [5].

observations[1]. The tracking algorithm developed in [5], [6] is a hierarchical algorithm that consists of a data fusion component at the node level(local) and a data association component at the supernode level(global). This algorithm is capable of tracking multiple objects without knowing the number of objects a priori. The number of objects is estimated along with estimating the tracks.

Our reputation mechanism is specifically designed to deal with the local data fusion. We describe these two steps of the tracking algorithm [5], [6] in more detail in the following subsections. From this point on, we refer to regular nodes simply as nodes.

A. Data Fusion

The data fusion component takes care of aggregating each of the observations by nodes in a local area into a single, fused observation that can be used by the data association component at the supernode level. We denote the location of the i-th node by s_i and the recorded signal strength by z_i. The sensor model used is:

$$z_i = \begin{cases} \frac{\beta}{1+\gamma\|s_i-x\|^\alpha} + w_i, & \text{if } \| s_i - x \| \leq R_s \\ w_i, & \text{if } \| s_i - x \| > R_s, \end{cases} \quad (1)$$

where α, β, and γ are internal parameters specific to the sensor hardware, and x is the position of the object with respect to a given Cartesian coordinate system. The signal strength z_i is normalized so that w_i has the standard Gaussian distribution, $w_i \sim N(0,1)$. If z_i is below a given threshold, namely η, the sensor can not differentiate the reading from noise, and therefore, the reading is not used in the data fusion.

Each sensor looks at the incoming signal strength readings from its neighbors. If the number of incoming readings is below some threshold, the sensor does not have enough neighbors to validate its own reading. Thus its reading will not be incorporated in the data fusion outcome. For example, in [5], the authors suggest that each node has at least 4 neighbors. If the sensor has enough incoming readings, it will compare its reading to that of its neighbors' to determine if it has the highest signal strength [5].

[1]This is accomplished using a Bayesian inference framework and maximizing the posterior probability estimation of the track given the observations. The general approach to finding the posterior probability estimation is using Markov Chain Monte Carlo (MCMC).

The sensor that has the highest signal strength, declares itself the leader and fuses its observation with those of its neighbors. Given that sensor s_{i0} is the leader, i.e. z_{i0} is larger than all incoming readings $z_{i1}, z_{i2}, ..., z_{ik}$, the position of the object is determined as follows:

$$\hat{s}_{obj} = \frac{\sum_{j=1}^{k} z_{ij} s_{ij}}{\sum_{j=1}^{k} z_{ij}}, \quad (2)$$

The logic behind this equation is as follows: the higher the signal strength recorded by a node, the closer the object is to that node. Therefore, if we take the weighted average of the position of all the nodes that have sensed a moving object, we will get an estimate of the position of the object. The wights used are the signal strength observed by each node. This value is sent to the governing supernode. Each supernode collects the estimated object positions from the leaders in its governing region in order to perform the data association phase.

B. Data Association

Once the supernode has received the fused observations from each of the local areas in its region, data association is performed. The goal is to associate the fused data points to form a track, as shown in Figure 2. The details of the data association step are beyond the scope of this paper, and we refer the reader to [5] for the details.

Each supernode maintains its own set of tracks through performing the data association process. Thus, a single object can have multiple tracks maintained by different supernodes. These multiple tracks from different supernodes are combined at the main supernode, i.e. the base station, using another iteration of the data association algorithm.

The motivation for our work is the following: if the compromised nodes inject faulty observations into the network, the local fused data will become contaminated. Therefore, when the supernodes use these contaminated fused data, the tracks they form will be faulty. As a result, there is a need have a mechanism in place that is able to filter out the bad data, which will help the track formation process.

IV. Reputation System

As authors in [23] have pointed out, the main challenge in designing a reputation system is to develop the mechanism by which the reputation is assigned for the particular application at hand. Our main contribution in this paper is to develop a reputation assignment mechanism to specifically address the data fusion phase of the tracking algorithm.

When a malicious node injects faulty observations into the network, it can throw off the value of the fused data calculated through Equation 2. This will affect the accuracy of the tracks that are formed as well as the number of the estimated tracks. By assigning reputations to nodes over time, we are able to weight their observations by the corresponding reputation. This approach will result in minimizing the effect of faulty observations on the data fusion by suppressing the readings from the malicious nodes.

Our reputation assignment scheme attempts to give reputations to nodes at a local level and does not attempt to solve the problem of secure leader selection. As mentioned in III-A, the node that claims to have the highest signal strength becomes the leader. In order to assure that a compromised node does not become the leader, we have to develop a secure leader election protocol, and have methods in place at the supernodes to determine the compromised nodes and filter out their data. Secure leader election is outside the scope of our work, but it is worth noting that there are standard distributed coin-flipping algorithms in the literature, using cryptographic commitments.

In our scheme, every time a node becomes the leader, it updates the reputation of its own neighbors, and keeps a table of the reputations values. There is no sharing of the reputation tables among nodes, i.e. no use of the second hand information.

At each time step, the nodes which have readings will receive an instantaneous reputation rating. For each node, the instantaneous rating is combined with its ratings from the previous time steps to form an overall reputation for the node. Observations from a node are then weighted by this overall reputation when data fusion is done.

A. Reputation Assignment Mechanism

During a single time step,t, the leader of a local neighborhood assigns reputations to those nodes it receives readings from. We call this the instantaneous reputation. To determine what the reputation of each node is, the leader uses a method similar to RANSAC(Random Sample Consensus) [18].

RANSAC relies on random sampling selection to search for the best model to fit a set of points that is known to contain outliers. In effect, RANSAC can be considered to seek the best model that maximizes the number of inlier data. The following is the set of steps taken by the RANSAC algorithm to find the best model parameters:

1) Randomly select a subset of the data points of size m and build the initial model from these points
2) Determine the set of data points that are within ϵ of the model and call this set M. This set defines the inliers of the original data set.
3) If $|M|$ is greater than a threshold T, we need to re-estimate the model using all the points in M, and the algorithm terminates.
4) If $|M|$ is less than T, select a new subset and repeat 2.
5) After N trials the largest M is selected, and the model is re-estimated using all the data points in M.

In order to use RANSAC, we need to determine the number of points in the small subsets, the number of iterations, and the threshold used to identify a point that fits well.

Following the RANSAC approach, our scheme chooses subsets of neighboring nodes at random. These subsets are then used to find the estimate of the object location using Equation 4. The size of each subset and the number of subsets required to get a good estimate can be found using the following formula [18]:

$$N = \frac{\log 1 - p}{\log 1 - (1-\epsilon)^s}, \tag{3}$$

where N is the number of subsets, s is the number of samples in each subset, and ϵ is the percentage of contamination. p is the probability of having at least one subset free of outlier data. This probability is usually set to 0.99, meaning with probability 0.99 there are no outliers in the data sample. The justification for the above equation can be found in [18]. In our problem, we need to ensure that at least half of the chosen subsets are free of outliers, so as to get a good overall estimate. This is due to the fact that the median is used as the best estimate. Therefore, we need to have $(1-\epsilon)^s < 0.5$, or $\epsilon \leq \frac{0.69}{s}$.

The reason for using the above scheme is to find the best possible estimate of the position of the object given the observed data. Since there is no model to fit the data to, as RANSAC suggests, we need to find a 'best guess' for the object location as a reference point in order to later assign the reputation to each node.

Equation 2 is used on each selected subset to find one value for the fused observation. We call this value S^i_{fused} to refer to the fused value for the i^{th} subset. The only difference in calculating S^i_{fused} is that we use the overall reputation for each node as the weighting factor in the summation, i.e.:

$$\hat{s}_{obj} = \frac{\sum_{j=1}^{k} z_{ij} s_{ij} l_{ij}^{t-1}}{\sum_{j=1}^{k} z_{ij} l_{ij}^{t-1}}, \tag{4}$$

where l_{ij} is the overall reputation of node j from neighbor i.

Once the estimated object locations are determined from all the subsets, the median of those estimates is calculated. Namely we find the median of S^i_{fused}, where $i \in \{1, ..., N\}$, and N is calculated from Equation 3. We use the median as opposed to mean since it has been shown analytically [20], [21] that the median is a robust estimator with a *breakdown point* of 50% in contrast to the mean

which has a breakdown point of zero. [2]
The median, m, and the corresponding subset of nodes that give the value m are assumed truthful. We call this subset S_{trust}. S_{trust} is used as a starting point for determining the reputations of the remaining nodes. If multiple subsets of nodes result in the same value of the median, we choose the subset of nodes with the lowest variance as the starting point. There are two counters, $(\alpha_{ij}, \beta_{ij})$, which are updated for the instantaneous reputation. Positive reputation is kept by α_{ij} and negative reputation is kept by β_{ij}. The nodes in S_{trust} receive an instantaneous reputation of $(1, 0)$ since we assume they are truthful. The leader node, which will be passing the fused signal to the supernode, also receives a $(1, 0)$ reputation since the data it has will be used in tracking calculations.
To determine the reputation of the remaining nodes in the neighborhood, we pick one node,s_{ij}, at a time and add it to the subset S_{trust}. Then the weighted sum of this node along with the nodes in S_{trust} is found using 4. We call the result of this calculation $\hat{m}_{ij}$. This new estimate, $\hat{m}_{ij}$, is compared with the median m and the reputation for the node s_{ij} is assigned as follows:

$$(\alpha_{ij}, \beta_{ij}) = \begin{cases} (1, 0), & \text{if } |\hat{m}_{ij} - m| = 0 \\ (\frac{T - |\hat{m}_{ij} - m|}{T}, 0), & \text{if } |\hat{m}_{ij} - m| < T \\ (0, 1), & \text{if } |\hat{m}_{ij} - m| > T, \end{cases} \quad (5)$$

where T is a threshold to determine how far $\hat{m}_{ij}$ can be pulled away from the median m, while node s_ij still gets a positive rating. Note that when $|\hat{m}_{ij} - m| < T$, the node does not get a positive rating of 1. Instead it gets a fraction proportional to how close it is to the median. The reason for this assignment is that a node that has a closer value to the median should be rated higher than a node that gives an estimate further away from the median. The threshold value T has to be adjusted so that it does not assign low rating to nodes that have a noisy observation within the reasonable bounds. At the same time, it can not be too large to allow the compromised nodes to inject faulty readings. The nodes whose estimate $\hat{m}_{ij}$ falls outside the threshold get a negative instantaneous reputation.

B. Reputation Fusion

The instantaneous reputations for the nodes, calculated in the previous section, are aggregated over time to calculate cumulative positive and negative reputation ratings which we represent by (r^t_{ij}, s^t_{ij}). Old reputation values may not be as relevant for the cumulative ratings, as a node may change behavior over time. Thus, to determine the cumulative ratings, we use a discounting factor, λ, that guarantees the old reputations will be gradually forgotten. The ratings are determined by the following equation:

$$\begin{aligned} r^t_{ij} &= \lambda r^{t-1}_{ij} + \alpha_{ij} \\ s^t_{ij} &= \lambda s^{t-1}_{ij} + \beta_{ij}, \end{aligned} \quad (6)$$

Using the cumulative ratings the overall reputation, l_{ij}, of a node at time t is calculated. The overall reputation varies in the range $[0, 1]$ where 0 represents the most untruthful and 1 represents the most truthful.
In order to determine where a node's overall reputation falls on this scale, we use the Beta distribution. We consider the sequence of observations as a sample from a binomial distribution, i.e. a sequence of independent coin tosses, with a bias parameter *P*. To make the statement clear, the *head* corresponds to an honest node and the *tail* corresponds to a compromised node, and the bias is the overall rating of the node. We can then estimate the rating of a node using Bayesian parameter estimation of the binomial distribution. If we use a Beta distribution as the prior distribution, the formula for calculating the posterior parameter estimate actually coincides with the maximum likelihood estimate. The fact that the posterior probability of binary events is most accurately represented by the Beta distribution, has been shown mathematically in [2]. The Beta distribution is a two parameter distribution whose parameters are denoted by a and b. The parameter a measures the number of successes (r^t_{ij}) and b measures the number of failures (s^t_{ij}). The bias estimate of the underlying binomial distribution, *P*, is given by the mode (average) of the Beta distribution, i.e. $P = \frac{a-1}{a+b-2}$.
The Beta distribution is well suited for our purposes since at each time step a node is either truthful or is lying, which is a binary event. The overall reputation is modeled as the expected value of the Beta distribution [2]:

$$l^t_{ij} = \frac{r^t_{ij} - s^t_{ij}}{r^t_{ij} + s^t_{ij} + 2}, \quad (7)$$

This give us a reputation ranging from [-1,1]; therefore, we need to rescale this interval to the interval $[0, 1]$. This overall reputation is used in the next time step as a weighting factor in (4) for calculating the median.
Two comments are in order regarding our reputation system. We are not considering second hand information, i.e. the information coming from the neighbors, since that will give the compromised nodes a window of opportunity to attack and degrade the reputation of their neighbors. The use of second hand information is useful if each node has a scheme for filtering out badmouthing. This in turn will add to the overhead incurred by the reputation system. Therefore, we chose not to add the second hand information in the reputation updating formula. Secondly, the reputations are stored locally at each leader node.

[2]The breakdown point of an estimator refers to how resistent the estimator is to the percentage of outliers and contaminated samples. The higher this point is, the more robust the estimator will be.

Every time a node becomes the leader, it will update its reputation table. This scheme avoids having to broadcast the reputation tables of each node to its neighbors. We speculate that the local reputations will converge to the true reputation values over time. This is due to the fact that over time almost all the nodes will become leaders. The validation of this speculation is the focus of our future work.

V. Simulation Results

To provide empirical evidence on the performance of our reputation system, we performed a number of simulations. Again, in this work we do not consider secure leader election. Therefore, in the simulations we do not allow for the compromised nodes to claim to be leaders.

In our settings, the surveillance region is a square grid of size $50mx50m$. There is one node placed at each corner of each square. Therefore, there is a total of 2500 nodes in the simulation grid. The number of objects we want to track is n_i. The sensing range of the sensors,R_s is set to $1.5m$. This sensing range has been shown to be optimal in terms of low estimation errors for different speeds of the moving targets [5]. We are simulating real sensors in our experiments, therefore, the sensor readings are noisy, and the noise is represented by a Gaussian standard distribution with mean of zero and variance of one.

The metric used to quantify the performance of the multi-object tracking algorithm is the average error in the number of tracks estimated by the algorithm compared to the actual number of tracks, ϵ_K where [5]:

$$\epsilon_K = \frac{1}{W} \sum_{w=1}^{W} \mid K_w - \tilde{K_w} \mid$$

where $\tilde{K_w}$ is the actual number of tracks at time w, K_w is the number of tracks the algorithm estimated, and W is the total time of simulation.

In our simulations we look at multiple scenarios. In the first scenario we keep the number of tracks,n_i, constant and change the number of compromised nodes from 250 to 1000. The results are shown in Figure 4 for differing values of n_i. In our second scenario, we fix the number of compromised nodes and vary the sensing radius,R_s, from $1.5m$ to $3m$ as shown in Figure 5. In these scenarios we use a threshold value, $T = 0.4$, and $s = 3$, where s is the number of nodes in each subset.

We can see from Figure 3 that the reputation framework gets rid of many of the spurious estimated tracks and decreases the size of the ones that remain. In addition, the estimated tracks associate with the ground truth are more accurate and more closely follow the actual movement of the object.

In our simulations, we did not allow for the compromised nodes to become leaders. As we discussed in the previous section, filtering out the compromised leaders requires either a secure leader election process, or a centralized scheme at the supernodes to keep reputations for the leader nodes.

VI. Conclusion

In this paper we described a local scheme that uses a best-effort approach to filter out the malicious nodes when computing the local estimates of an object's location. This approach depends on collecting readings from a given area and using the redundancy in the data to verify the trustworthiness of a node's observation. We use a RANSAC-like scheme along with the Beta distribution to design a mechanism for assigning reputations to the nodes at a local level. The reputation gets updated every time a node sends in a new observation. The simulation results indicate that this reputation mechanism is successful in partially filtering out the bad readings. As part of the future work, we plan to extend the reputation system to include a global scheme, i.e. at the supernode level, in order to filter out the compromised leaders.

References

[1] C. Karlof, N. Sastry, and D. Wagner, *TinySec: A Link Layer Security Architecture for Wireless Sensor Networks*,Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004), pages 162-175, November 2004.

[2] A. Josang and R. Ismail, *The Beta Reputation System*,Bled Electronic Commerce Conference,June 2002.

[3] D. Liu, P. Ning, and W. Du,*Attack-Resistant Location Estimation in Sensor Networks*,International Symposium on Information Processing in Sensor Networks,April 2005,Los Angeles, CA.

[4] M. Fan, Y. Tan, and A.B. Whinston,*Evaluation and Design of Online Cooperative Feedback Mechanisms for Reputation Management*,IEEE Transactions on Knowledge and Data Engineering,February 2005, vol. 17,no. 2,pages 244-254

[5] S. Oh, L. Schenato, and S.Sastry,*A Hierarchical Multiple-Target Tracking Algorithm for Sensor Networks*,IEEE International Conference on Robotics and Automation, April 2005,Barcelona, Spain.

[6] S. Oh, S. Russell, and S. Sastry,*Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems*,IEEE International Conference on Decision and Control,December 2004,Paradise Island, Bahamas.

[7] P. Yau and C.J. Mitchell, *Reputation Methods for Routing Security*, Symposium on Trends in Communication, October 2003

[8] J. Mundinger and Jean-Yves Le Boudec,*Analysis of a Reputation System for Mobile Ad-Hoc Networks with Liars*,International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks,April 2005.

[9] S. Buchegger and J. Le Boudec,*Self-Policing Mobile Ad Hoc Neworks by Reputation Systems*,IEEE Communications Magazine,July 2005,pages 101-107

[10] J. Mundinger and J. Le Boudec, *Analysis of a Robust Reputation System for Self-Organized Networks*, University of Cambridge,Statistical Laboratory Research Report,January 2004.

[11] Z. Li, W. Trappe, Y. Zhang, and B. Nath, *Robust Statistical Methods for Securing Wireless Localization in Sensor Networks*, International Symposium on Information Processing in Sensor Networks, April 2005.

[12] P. Michiardi and R. Molva, *Core: A collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks*, Conference on Communications and Multimedia Security,September 2002,

[13] L. Mui, M. Mohtashemi, and A. Halberstadt,*Notions of Reputation in Multi-Agent Systems: A Review*, International Joint Conference on Autonomous Agents and Multiagent Systems,2002.

[14] P. Resnick and R. Zeckhauser, *Trust Among Strangers in Internet Transactions: Empirical Anlysis of Ebay's Reputation System*,Advances in Applied Microeconomics: The Economics of the Internet and E-Commerce,vol. 11,pages 127-157,November 2002.

[15] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, *Reputation Systems*,Communications of the ACM,vol. 43(12),pages 45-48, 2000.

[16] P. Yau and C.J. Mitchell,*Security Vulnerabilities in Ad Hoc Networks*, International Symposium on Communication Theory and Applications,July 2003.

[17] M. Carbone, M. Nielsen, and V. Sassone, *A Formal Model for Trust in Dynamic Networks*, IEEE International Conference on Software Engineering and Fromal Methods,2003.

[18] H.J. Kuxhner and G.G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*, Springer-Verlag,Second Edition,2003.

[19] M.A. Fischler and R.C. Bolles, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, Comm. of the ACM,vol. 24, pages 381-395,1981.

[20] D. Wagner, *Resilient Aggregation in Sensor Networks*, ACM Workshop on Security of Ad Hoc and Sensor Networks,October,2004.

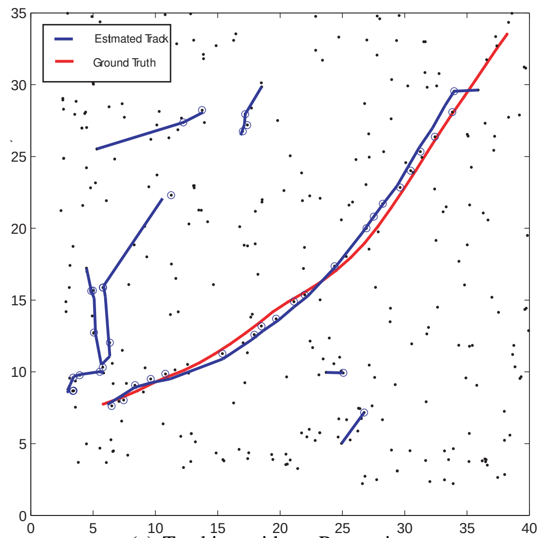
[21] P. Rousseeuw and A. Leroy,*Robust Regression and Outlier Detection*,John Wiley and Sons, Inc.,1987.

[22] S. Buchegger and J. Le Boudec, *Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes - Fairness in Dynamic Ad-Hoc Networks*, IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing,June 2002.

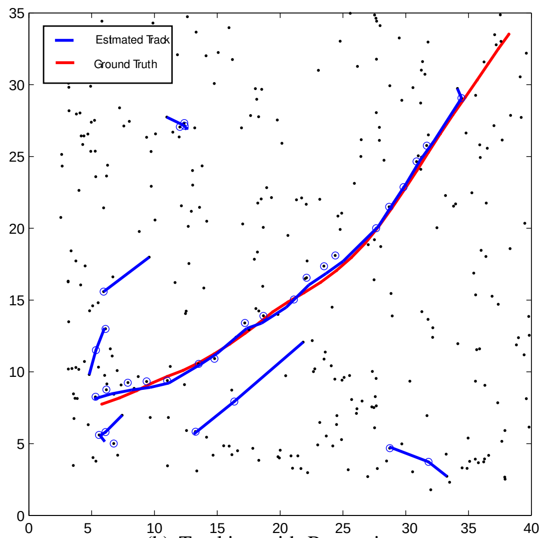
[23] S. Ganeriwal, M. B. Srivastava, *Reputation-based framework for high integrity sensor networks*, ACM Security for Ad-hoc and Sensor Networks,2004.

[24] Chu, M.; Mitter, S. K.; Zhao, F. *Distributed multiple target tracking and data association in ad hoc sensor networks*. 6th International Conference on Information Fusion; 2003 July 08-11; Cairns; Australia.

[25] George Theodorakopoulos and John S. Baras. On trust models and trust evaluation metrics for ad hoc networks, IEEE Journal on Selected Areas in Communications, Volume 24, Issue 2, 318-328, February 2006.

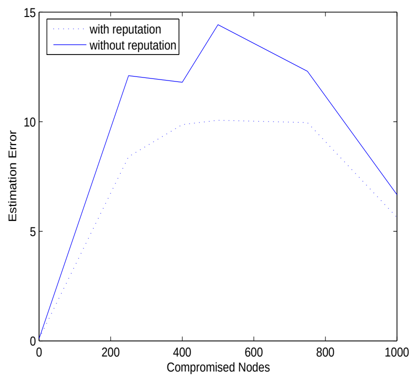


(a) Tracking without Reputation

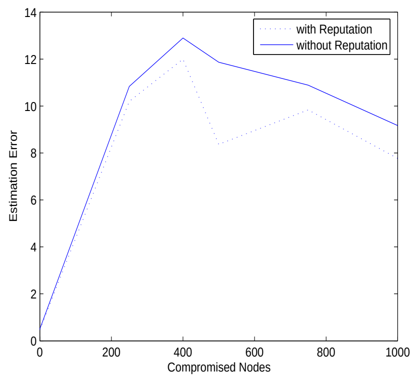


(b) Tracking with Reputation

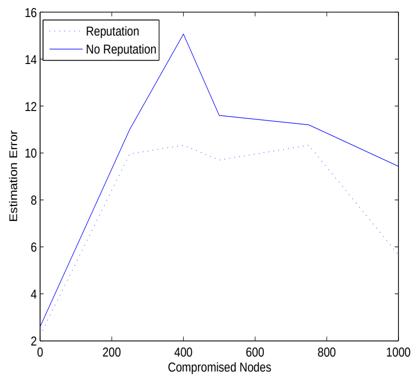
Fig. 3. The estimated object track compared to ground truth. This shows a subset of the 50 x 50 node grid of sensors where the ground truth is shown in red and the estimated track is shown in blue.



(a) 3 Objects

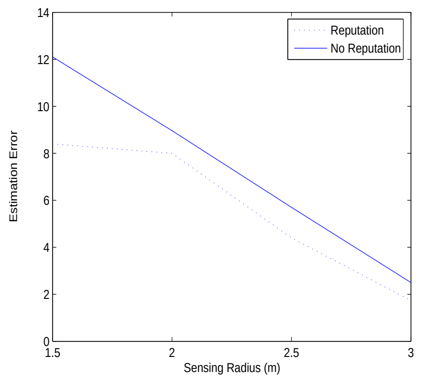


(b) 6 Objects

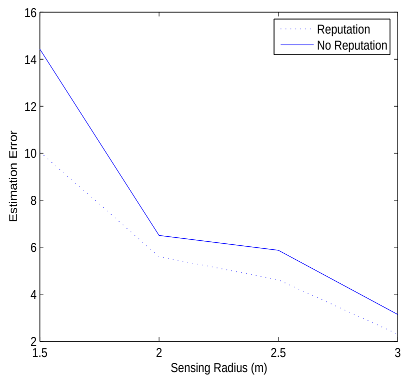


(c) 15 Objects

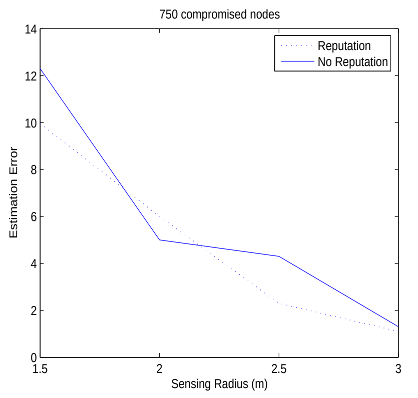
Fig. 4. Estimation error ϵ_K



(a) 250 Compromised Nodes



(b) 500 Compromised Nodes



(c) 750 Compromised Nodes

Fig. 5. Estimation error ϵ_K