This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2010 proceedings.

Robust State Estimation Under False Data Injection in Distributed Sensor Networks

Shanshan Zheng, Tao Jiang, John S. Baras
Institute for Systems Research
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD, 20742
Email: {sszheng,tjiang,baras}@umd.edu

***Abstract*—Distributed sensor networks have been widely employed to monitor and protect critical infrastructure assets. The network status can be estimated by centralized state estimation using coordinated data aggregation or by distributed state estimation, where nodes only exchange information locally to achieve enhanced scalability and adaptivity to network dynamics. One important property of state estimation is robustness against false data injection from sensors compromised by attackers. Different from most existing works in the literature that focus on centralized state estimation, we propose two novel robust distributed state estimation algorithms against false data injection. They are built upon an existing distributed Kalman filtering algorithm. In the first algorithm, we use variational Bayesian learning to estimate attack parameters and achieve performance similar to a centralized majority voting rule, without causing extra communication overhead. In the second algorithm, we introduce heterogeneity into the network by utilizing a subset of pre-trusted nodes to achieve performance better than majority voting. We show that as long as there is a path connecting each node to some of the pre-trusted nodes, the attackers can not subvert the network. Experimental results demonstrate the effectiveness of our proposed schemes.**

I. Introduction

A distributed sensor network consists of a set of spatially scattered sensors that can measure various properties of the environment, formulate local and distributed inferences, and make responses to events or queries [1]. It represents an evolving frontier in technology and can be deployed to monitor and protect critical infrastructure assets, such as power grids, automated railroad control, water and gas distribution, etc. In a distributed sensor network, the data gathered by sensors are analyzed to provide state estimation of the network status. For instance, a surveillance system tracks objects based on different kinds of signals measured by sensors, such as acoustic and video signals. The output of state estimation may include positions, speeds, moving directions of the objects, which will be used for the response algorithm of the system. Due to the unattended operating environment of sensor networks, it is possible for an attacker to compromise sensors to introduce malicious measurements, which we call *false data injection*. The false measurements can affect the outcome of state estimation and mislead the system into making wrong decisions.

Detecting and purging false measurements injected by compromised nodes is a challenging research problem in sensor networks [2]. The energy and cost constraints restrict the deployment of tamper resistant hardwares for the whole network. Therefore, once a node is compromised, all the security information stored in the node will be exposed to the attackers. The normal nodes cannot distinguish the attackers by only using cryptographic techniques. Most existing works use *majority voting* based schemes. Ye et al. [2] presented a statistical en-route filtering mechanism in which the truthfulness of data from each node is determined via collective decision-making by multiple nodes. Zhu et al. [3] proposed an interleaved hop-by-hop authentication scheme where pairwise keys are established between nodes $t+1$ hops away and up to t compromised nodes can be tolerated. In [4], Shukla et al. presented a secure statistical scheme to distinguish data transience from false injection in a clustered sensor network by utilizing statistical digests sent from each sensor. For additional results see [5]–[7]. However, all these works rely on a hierarchical architecture and central coordination among sensors, which limits the sensor network's scalability due to the large number of nodes and the volatility of the network.

In this paper, we focus on robust *distributed state estimation* against false data injection, where each node in the network tries to estimate the states as accurately as possible, by using only local information. This type of algorithms is usually known as gossip-based algorithms, emerging as an important communication paradigm for large scale distributed systems [8]. Under this setting, how to utilize *majority voting* is not obvious. If each node reports its data to all nodes, the network will be congested by the large communication overhead. We propose a distributed state estimation method with Bayesian learning, which achieves performance similar to a centralized majority voting rule without causing extra communication overhead. One limitation of using majority voting is that the network may be totally subverted when more than half of the sensors are compromised. To overcome this problem, we introduce heterogeneity into the network through a subset of pre-trusted nodes, which are special nodes that can be highly trusted, e.g., sensors equipped with tamper resistant hardwares. We propose a trust-aware scheme, where the trustworthiness of a node is computed according to the truthness and accuracy of the information it provided. The trust propagates from the pre-trusted nodes to other nodes, and the distributed state estimation algorithm will utilize data from nodes according to their trustworthiness. We show that as long as there is a path connecting every node in the network to some of the pre-trusted nodes, the attacker cannot subvert the network even when more than half of the sensors are compromised.

The rest of the paper is organized as follows. We formulate the problem in Section II. Then we describe our Bayesian learning based scheme and trust-aware scheme in Sections III and IV, respectively. Section V presents conclusions and future work.

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

II. PROBLEM FORMULATION

We consider a sensor network composed of n sensors, which is used for state estimation of a linear random process $\mathbf{x}(k+1) = A\mathbf{x}(k) + \mathbf{w}(k)$. $\mathbf{x}(k) \in R^m$ is the state vector, $\mathbf{w}(k) \in R^m$ is the state noise, which accounts for modeling errors, uncertainties or perturbations to the process and is assumed to be Gaussian with 0 mean and covariance matrix Q. The initial state $\mathbf{x}_0$ has a Gaussian distribution with mean $\boldsymbol{\mu}_0$ and covariance matrix M_0. Such a linear random process can represent a physical phenomenon or a moving object. We assume each sensor can sense this process and the sensing model for each sensor is given by $\mathbf{y}_i(k) = H_i\mathbf{x}(k) + \mathbf{v}_i(k)$, where $\mathbf{y}_i(k) \in R^{p_i}$ is the observation made by sensor i, and $\mathbf{v}_i(k) \in R^{p_i}$ is the measurement noise, assumed to be Gaussian with zero mean and covariance V_i. In distributed sensing, the measurements of an individual sensor may only contain partial information of the state, so state estimation requires the cooperation of sensors in the network.

In distributed state estimation, the goal of each sensor is to compute an accurate estimation of the state $\mathbf{x}(k)$ using its local measurements and the information received from its communication neighbors. Distributed Kalman Filtering (DKF) provides a solution for this problem. The main idea of DKF [9], [10] is to use a standard Kalman filter locally, together with a consensus step to ensure that the local estimates agree. However, existing DKF algorithms are derived in a benign setting without considerations of possible malicious attacks. Therefore they are not robust against false data injection. In what follows, we use the DKF algorithm introduced in [10] as a basic algorithm to build our robust distributed estimation algorithms. Let $G = (\mathcal{V}, \mathcal{E})$ be a graph with an adjacency matrix $[g_{ij}]$ that specifies the topology of the sensor network, and $\mathcal{N}_i = \{j \in \mathcal{V} | g_{ij} \neq 0\}$ be the communication neighborhood of sensor i, i.e., the subset of sensors with whom i can exchange information. Let $\mathcal{J}_i = \mathcal{N}_i \cup \{i\}$ denote the inclusive neighbors of node i and $\mathbf{y}(k) = [\mathbf{y}_1(k), \ldots, \mathbf{y}_n(k)]^T$ denote the measurements obtained at all sensors by time k. Then given the observations $\mathbf{y}(1:k) = \{\mathbf{y}(1), \ldots, \mathbf{y}(k)\}$, we denote the state estimates at node i as follows

$$\hat{\mathbf{x}}_i(k) = \mathrm{E}[\mathbf{x}(k)|\mathbf{y}(1{:}k)],\ \bar{\mathbf{x}}_i(k) = \mathrm{E}[\mathbf{x}(k)|\mathbf{y}(1{:}k-1)],$$
$$\hat{M}_i(k) = \mathrm{Cov}(\mathbf{x}(k)|\mathbf{y}(1{:}k)),\ \bar{M}_i(k) = \mathrm{Cov}(\mathbf{x}(k)|\mathbf{y}(1{:}k-1)).$$

The basic algorithm is described in Algorithm I, where ϵ is a small positive constant. For simplicity, we omitted the time index. In each round, node i only needs to broadcast the message $(\mathbf{u}_i, U_i, \bar{\mathbf{x}}_i)$ to its neighboring nodes.

Algorithm I: Basic DKF Algorithm [10]

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i = \boldsymbol{\mu}_0$
2. While new data exists do
3. Locally aggregate data and covariance matrices:
   $\forall\ j \in \mathcal{J}_i,\ \mathbf{u}_j = H_j^T V_j^{-1}\mathbf{y}_j,\ \mathbf{z}_i = \sum_{j\in\mathcal{J}_i}\mathbf{u}_j,$
   $U_j = H_j^T V_j^{-1} H_j,\ S_i = \sum_{j\in\mathcal{J}_i} U_j$
4. Compute the Kalman-consensus estimate:
   $\hat{M}_i = (\bar{M}_i^{-1} + S_i)^{-1}$,
   $\hat{\mathbf{x}}_i = \bar{\mathbf{x}}_i + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i) + \epsilon\hat{M}_i\sum_{j\in\mathcal{N}_i}(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)$
5. Update the state of the Kalman-consensus filter:
   $\bar{M}_i = A\hat{M}_i A^T + Q,\ \bar{\mathbf{x}}_i = A\hat{\mathbf{x}}_i$
6. end while

Now we introduce our threat model. Let $\mathbf{y}^a$ represent the vector of measurements that may contain malicious data, i.e., $\mathbf{y}^a(k) = \mathbf{y}(k) + \mathbf{a}(k)$, where $\mathbf{a}(k) = [\mathbf{a}_1(k), \ldots, \mathbf{a}_n(k)]^T$ is the malicious data added to the original measurements. We call $\mathbf{a}(k)$ the *attack vector*. The i^{th} element of $\mathbf{a}(k)$ being non-zero means that the attacker compromises the i^{th} sensor and replaces its original measurement $\mathbf{y}_i(k)$ with a false measurement $\mathbf{y}_i(k) + \mathbf{a}_i(k)$. The attacker can choose any arbitrary vector as the attack vector $\mathbf{a}(k)$. To illustrate the effect of this attack on the outcome of state estimation, we study an example where the attack vector is drawn from a multi-variate Gaussian distribution with mean $\boldsymbol{\mu}^a = [\boldsymbol{\mu}_1^a, \ldots, \boldsymbol{\mu}_n^a]^T$ and covariance matrix $V^a = \mathrm{diag}(V_1^a, \ldots, V_n^a)$. The distributed estimation of states $\mathbf{x}(k)$ follows the procedure in Algorithm II, where $\hat{\mathbf{x}}^a(k)$ denotes the estimated states using $\mathbf{y}^a(1:k)$, and $\hat{V}_i = V_i + V_i^a$.

Algorithm II: DKF under a Gaussian attack vector

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i^a = \boldsymbol{\mu}_0$
2. While new data exists do
3. Locally aggregate data and covariance matrices:
   $\forall\ j \in \mathcal{J}_i,\ \mathbf{u}_j = H_j^T\hat{V}_j^{-1}(\mathbf{y}_j^a - \boldsymbol{\mu}_j^a),\ \mathbf{z}_i = \sum_{j\in\mathcal{J}_i}\mathbf{u}_j,$
   $U_j = H_j^T\hat{V}_j^{-1}H_j,\ S_i = \sum_{j\in\mathcal{J}_i} U_j$
4. Compute the Kalman-consensus estimate:
   $\hat{M}_i = (\bar{M}_i^{-1} + S_i)^{-1}$,
   $\hat{\mathbf{x}}_i^a = \bar{\mathbf{x}}_i^a + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i^a) + \epsilon\hat{M}_i\sum_{j\in\mathcal{N}_i}(\bar{\mathbf{x}}_j^a - \bar{\mathbf{x}}_i^a)$
5. Update the state of the Kalman-consensus filter:
   $\bar{M}_i = A\hat{M}_i A^T + Q,\ \bar{\mathbf{x}}_i^a = A\hat{\mathbf{x}}_i^a$
6. end while

Comparing Algorithm I and II, we can see that when the sensor nodes are not aware of the existence of false data injection, i.e., assuming $\boldsymbol{\mu}_j^a = 0$ and $\hat{V}_j = V_j$ in Algorithm II, while they are actually not, then even if only one sensor is compromised, the state estimation $\hat{\mathbf{x}}^a(k)$ can deviate from the true state $\mathbf{x}(k)$ arbitrarily or be arbitrarily noisy. This verifies that false data injection can cause serious errors.

III. DKF WITH BAYESIAN LEARNING

Our DKF algorithm with Bayesian learning is inspired by the observation in Algorithm II that the true state can be accurately estimated if the attack parameters are known. In reality, the attack parameters are unknown, but we can use Bayesian learning to estimate them and then follow the same procedure of Algorithm II to obtain a more reliable estimate of the states. Bayesian learning usually involves highly complicated computational techniques such as Markov Chain Monte Carlo. However, the resource constraints in a sensor network requires algorithms to be fast and computationally efficient. Therefore, we propose to use variational Bayesian learning, which is an approximate but efficient method for estimating the marginal likelihood of probabilistic models with latent variables or incomplete data [11].

a) Variational Bayesian Learning: Assuming that $\mathbf{y}$ is the observed data, $\mathbf{x}$ is the latent variables, and $\boldsymbol{\theta}$ is the model parameters, variational Bayesian learning constructs and optimizes a lower bound on the marginal likelihood $f(\mathbf{y})$ using variational calculus. More specifically, the $\log f(\mathbf{y})$ is lower bounded by first introducing any distribution over both latent variables and parameters which has the same support

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

as $f(\mathbf{x},\boldsymbol{\theta}|\mathbf{y})$ does, and then applying Jensen's inequality [11], i.e.,

$$\log f(\mathbf{y}) = \log\int f(\mathbf{y},\mathbf{x},\boldsymbol{\theta})d\mathbf{x}d\boldsymbol{\theta} \geq \int q(\mathbf{x},\boldsymbol{\theta})\log\frac{f(\mathbf{y},\mathbf{x},\boldsymbol{\theta})}{q(\mathbf{x},\boldsymbol{\theta})}d\mathbf{x}d\boldsymbol{\theta}.$$

Using a simple factorized approximation $q(\mathbf{x},\boldsymbol{\theta}) \approx q_{\mathbf{x}}(\mathbf{x})q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$, the variational Bayesian algorithm iteratively maximizes the above lower bound of $\log f(\mathbf{y})$ with respect to the free distributions $q_{\mathbf{x}}(\mathbf{x})$ and $q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$. Elementary variational calculus leads to the following updating rules [11]:

$$q_{\mathbf{x}}^{(t+1)}(\mathbf{x}) \propto \exp[\int \log f(\mathbf{y},\mathbf{x}|\boldsymbol{\theta})q_{\boldsymbol{\theta}}^{(t)}(\boldsymbol{\theta})d\boldsymbol{\theta}],$$
$$q_{\boldsymbol{\theta}}^{(t+1)}(\boldsymbol{\theta}) \propto f(\boldsymbol{\theta})\exp[\int \log f(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})q_{\mathbf{x}}^{(t+1)}(\mathbf{x})d\mathbf{x}].$$

b) DKF with variational Bayesian learning: In the DKF model, the latent variables are the system states to be estimated: $\mathbf{x}(1:k)=\{\mathbf{x}(1),\ldots,\mathbf{x}(k)\}$, and the observations are the malicious measurements $\mathbf{y}^a(1:k)=\{\mathbf{y}^a(1),\ldots,\mathbf{y}^a(k)\}$. In general, the attack vector can be modeled by a Gaussian mixture model (GMM), as GMM comprises a finite or infinite number of different Gaussian distributions that can describe different features of data. To simplify the problem and focus on the performance of the algorithm, we assume the attack vector is drawn from a single multi-variate Gaussian distribution with parameters $\boldsymbol{\mu}^a$ and V^a and the attack signals at each sensor are uncorrelated, so V^a is a block-wise diagonal matrix. Let $\hat{V} = diag(V_1+V_1^a,\ldots,V_n+V_n^a)$ and denote the model parameters by $\boldsymbol{\theta}=\{\boldsymbol{\mu}^a,\hat{V}\}$, the marginal likelihood of the model can be expressed by

$$f(\mathbf{y}^a(1{:}k))=\int\prod_{i=1}^{n} f(\mathbf{y}_i^a(1{:}k)|\mathbf{x}(1{:}k),\boldsymbol{\theta})f(\mathbf{x}(1{:}k))f(\boldsymbol{\theta})d\mathbf{x}d\boldsymbol{\theta}.$$

Denoting the Gaussian distribution by $N(\cdot)$, we have

$$f(\mathbf{y}_i^a(1:k)|\mathbf{x}(1:k),\boldsymbol{\theta}) = \prod_{j=1}^{k} N(H_i\cdot\mathbf{x}(j)+\boldsymbol{\mu}_i^a,\hat{V}_i),$$
$$f(\mathbf{x}(1:k)) = \prod_{j=1}^{k} N(A\cdot\mathbf{x}(j-1),Q).$$

Furthermore, we assume that $\boldsymbol{\mu}^a$ and $\hat{V}$ have conjugate priors, so the posterior probability distribution of $\boldsymbol{\theta}$ will be in the same family as the prior probability distribution, i.e.,

$$\boldsymbol{\mu}^a|(\boldsymbol{\mu}^0,\rho^0,\hat{V}) \sim N(\boldsymbol{\mu}^0,\hat{V}/\rho^0), \hat{V}^{-1}|(\gamma^0,B) \sim W(\gamma^0,B^{-1}/\gamma^0),$$

where $\boldsymbol{\mu}^0$, ρ^0, γ^0 and B are hyper-parameters and $W(\cdot)$ represents the Wishart distribution. Since the distributions $f(\mathbf{y}^a(1{:}k))$, $f(\mathbf{x}(1{:}k))$ and $f(\boldsymbol{\theta})$ all belong to the exponential family, following the theorem in [11], we can obtain the updating rules for the attack parameters $\tilde{\boldsymbol{\mu}}_i^a = \mathrm{E}[\boldsymbol{\mu}_i^a|\mathbf{y}(1:k)]$ and $\tilde{V}_i(k) = \mathrm{E}[\hat{V}_i|\mathbf{y}(1:k)]$ as shown in (1) and (2). For simplicity, we assume B is diagonal, that is, we assume the measurement noise is uncorrelated in each dimension.

$$\tilde{\boldsymbol{\mu}}_i^a(k) = \frac{\rho^0\boldsymbol{\mu}_i^0+\sum_{j=1}^{k}[\mathbf{y}_i(j)-H_i\cdot\hat{\mathbf{x}}_i(j)]}{\rho^0+k}, \quad (1)$$

$$\tilde{V}_i(k) = \frac{[\mathrm{diag}(\rho^0\boldsymbol{\mu}_i^0+\sum_{j=1}^{k}\mathbf{y}_i(j)-H_i\hat{\mathbf{x}}_i(j))]^2+\gamma^0 B_i}{\gamma^0+k}$$
$$+\frac{\rho^0\cdot[\mathrm{diag}(\boldsymbol{\mu}_i^0)]^2+\sum_{j=1}^{k}[\mathrm{diag}(\mathbf{y}_i(j)-H_i\hat{\mathbf{x}}_i(j))]^2}{\gamma^0+k}, \quad (2)$$

where $diag(\mathbf{v})$ represents a diagonal matrix whose diagonal entries are the components of the vector $\mathbf{v}$. Equations (1) and (2) require all data until time k, so they cannot be computed online. We rewrite them in a form using only current observed values, as described in Algorithm III. Algorithm III presents our DKF algorithm with variational Bayesian learning.

Algorithm III: DKF Algorithm with Bayesian Learning

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i = \boldsymbol{\mu}_0$, $\bar{V} = B$, $\bar{\boldsymbol{\mu}}^a = \boldsymbol{\mu}^0$, $\rho = \rho^0$, $\gamma = \gamma^0$
2. While new data exists do
3. Locally aggregate data and covariance matrices:
   $\forall\ j\in\mathcal{J}_i,\ \mathbf{u}_j = H_j^T\bar{V}_j^{-1}(\mathbf{y}_j^a-\bar{\boldsymbol{\mu}}_j^a),\ \mathbf{z}_i=\sum_{j\in\mathcal{J}_i}\mathbf{u}_j,$
   $U_j = H_j^T\bar{V}_j^{-1}H_j,\ S_i=\sum_{j\in\mathcal{J}_i}U_j$
4. Compute the Kalman-consensus estimate:
   $\hat{M}_i = (\bar{M}_i^{-1}+S_i)^{-1}$,
   $\hat{\mathbf{x}}_i = \bar{\mathbf{x}}_i + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i) + \epsilon\hat{M}_i\sum_{j\in\mathcal{N}_i}(\bar{\mathbf{x}}_j-\bar{\mathbf{x}}_i)$
5. Compute the attack parameter estimate:
   $\tilde{\boldsymbol{\mu}}_i^a = (\rho\bar{\boldsymbol{\mu}}_i^a+\mathbf{y}_i^a-H_i\hat{\mathbf{x}}_i)/(\rho+1)$,
   $\tilde{V}_i = (\gamma\bar{V}_i+\rho[\mathrm{diag}(\bar{\boldsymbol{\mu}}_i^a)]^2-(\rho+1)[\mathrm{diag}(\tilde{\boldsymbol{\mu}}_i^a)]^2+$
   $[\mathrm{diag}(\mathbf{y}_i-H_i\hat{\mathbf{x}}_i)]^2)/(\gamma+1)$
6. Update the state of the Kalman-consensus filter:
   $\bar{M}_i = A\hat{M}_iA^T+Q,\ \bar{\mathbf{x}}_i = A\hat{\mathbf{x}}_i,\ \rho=\rho+1,$
   $\gamma=\gamma+1,\ \bar{\boldsymbol{\mu}}_i^a=\tilde{\boldsymbol{\mu}}_i^a,\ \bar{V}_i=\tilde{V}_i$
7. end while

To evaluate the performance of Algorithm III, we use it to estimate the position of a moving object in R^2 that approximately moves in circles. The output matrix is $H_i = I_2$ and the state of the process dynamics is 2-dimensional corresponding to the continuous-time system $\dot{\mathbf{x}} = A_0\mathbf{x}+C_0\mathbf{w}$ with $A_0 = [0,-1;1,0]$ and $C_0 = 50I_2$. We use the discrete-time model with step-size $\zeta = 0.02$, i.e., $\mathbf{x}(k+1) = A\mathbf{x}(k)+C\mathbf{w}(k)$ with parameters $A = I_2+\zeta A_0+\zeta^2A_0^2/2+\zeta^3A_0^3/6$ and $C=\zeta C_0$. The sensor networks used in our experiments are comprised of 100 randomly located nodes. There are two main observations. First, the algorithm is very robust under attacks that have $\boldsymbol{\mu}^a = \mathbf{0}$. No matter what is the value of V^a and how many sensors are compromised, as long as there is one sensor not compromised, the network always gives accurate estimations. Second, the algorithm performs as *majority voting* when $\boldsymbol{\mu}^a \neq \mathbf{0}$, as we can see in Figure 1. The performance is very

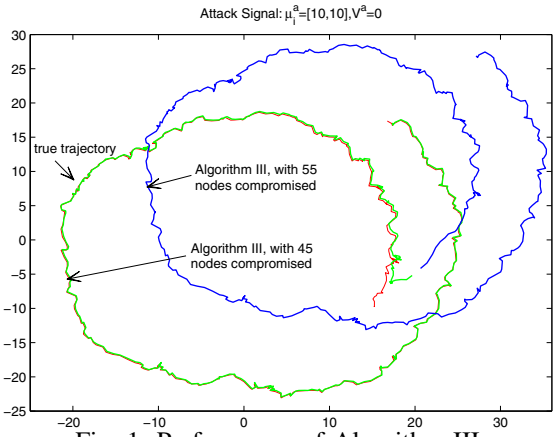

Fig. 1: Perfomrance of Algorithm III

good when 45 sensors are compromised, but degrades abruptly when 55 sensors are compromised. The attack parameters in Figure 1 are set as $\boldsymbol{\mu}_i^a = [10,10]$ and $V^a = 0$. The reason for the two observations is that in Algorithm III nodes use all

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

the data received for estimation and resolve the inconsistency among data by adjusting the estimated mean and covariance matrix of the measurement noise. When $\boldsymbol{\mu}^a = 0$, the algorithm handles the perturbation on the covariance matrix well since it knows that large noise covariance matrix indicates low accuracy. But when the perturbation is on the mean value, the algorithm cannot identify compromised nodes and only the majority voting rule can be used.

IV. Trust-aware Distributed Kalman Filtering

In this section, we propose a trust-aware scheme for DKF and show that it is robust under false data injection even when the majority of nodes are compromised. We introduce a subset of pre-trusted nodes into the network, which are special nodes that can be highly trusted, e.g., sensors equipped with tamper resistant hardwares. We show that as long as there is a path connecting every node in the network to some of the pre-trusted nodes, the attacker can not subvert the system.

a) Computational Model for Trust: To handle the uncertainty on the possible existence of malicious nodes, we assume each node is in one of two states: normal or malicious. We denote the state space by $\mathcal{S} = \{S_1, S_2\}$, where S_1 is the normal state and S_2 is the malicious state, and denote the true state of node i by $S(i)$. Then the local trust opinion of node i on node j is defined by the transition probability matrix:

$$P_{i,j}=\begin{bmatrix}\Pr(S(i)=S_1|S(j)=S_1) & \Pr(S(i)=S_1|S(j)=S_2)\\ \Pr(S(i)=S_2|S(j)=S_1) & \Pr(S(i)=S_2|S(j)=S_2)\end{bmatrix}.$$

$P_{i,j}$ is a column stochastic matrix, i.e., the sum of each column is equal to 1. The computation of $P_{i,j}$ depends on the particular application. As our first step for the design, we model $P_{i,j}$ by a logistic function that depends on the Euclidean distance between the measurements from two neighboring nodes. We assume neighboring nodes have the same observation matrix H_i, so the larger the distance between the measurements, the smaller the trust the nodes have for each other. More general assumptions on H_i can be made and may require different models for $P_{i,j}$; we will explore this issue in the future. Let i and j be neighbors that exchange measurements $\mathbf{y}_i^a(k)$ and $\mathbf{y}_j^a(k)$ at the kth step, and denote $d_{ij}(k) = ||\mathbf{y}_i^a(k) - \mathbf{y}_j^a(k)||_2$. We compute the conditional probability by

$$P_{ij}(1,1) = P_{i,j}(2,2) = f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k)) = \frac{2e^{-d_{ij}(k)/\alpha}}{1+e^{-d_{ij}(k)/\alpha}},$$

The reason we set $P_{ij}(1,1) = P_{i,j}(2,2)$ is that there is no prior information about the normal state and malicious state, so we treat the two cases as equally likely. The function $f(\mathbf{y}_i^a(k), \mathbf{y}_j^a(k))$ is actually a 'soft' step function, i.e., when $d_{ij}(k) = 0$, it equals to 1 and when $d_{ij}(k) = +\infty$, it equals to 0. The parameter α controls the slope and cutoff value of the function. To accumulate previous experience into the local trust opinion, we update the matrix $P_{i,j}(k)$ by

$$P_{i,j}^{(k)}=(1-\beta)\begin{bmatrix} f(\mathbf{y}_i^a(k),\mathbf{y}_j^a(k)) & 1-f(\mathbf{y}_i^a(k),\mathbf{y}_j^a(k))\\ 1-f(\mathbf{y}_i^a(k),\mathbf{y}_j^a(k)) & f(\mathbf{y}_i^a(k),\mathbf{y}_j^a(k))\end{bmatrix}+\beta P_{i,j}^{(k-1)}$$

where β is a time discount factor.

The global trust value for node i is defined to be a 2×1 vector $\mathbf{g}_i$, which represents the probability of node i in one of the two states, i.e.,

$$\mathbf{g}_i = [\Pr(S(i) = S_1), \Pr(S(i) = S_2)]^T.$$

Given node j's global trust value, node i's posterior probability in one of the two states can be computed by $P_{i,j} \cdot \mathbf{g}_j$. Assuming a uniform prior for node i's neighboring nodes, the estimate of the posterior probability of i can be written as

$$\mathbf{g}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} P_{i,j} \cdot \mathbf{g}_j.$$

Since the global trust values for the pre-trused nodes are known, we propagate the global trust from them to other nodes in the network. Denote the set of indices of the pre-trusted nodes by $\mathcal{T}$ and the set of indices of other nodes by $\mathcal{U}$. Without loss of generality, we assume the pre-trusted nodes are indexed from 1 to t. Let $\mathbf{g}_\mathcal{T} = [\mathbf{g}_1, \ldots, \mathbf{g}_t]^T$, $\mathbf{g}_\mathcal{U} = [\mathbf{g}_{t+1}, \ldots, \mathbf{g}_n]^T$, $D_{\mathcal{U}\mathcal{U}} = diag(|\mathcal{N}_{t+1}|, \ldots, |\mathcal{N}_n|)$, and $P_{\mathcal{U}\mathcal{U}}$ be the transition probability matrix from $\mathcal{U}$ to $\mathcal{U}$, i.e., $P_{\mathcal{U}\mathcal{U}}(i,j) = P_{t+i,t+j}$, $P_{\mathcal{U}\mathcal{T}}$ be the transition probability matrix from $\mathcal{U}$ to $\mathcal{T}$, i.e., $P_{\mathcal{U}\mathcal{T}}(i,j) = P_{t+i,j}$. Then the computation of the global trust values for nodes in $\mathcal{U}$ can be written in a matrix form, i.e.,

$$D_{\mathcal{U}\mathcal{U}} \cdot \mathbf{g}_\mathcal{U} = P_{\mathcal{U}\mathcal{U}} \cdot \mathbf{g}_\mathcal{U} + P_{\mathcal{U}\mathcal{T}} \cdot \mathbf{g}_\mathcal{T}. \quad (3)$$

Since $D_{\mathcal{U}\mathcal{U}}^{-1} P_{\mathcal{U}\mathcal{U}}$ is a sub stochastic matrix, it can be proved [12] that if for every node $i \in \mathcal{U}$, there exists a path that connects some node $j \in \mathcal{T}$ to i, then the spectral radius $\rho(D_{\mathcal{U}\mathcal{U}}^{-1} P_{\mathcal{U}\mathcal{U}})$ is less than 1 and the following Jacobi iteration converges to the unique solution of equation (3),

$$\forall i \in \mathcal{U},\ \mathbf{g}_i^{(k+1)} = \frac{1}{|\mathcal{N}_i|}\Big[\sum_{j \in \mathcal{U}, j \in \mathcal{N}_i} P_{i,j}\mathbf{g}_j^{(k)} + \sum_{l \in \mathcal{T}, l \in \mathcal{N}_i} P_{i,l}\mathbf{g}_l\Big]. \quad (4)$$

The convergence speed of equation (4) depends on the graph structure. More specifically, it is related to the second largest eigenvalue of the matrix $D_{\mathcal{U}\mathcal{U}}^{-1} P_{\mathcal{U}\mathcal{U}}$ [13]. Experimental results show that this computation converges very fast, usually in less than 10 iterations. The reason for the fast convergence is discussed in [13]. Figure 2 shows the convergence of equation (4) in several different networks. These networks are randomly generated with uniform sensor deployment. In each network, 10 randomly selected sensors are set as the pre-trusted nodes with global trust value $[0.95, 0.05]^T$.

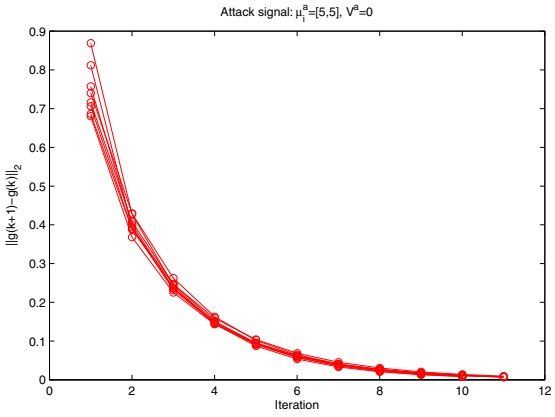

Fig. 2: Convergence speed of trust propagation

To handle the convergence of the distributed Kalman filtering and the trust propagation process together, we evaluate trust in a smaller time scale than the DKF. That is, we let the trust propagation algorithm achieve convergence between each interval of state updates in DKF. This is possible since equation (4) converges very fast. Moreover, since an abrupt change to a node's trust value does not occur frequently, we can expect that after the first trust evaluation, the following

978-1-4244-5637-6/10/$26.00 ©2010 IEEE

ones will converge much faster. Let $r_i = 1 \times \mathbf{g}_i(1) + 0 \times \mathbf{g}_i(2)$ be the trust weight of node i, which is the probability that node i is in the normal state, Figure 3 illustrates the performance of the trust evaluation algorithm on distinguishing malicious nodes from normal nodes. We can see that the trust weights

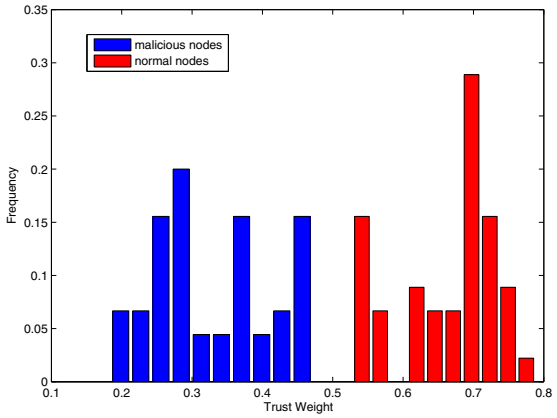

Fig. 3: Distribution of trust weights for nodes in the network

for the normal nodes are always larger than 0.5, while the malicious nodes always have trust weights less than 0.5.

b) Trust-aware DKF algorithm: After trust propagation, the DKF algorithm can weight the data sent by a node by its trust weight r_i. Algorithm IV presents the trust-aware DKF algorithm, where $d(i)$ is the degree of node i.

Algorithm IV: Trust-aware DKF Algorithm

1. Initialization: $\bar{M}_i = M_0$, $\bar{\mathbf{x}}_i = \boldsymbol{\mu}_0$
2. While new data exists do
3. Locally aggregate data and covariance matrices:
$\forall\ j \in \mathcal{J}_i,\ \mathbf{u}_j = H_j^T V_j^{-1} \mathbf{y}_j^a,\ U_j = H_j^T V_j^{-1} H_j,$
$\mathbf{z}_i = \frac{(d(i)+1)\sum_{j\in\mathcal{J}_i} r_j \mathbf{u}_j}{\sum_{j\in\mathcal{J}} r_j},\ S_i = \frac{(d(i)+1)\sum_{j\in\mathcal{J}_i} r_j U_j}{\sum_{j\in\mathcal{J}_i} r_j}$
4. Compute the Kalman-consensus estimate:
$\hat{M}_i = (\bar{M}_i^{-1} + S_i)^{-1},$
$\hat{\mathbf{x}}_i = \bar{\mathbf{x}}_i + \hat{M}_i(\mathbf{z}_i - S_i\bar{\mathbf{x}}_i) + \epsilon\hat{M}_i \frac{d(i)\sum_{j\in\mathcal{N}_i} r_j(\bar{\mathbf{x}}_j - \bar{\mathbf{x}}_i)}{\sum_{j\in\mathcal{N}_i} r_j}$
5. Update the state of the Kalman-consensus filter:
$\bar{M}_i = A\hat{M}_i A^T + Q,\ \bar{\mathbf{x}}_i = A\hat{\mathbf{x}}_i$
6. end while

In Algorithm IV, less trusted nodes have less impact on the algorithm. We can also isolate those less trusted nodes by ignoring data sent by them. In this way, the algorithm can achieve better performance. We name this algorithm of isolating less-trusted nodes as Algorithm V. Table I shows the performance of Algorithm III, IV and V when 60 randomly selected sensors are compromised. We measure the algorithm performance by the mean square error till time k, i.e., $e_i^{MSE} = \sqrt{(\sum_{l=1}^{k}(\hat{\mathbf{x}}_i(l) - \mathbf{x}(l))^2)/k}$, where $\hat{\mathbf{x}}_i(l)$ is the state estimate by sensor i at time l. In the experiment, we set $k = 400$. Since the sensors can approximately achieve consensus in DKF [9], the estimated state $\hat{x}_i(l)$ is approximately the same for each sensor i, and so is the value of e_i^{MSE}.

We can see that the trust-aware schemes perform better than the DKF algorithm with Bayesian learning, and Algorithm V performs better than Algorithm IV. The only drawback of Algorithm V is that the isolation of the less-trusted nodes may

TABLE I: Performance Comparison of Algorithm III, IV, V

μ_i^a, V_i^a	$[5,5], 0$	$[5,5], 2$	$[10,10], 0$	[10,10], 5
Algorithm III	4.7345	3.1572	9.8317	6.8935
Algorithm IV	1.2816	1.6131	3.3624	2.1507
Algorithm V	0.6724	0.5298	1.1997	0.8966

lead to a disconnected network. Also we observe that when $V_i^a \neq 0$, the algorithms may perform better than the case when $V_i^a = 0$, which means that V_i^a being non-zero in some sense reduces the impact of $\boldsymbol{\mu}_i^a$ on the estimated states.

V. Conclusions

We proposed two schemes for robust state estimation in distributed sensor networks against false data injection. The first scheme is based on variational Bayesian learning, which achieves good performance as a majority voting rule without causing extra communication overhead. The second scheme achieves performance better than majority voting by introducing a subset of pre-trusted nodes into the network. Experimental results demonstrate the effectiveness of our schemes. In future work, we will further investigate our trust-aware scheme, e.g., different models for the probability transition matrix $P_{i,j}$, the impact of the number of pre-trusted nodes and their positions on the performance of the trust-aware scheme.

Acknowledgement

Research supported by the Defense Advanced Research Projects Agency (DARPA) under award number 013641-001 for the Multi-Scale Systems Center (MuSyC), through the FRCP of SRC and DARPA, and by US Air Force Office of Scientific Research MURI award FA9550-09-1-0538.

References

[1] K. Pratik and H. Sajid, "Special issue on information fusion in distributed sensor networks," *Information Fusion*, 2008.
[2] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," in *Proceedings of IEEE INFOCOM*, 2004.
[3] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "Interleaved hop-by-hop authentication against false data injection attacks in sensor networks," *IEEE Transaction on Sensor Networks*, 2007.
[4] V. Shukla and D. Qiao, "Distinguishing data transience from false injection in sensor networks," in *SECON 2007*, 2007.
[5] Y. Zhang, J. Yang, and H. Yu, "Interleaved authentication for filtering false reports in multipath routing based sensor networks," in *Proceedings of IEEE IPDPS*, 2006.
[6] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," in *Proceedings of IEEE INFOCOM*, 2006.
[7] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure information aggregation in sensor networks," in *Proceedings of ACM Sensys.*, 2003.
[8] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003.
[9] R. Olfati-Saber, "Distributed Kalman filter with embedeed consensus filters," in *Proceedings of 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference (CDC-ECC 05)*, 2005.
[10] ——, "Distributed Kalman filtering for sensor networks," in *Proceedings of IEEE Conference on Decision and Control*, 2007.
[11] M. J. Beal and Z. Ghahramani, "The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures," *Bayesian Statistics*, 2003.
[12] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*. MIT Press, 2006.
[13] T. H. Haveliwala, S. D. Kamvar, and A. D. Kamvar, "The second eigenvalue of the Google matrix," 2003.

978-1-4244-5637-6/10/$26.00 ©2010 IEEE