IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 5, Ver. II (Sept - Oct. 2016), PP 01-07
www.iosrjournals.org

Detecting Malicious Nodes in Wireless Sensor Networks

[1]K.V.V.S. Pravallika, M.Tech ; [2]Mrs. S. Venkata Lakshmi, Asst.Prof., IT Dept.
G. Narayanamma Institute of Technology and Science

Abstract: *A typical wireless sensor network consists of several tiny and low-power sensors which use radio frequencies to perform distributed sensing tasks. Wireless sensor networks are used to detect the occurrence of events such as fires, intruders, or heart attacks, malicious data can be injected to create fake events, and thus trigger an undesired response, or to mask the occurrence of actual events. In this project we consider directly the scenario where an attacker gains full control of one or more sensors and can run arbitrary malware on them to fabricate new measurements and report them in place of the observed ones. Our base work only concentrated on the malicious data injection, we enhanced our base work with detecting the sink hole attack and avoiding the sink hole node to transfer the data.*
Keywords:*Malicious node, Detection, Malicious data.*

I. Introduction.

A wireless sensor network consists of sensor nodes capable of collecting information from the environment and communicating with each other via wireless transceivers. The collected data will be delivered to one or more sinks, generally via multi-hop communication. The sensor nodes are typically expected to operate with batteries and are often deployed to not-easily-accessible or hostile environment, sometimes in large quantities. It can be difficult or impossible to replace the batteries of the sensor nodes. On the other hand, the sink is typically rich in energy.

II. Design

There are three basic modules of design that are necessary for the detection of malicious node in a WSN. They are:
1) Estimation
2) Similarity check.
3) Characterization.

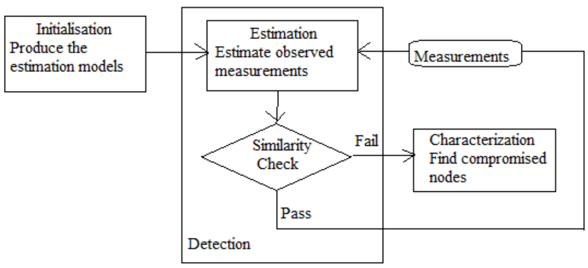

Fig.1. Outline of the system.

1) ESTIMATION

The estimation process consist of estimating the other nodes values, through which a trust based mechanism can be established between the nodes and the network to know which node has a possibility of being tampered with and the possibility of being a malicious node.

Consider a node Si, which releases a value of i. here, the value released by the node Si, has to be estimated by the other neighboring nodes, in order to create a basic trust within the network. This can be done by using two types of estimation processes.
a) Pair-estimation.
b) Aggregate estimation.

a) Pair Estimation

The pair estimation process is done by creating clusters of nodes within the network and selecting the cluster heads for the clusters.

The process of selecting the cluster head is to calculate the mean and mode of the estimated value in estimation. The measurements of two sensors are related, and in particular spatially correlated, because the sensed physical phenomena affect and propagate across the environment in which the sensors are placed. Ideally, the relationship could be characterized in a mathematically precise way, given by the laws of the physical phenomenon and its propagation.

Algorithm 1 Estimation models calculation
Input: O_i $i \in S$
Output: (a_{ij}, b_{ij}) $\forall i \neq j$
1: {Initialization: align the measurements with the inter-sensor delay}
2: **for all** $i \in S$ **do**
3: **for all** $j \in N(i)$ **do** {$N(i)$ indicates i's neighbors}
4: $a_{ij} = cov(O_i,O_j)/var(O_i)$
5: $b_{ij} = E[O_i] - a_{ij}E[O_j]$
6: store (a_{ij}, b_{ij})
7: **end for**
8: **end for**

b) Estimate Aggregation

For every new measurement collected by a sensor, multiplepairwise estimates are calculated through the estimation models.At this point we aggregate them into a final estimate $\hat{O}_i$that approximates $O_i$and allows us to detect the presence ofmalicious data injections. To achieve this, $\hat{O}_i$must aggregate estimates in a way that is both accurate andminimally corruptedby malicious estimates. In particular, the second requirementdemands us to not trust the relationships between differentestimates. Indeed, different estimates for the same measurementsshare some mutual information, or in other words theinformation brought in by an estimate is reduced by knowledge of other estimates. Nevertheless, such property holds only in theabsence of malicious interference. With respect to malicious data injections instead, even two estimates that are expectedto be perfectly correlated bring in independent information,since we assume independent probabilities of compromise fordifferent nodes. For this reason, our weighting scheme does notconsider inter-estimate correlation.

Two candidates to aggregate pairwise estimates are *weightedmean* and the *weighted median*: both take as input a set ofestimates and their prior weights and return an aggregatedvalue. The *weighted mean* can achieve a smaller error thanthose of the single estimates. However, it is highly sensitiveto compromise, since the final result is proportional to the input values: even one compromised (outlier) estimate canintroduce an arbitrary deviation in the result. In contrast, the*weighted median* is more resistant to compromise. It first sorts the values ascendingly,then arranges the weights withthe same order, transforms them into substrings with a length proportional to the weight and picks the element at the half-lengthof the resulting string. Its drawback is that by pickingone among all estimates, the error cannot be reduced further.

Since there is a trade-off between accuracy and compromiseresistance, we propose to combine the two operators with thefollowing heuristic: first, the weighted median operator is applied; then the weighted mean is calculated with new weights,the *posterior weights* (w^+_{ij}), obtained as the prior weights times a function which penalizes values distant from the result ofthe first step. Such function is the complementary cumulativedistribution function of the estimation error, where the latter iscalculated as the difference between the pairwise estimates and the result of the weighted median.

$$p_{ij}(\hat{O}', \hat{O}_{ij}) = P(|\epsilon_{ij}| > |\hat{O}'_i - \hat{O}_{ij}|) = 1 - erf\frac{|\hat{O}'_i - \hat{O}_{ij}|}{\sqrt{2}\,std(\epsilon_{ij})} \qquad (2)$$

Where $erf$is the error function and $std(\epsilon_{ij})$ is the residualstandard deviation, calculated together with the respectiveestimation model. The overall procedure is detailed inAlgorithm 2 below, where $\hat{\boldsymbol{O}}_{\boldsymbol{i}\boldsymbol{N(i)}}$ are the estimates for i'sobserved measurement from its neighbors and $\boldsymbol{w^-_{iN(i)}}$ are theirrespective prior weights.

Algorithm 2 Calculation of the aggregated estimation
Input: $\boldsymbol{w^-_{iN(i)}}$, $\hat{\boldsymbol{O}}_{\boldsymbol{iN(i)}}$
Output: $\hat{O}_i$
1: $\hat{O}'_i$= weightedMedian$(\boldsymbol{w^-_{iN(i)}}, \hat{\boldsymbol{O}}_{\boldsymbol{iN(i)}})$
2: **for all** $j \in N(i)$ **do** {Calculate the posterior weights}
3: $w^+_{ij} = w^-_{ij} \cdot p_{ij}(\hat{O}', \hat{O}_{ij})$
4: $\boldsymbol{w^+_{iN(i)}}$.append(w^+_{ij})

5: **end for**
6: $w+iN(i)=\frac{w+iN(i)}{\sum j\in N(i)\ w+ij}$
7: $\hat{O}i$= weightedMean$(w+iN(i), \hat{O}iN(i))$
8: **return** $\hat{O}i$

To control the aggregation result, an attacker must ensure that the weighted median is one of the compromised estimates andthus that the sum of the weights of compromised estimates is >0.5. This condition enables non-detectable injections into asingle sensor but is not sufficient to keep the attack undetected.The attacker also needs to control the estimations for the othercompromised sensors. The total number of sensors neededto keep all compromised sensors undetected depends on thestrength of the pairwise correlations. Instead, the number ofsensors needed to mask or elicit an event depends on theevent detection criterion. Our empirical evaluations show thatalthough a few sensors are generally required to subvert theevent detection, a substantial additional number of sensors isrequired to avoid detection.

2) SIMILARITY CHECK

From the estimate aggregation step, each reported measurementSi has an estimate $\hat{O}i$of the observed value. To detectdata injections in Si, we compare the two using a *similaritymetric* that must be consistent with the event detection criterion. So, two signals that are similar according to the metric mustalso have similar effects on the event detection and vice-versa. Here we propose two tests that capture the characteristicsof most event detection criteria.

a) Magnitude Test
b) Shape Test

a) ***Magnitude Test***-This test verifies that reported measurements are close in magnitude to their estimates.
b) ***Shape Test***-This test verifies that the estimate and reportedsignal have a similar shape. The choice of the most appropriatetest, or a combination of the two should be made at design timebased on the event detection criterion.

a) Similarity Test 1: Magnitude

In some WSNs, events are triggered when measurements are higher or lower than a reference value. For example, firealarms trigger when the temperature is above a threshold. An attacker must therefore inject measurements, which differ in magnitude with the observed ones. In such cases we use $Mi=(\hat{O}i- Si)$—the difference between the reported measurementand its estimate—to build a *magnitude test*, which checks thatthe difference is small enough.We assumed that the regression *residual*, i.e. the error betweena value and its estimate, is zero-mean and normallydistributed. Even if $\hat{O}i$is the result of the aggregation, the error $\epsilon i = (\hat{O}i- Oi)$ can still be assumedto be normally distributed. Indeed, our aggregate is a weightedmean of pairwise estimates, so it equals the true value plusthe weighted mean of the pairwise residuals as shown below,where ϵijdenotes the residual in the regression of sensor i'smeasurement based on sensor j's.

$$\hat{O}i=\sum j\in N(i)w+\sum ij\hat{O}ij=\sum j\in N(i)w+ij(Oi+\epsilon ij)=Oi+\sum j\in N(i)w+ij\epsilon ij \quad (3)$$

Assuming that neighbors have independent residuals (e.g.,because of independent noise), _i is a linear combination ofindependent normally distributed samples, and is thus normally distributed too. Its mean is still zero, and its variance is:

$$var(\epsilon i) =\sum j\in N(i)w+ij2var(\epsilon ij) \quad (4)$$

This equation has an important characteristic: the variance ofthe estimate is a combination of the variances given by eachneighbor. Therefore, if a sensor joins or leaves the network,it is sufficient that all its new/old neighbors re-compute thevariance instead of learning a new one. Since $\epsilon istd(\epsilon i)$ follows the standard normal distribution, also $\epsilon Mi= Mistd(\epsilon i)$ does when the measurements are genuine.We refer to ϵMias the *magnitudedeviation*.

Increasing the threshold reduces false positives, but decreasesthe detection rate. However, in event detection WSNsthe false positives can be partly reduced without losing thedetection rate by elaborating magnitude deviations in the sameway as the event detection criterion elaborates the measurements.Consecutive magnitude deviations are unlikelyto cause genuine anomalies with a long duration, unless thereis a permanent fault that the fault-detection module shoulddetect. Anomalies due to compromise, instead, have a longer duration as the attacker aims to subvert the event detectionresult. The final step consists of comparing the elaboratedmagnitude deviation to the *threshold TM*.

b) Similarity Test 2: Shape

Some event detection algorithms trigger based on changes inthe time evolution of measurements such as changes in trend orof frequency. These are characteristics of the shape of the signal rather than its magnitude.

A metric that measures similarity in the shapes of two signalsis the Pearson correlation coefficient. Since our purpose is to check the shape of the measurements used for event detection,we calculate this coefficient within a moving time window ofsizeWE: the event detection time window. Calculating Pearsoncorrelation for all sensor pairs in a neighborhood would have a computational complexity of $O(N2NWE)$, with NN beingthe neighborhood size. In contrast, we evaluate the Pearsoncorrelation coefficient of a sensor's measurements with itsestimates, achieving a complexity of $O(N2N+ WENN)$. Indeed,we compute the coefficient $RSi,\hat{}Oi$, between WE consecutivevalues of Si and $\hat{}Oi$, and compare it against the distributionof $ROi,\hat{}Oi$. Specifically, if the coefficient is below the median, we check if at least 100 − CR% samples are expected to be solow by testing

$ROi,\hat{}Oi=ROi,\hat{}O\, i-MED(ROi,\hat{}O\, i)DRi>1$, where DRiistheCR-th percentile of $ROi,\hat{}Oi$.

To eliminate the need for the distribution of $ROi,\hat{}Oi$,the quantities $MED(ROi,\hat{}Oi)$ and$DRi$are approximated with$MED(ROi,\hat{}Oi)$ and $\hat{}DRi$respectively. These are calculated witha heuristic described in Algorithm 3 for a generic sensor i.We find the best neighbor $j*$, for which the median Pearsoncorrelation coefficient is maximum. Then we approximate$MED(ROi,\hat{}Oi)$ with its median and $DRi$with its respectivedistance to the CR-th percentile. We characterize the samples below the median since the injected measurementsare supposed to have a low correlation with the real values.

Algorithm 3 Characterization of the distribution of RSi,Oi
Input: $Rij:j\in N(i)(r), CR$
Output: $MED(ROi,\hat{}Oi), \hat{}DRi$
for all $j \in N(i)$ **do**
$MEDRij= MED(Rij(r))$
$\boldsymbol{MEDRij}$.append$(MEDRij)$
$rLOW= \{r : r<MEDRij\}$
$DRij=$ percentile$(MEDRijRij(rLOW),CR)$
$\boldsymbol{DRij}$.append$(DRij)$
end for
$j* =$ argmax$j\in N(i)(\boldsymbol{MEDRij})$
$MED(ROi,\hat{}Oi) = \boldsymbol{MEDRij}[j*]$
$\hat{}DRi= \boldsymbol{DRij}[j*]$
return $(\epsilon MED(ROi,\hat{}Oi), \hat{}DRi)$

In the absence of the distributions $Rij\in N(i)(r)$, we estimate$MEDRij$and $DRij$on historical data. For genuine sensors $Si = Oi$, then$_R\ Si,\hat{}Oi\leq 1$ for CR% genuinesamples. We thus define$_R\ Si,\hat{}Oi$as the *shape deviation* andcalculate CR as the lowest value that achieves a reasonablefalse alarm rate. The false positives due to short term anomaliescan be reduced in a similar way to that used in the magnitudetest i.e., by computing the median of WSmconsecutive correlationcoefficients calculated on overlapping time windows. WSmshould never exceed WE, otherwise the information from disjoint time windows would be merged.

3) CHARACTERIZATION

When the similarity check fails for a sensor, the sensormay have been compromised by malicious data. However, insome cases the similarity check could also fail on genuinesensors, because the wrong modality was chosen (e.g., a non-eventmodality rather than an event modality) or because the estimation was disturbed by compromised sensors.

The latter occurs when several nearby sensors collude inproviding malicious estimates. However, to bias the estimates for genuine sensors by a certain quantity and increase theirdeviation, compromised nodes typically need to inject measurementsthat have even larger deviations (if they do not need this, colluding sensors have probably enough influenceover the measurements to remain undetected). Therefore, ourcharacterization step consists in removing the sensors with thehighest deviation, one by one, and re-computing the similaritycheck on the remaining sensors in the neighborhood. Eachtime we remove a sensor, which we presume compromised,the genuine sensors gain in consistency with their estimatewhereas colluding sensors lose the benefits of the removed sensor's estimates. The procedure stops when all the remainingsensors pass the similarity check. The overall characterization algorithm is shown in Algorithm 4, where SCheck is thesimilarity check and Di is the generic deviation (coming from the magnitude/shape tests) calculated for the similarity check.

Algorithm 4 Characterization algorithm
Input: $Di\ \forall i \in S$
Output: compromisedSet
1: compromisedSet= {}
2: residualSet= S
3:**while**ORi∈residualSet*(*SCheck*(Di)fails)***do**
4: $s*$ = argmax$i$∈*residualSetDi*
5: compromisedSet.append*(s∗)*
6: residualSet.remove*(s∗)*
7: **for all** $j \in S$:$s* \in N(j)$ **do**
8: $N(j) = N(j) \setminus s*$
9: re-compute Dj
10: **end for**
11: **end while**
12: **return** compromisedSet

Another factor to consider when the similarity check failsis the *modality assumption* (Section IV-A1). When differentmodalities are used in event conditions and non-event conditions,there is some uncertainty about which modality touse because malicious data may have compromised the eventdetection output. In this case, the wrong estimation model may be used and genuine sensors may fail it. Our solutionis to run Algorithm 4 in both modalities when the similaritycheck fails and then choose the modality in which the smallestcompromised set is returned. It is reasonable to choose thecorrect modality based on a majority approach, as the attackcosts increase with the number of measurements that need to be controlled. Note that this is different from event detectionwith majority voting, since the measurements are not requiredto trigger in majority, but to reflect event propagation and showgraceful transitions in their measurements.

III. Project Analysis

The calculations rely on solid raw measurements of the nodes which are trustworthy. These calculations are sent to the base station which has a global view on and can deal with collusion of compromised nodes. This is also the case with aggregate and estimate nodes that have a defect in their measurements. For the WSN nodes that do not act as aggregators, the estimation-based framework adds no overhead, because no additional software is run on the sensor nodes to manage votes or trust values like in high integrity sensor nodes. For the base station and aggregators, the most computationally expensive operation of our approach is the calculation of the estimation models. When this operation is done one-off, powerful devices may be used offline, but when this is not possible, for instance because there is not enough historical data, the models need to be estimated in real time. In this case using external devices may be infeasible, and an efficient calculation is required to estimate the models with the sensor nodes.

IV. Results Discussion

Our current approach has shown to detect malicious interference also with sophisticated attacks, based on injection of credible measurements. Based on our characterization algorithm, we are able to detect correctly the set of compromised sensors when the number of genuine sensors is low compared to the expected correlation.

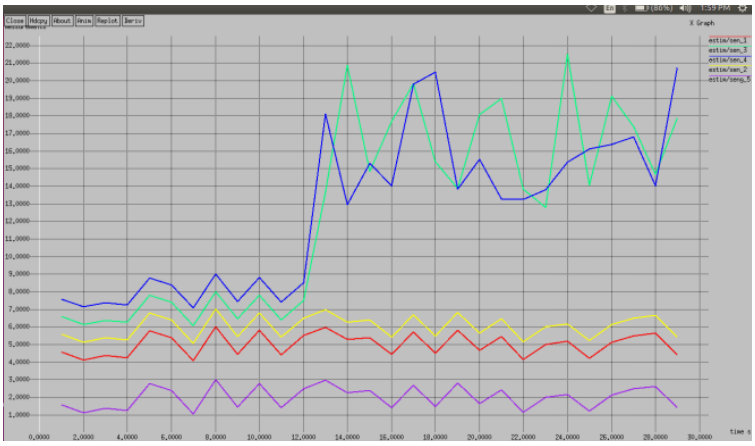

Fig.2. Estimation Graph

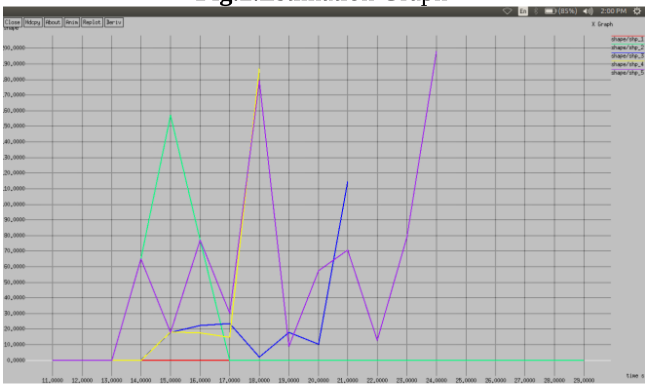

Fig.3. Shape Graph

The number of compromised sensors that can be tolerated is correlation-dependent. In one of our experiments attacks could be detected whenever fewer than 88% sensors were compromised. Voting-based frameworks instead, cannot tolerate more than 50% compromised sensors and, when our algorithm tolerated less than 50% compromised sensors, majority voting tolerated a substantially lower percentage. The reason behind this result is that the correlation between the sensors used in the experiments is not high enough to guarantee correct votes from all the genuine sensors and votes become inaccurate.

We simulate the attacks by injecting measurements describing normal circumstances but thatsubvert the event detection result, i.e., elicit a non-existent event, or mask a real event. In some cases, the attacker may need to inject measurements substantially different from the observed ones, but this will not be easily noticeable because the data describes wrong but still normal circumstances.

VII. Conclusion And Future Work

In this paper we have focused on detecting malicious data injections in event detection WSNs, in particular when collusion between compromised sensors occurs. We have proposed an algorithm that can be customized and used in different applications, and for different kinds of events.

Addressing this challenge has exposed several trade-offs in the design of the algorithm. Firstly, resistance to collusionrequires to compare measurements over a broader set of sensors and thus introduces additional complexity and computational cost. This trade-off is particularly visible in the selection of neighborhoods, which becomes a simple ranking-based choice when using our pairwise estimation models. Another trade-off arises when merging information with potentially malicious sources. While information coming from genuine sensors increases the estimates accuracy, it is important to select only information that appears reliable. Colluding sensors should not be allowed to compensate for each other in the detection metric whilst still injecting malicious data. This requires the use of pairwise comparisons and an aggregation operator that is accurate in the presence of genuine measurements as well as resistant to malicious data.

References

[1]. A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Mag. Commun.*, vol. 47, no. 6, pp. 53–57, Jun. 2004.

[2]. A. B. Sharma, L. Golubchik, R. Govindan, "Sensor faults: Detectionmethods and prevalence in real-world datasets," *Trans. Sensor Netw.*,vol. 6, no. 3, pp. 23–61, 2010.

[3]. B. Sun, X. Shan, K. Wu, and Y. Xiao, "Anomaly detection based securein-network aggregation for wireless sensor networks," *Syst. J.*, vol. 7, no. 1, pp. 13–25, Mar. 2013.
[4]. B. Przydatek, D. X. Song, and A. Perrig, "SIA: Secure information aggregationin sensor networks," in *Proc. SenSys*, 2003, pp. 255–265.
[5]. C. Otto, A. Milenkovi´c, C. Sanders, and E. Joranov, "System architecture of a wireless body area sensor network for ubiquitous health monitoring," *J. Mobile Multimedia*, vol. 1, no. 4, pp. 307–326, Jan. 2005.
[6]. C. V. Hinds, "Efficient detection of compromised nodes in a wirelesssensor network," in *Proc. SpringSim*, 2009, Art ID. 95.
[7]. C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *AdHocNetw.*, vol. 1, no. 2/3, pp. 293–315, 2003.
[8]. G.Werner-Allen *et al.*, "Deploying a wireless sensor network on an active volcano," *Internet Comput.*, vol. 10, no. 2, pp. 18–25, Mar./Apr. 2006.
[9]. M. Mathews, M. Song, S. Shetty, and R. McKenzie, "Detecting compromisednodes in wireless sensor networks," in *Proc. SNPD*, 2007, vol. 1, pp. 273–278.
[10]. M. K. Khan and K. Alghathbar, "Cryptanalysis and security improvements of 'two-factor user authentication in wireless sensor networks'," *Sensors*, vol. 10, pp. 2450–2459, 2010.
[11]. M. Raya, P. Papadimitratos, V. D. Gligor, and J.-P. Hubaux, "On datacentrictrust establishment in ephemeral ad hoc networks," in *Proc. 27thIEEE INFOCOM*, 2008, pp. 1–11.
[12]. Q. Zhang, T. Yu, and P. Ning, "A framework for identifying compromisednodes in wireless sensor networks," *Trans. Inf. Syst. Secur.*, vol. 11, no. 3,pp. 1–37, Mar. 2008.
[13]. S. Roy, M. Conti, S. Setia, and S. Jajodia, "Secure data aggregationin wireless sensor networks: Filtering out the attacker's impact," *IEEETrans. Inf. Forensics Security*, 2014, pp. 681–694.
[14]. T. He *et al.*, "Energy-efficient surveillance system using wireless sensor networks," in *Proc. MobiSys*, 2004, pp. 270–283.
[15]. V. Chatzigiannakis and S. Papavassiliou, "Diagnosing anomalies andidentifying faulty nodes in sensor networks," *IEEE Sensors J.*, vol. 7,no. 5, pp. 637–645, May 2007.
[16]. W. Zhang, S. K. Das, and Y. Yonghe, "A trust based framework for securedata aggregation in wireless sensor networks," in *Proc. 3rd Annu. IEEESECON*, 2006, pp. 60–69.
[17]. Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks againststate estimation in electric power grids," *Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 21–32, May 2011.