A Two-Phase Security Algorithm for Hierarchical Sensor Networks

Jingjun Zhao
Department of Computer Science
North Dakota State University
Fargo, ND, USA
jingjun.zhao@my.ndsu.edu

Kendall E. Nygard
Department of Computer Science
North Dakota State University
Fargo, ND, USA
Kendall.Nygard@my.ndsu.edu

***Abstract* – We present a two-phase security system designed for hierarchical wireless sensor networks and show how it can be used to detect Denial-of-Service attacks and track harmful intruders. This type of energy-exhaustion attacks can break the connection of a sensor network and decreases its lifetime. First, we apply a Dendritic Cell Algorithm inspired by danger theory to actively detect attacked sensors that are implementing battery exhaustion attack to neighbors. Second, the system passively analyzes the tracks of the moving harmful intruders. The tracking information is used by the base station to more efficiently monitor and control the network. We adopt Markov Chain Monte Carlo methods to track the intruders and a Tabu Search technique to accelerate the searching of the final intruder tracks. The simulation results demonstrate good performance of this corrective and preventive security mechanism on detecting the malfunction sensors and tracking the intruders.**

Keywords-Wireless Sensor Networks; Dendritic Cells Algorithm; Markov Chain Monte Carlo; Tabu Search

I. INTRODUCTION

Improved wireless communication and electronics promote the development of low-power, low-cost and multifunctional sensor nodes [11]. Large numbers of such nodes can be deployed in a wireless sensor network (WSN) to sense and report data of importance. Common application areas include hospitals, homes, battle fields, and transportation systems. Security is of high importance in most WSNs. The sensors deployed typically run on batteries with limited power and computation ability. The communication channels can be unreliable and unattended operations result in vulnerability to attacks and sensor failures such as wormhole [22], denial-of-service or sinkhole attacks [23]. Traditional cryptographic security algorithms assume that all nodes are cooperative and trustworthy [12]. This assumption is not satisfied in most real-world WSN applications so that traditional security approaches may not apply.

In this paper, we focus on detecting Denial-of-Service (DoS) attacks aimed at multiplying the rate of battery exhaustion in sensors. Harmful intruders maliciously consume sensors energy by jamming a portion of a network. Local disconnection may cause the cluster heads and the base station failure of receiving correct and complete sensing data. This type of attacks is very harmful to highly critical and sensitive applications.

We adopt Artificial Immune System (AIS) approach and multiple-target tracking techniques to detect security threats in WSNs. This proactive approach of detecting malfunction sensors and harmful intruders is more effective on increasing the service lifetime of a given sensor network than passive methods that only detect dead sensors. AIS is a problem-solving methodology inspired by how biological immune systems in mammals detect pathogens and destroy them before they cause harm to the body. In real-world WSNs it is difficult to know how many invaders of different types are present. But knowing the distribution and the tracks of intruders is very helpful information for a Base Station (BS) to have for defending the network. In our work, we utilize a Multi-target tracking technique to track mobile harmful intruders. A track is a path in time-space traveled by a target [3]. The data association problem is to identify the tracks of targets from noisy observations of target positions at known points in time.

The main contribution of our work is the development of a two-phase security mechanism for WSNs with hierarchical structure by combining a Dendritic Cell Algorithm (DCA) algorithm with a multiple-target tracking algorithm and a Tabu Search technique. The two-level hierarchical sensor network that we use for our experiments has a static backbone of sparsely placed high-end sensors nodes called Cluster Heads (CH). The low-end sensor nodes belong to different clusters based on their physical position. This hierarchical structure is well-suited for large scale sensor networks and is understood to be energy preserving [10]. The DCA algorithm works on the low-end nodes and identifies malfunctioning neighbors that have been attacked by harmful intruders. The primary malfunctions include packet change, fake packet and energy-exhaustion. The ability to defend against these basic but widely existing types of attacks in a WNS makes the algorithm a good fit in practice. The multiple-target tracking algorithm implementing on Cluster Heads is used to track the mobile harmful intruders. Information about the distribution and the trajectory of harmful intruders is used by the Base Station (BS) to assess and evaluate current network defense capability. The simulation results show that the immune-inspired algorithm can effectively detect the low-end sensor nodes that have been attacked and accurately track the mobile intruders.

Figure 1 shows the two-phase security architecture. Each low-end sensor detects malfunctioning sensor nodes and reports their positions to its cluster head. When a low-end sensor receives a message it utilizes the DCA algorithm to identify whether or not the message is abnormal. Messages are reported to head node as normal or abnormal. Before implementing the multi-target tracking algorithm, a cluster head performs a K-Neighbor Query Algorithm (KNQA) to assess the credibility of received abnormal observations. This algorithm improves the accuracy of reported abnormal observation. The tracking results are ultimately sent to the base station.

 ISBN: 978-1-61208-154-0

The rest of the paper is organized as follows. Section II discusses the related works. We detail the Dendritic Cell Algorithm in Section III. In Section IV, we present the query algorithm based on the K-Nearest Neighbor (KNN) technique. In Section V, we present the MCMC algorithm for multiple-target tracking. The Tabu Search technique used to identify the tracks of the targets in a WSN is represented in Section VI. We simulate the proposed algorithms and compare the performance with the MCMCDA algorithm in Section VII. Finally, the conclusion is given in Section VIII.

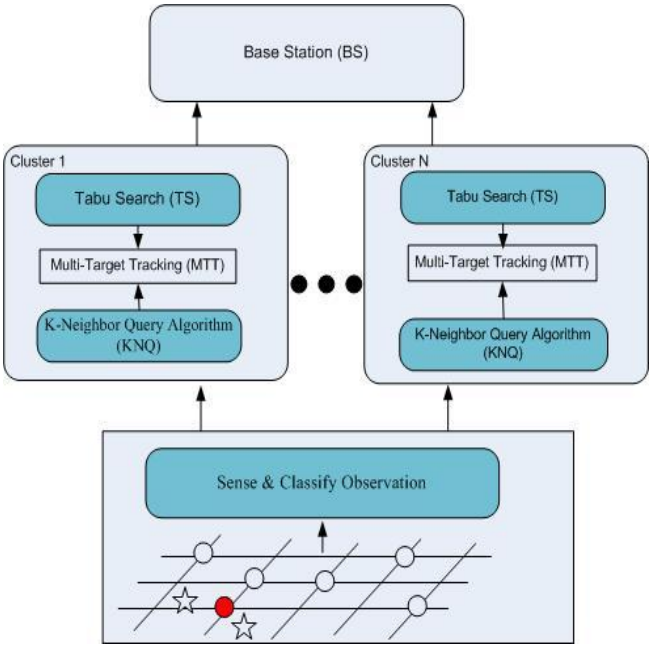


Figure 1. The two-phase security architecture (hollow circles represent normal sensors; the solid circle represents a malfunctioning sensor; and the stars represent mobile intruders)

II. RELATED WORKS

A particular class of AIS methodologies called a Negative Selection Algorithms (NSA) has been applied to anomaly detection problems [14]. Inspired by immunology, the approach uses a learning phase to construct detectors that can identify and dispatch invaders, but are not harmful to the organism itself. A fundamental issue in a NSA is maintaining distinguishing units from the host (self units) from the invaders (non-self units).[13]. Following another type of AIS, the work in [15] advanced the Danger Theory (DT) approach to intrusion detection. In Danger Theory, the central idea is that the immune system detects and responds to damage to the host, rather than upfront discrimination between self and non-self units. Dendritic cells play a central role in Danger theory. Work by Nauman and Muddassar [16] established a security system based on the behaviors of Dendritic Cells. The work reported in [17] includes detailed rules for a Dendritic Cell Algorithm (DCA) for analyzing abnormal signals. These DCAs are more flexible at detecting misbehaviors than NSAs, but do require a monitoring period to identify an intruder, resulting in inefficiency in situations with moving intruders.

Multiple-target tracking is a competent technique on tracing moving intruders in WSNs. In [3], the authors proposed a Markov Chain Monte Carlo Data Association (MCMCDA) algorithm for tracking a variable number of targets in real-time. It was established that MCMCDA is computationally efficient compared to multiple hypothesis tracker (MHT) [18], which is the prominent methodology for solving the data association problem, and outperforms MHT when there are large number of targets. MCMCDA partitions the observations into groups corresponding to candidate tracks. The collection of different partitions forms the state space for the MCMC method that searches for the most likely partitioning into intruder tracks. To make the computations of the proposal distribution easier, the authors include two additional assumptions: (1) the maximal directional speed of any target is less than $\bar{v}$; and (2) the number of consecutive missing observations of any track is less than $\bar{d}$.

Convergence rate is an important criterion on assessment of MCMC algorithm ability [20]. Although the MCMCDA makes two assumptions to decrease the size of state space, it doesn't have obvious improvement when the state space is very large. To accelerate the search of the final intruder tracks, we design two ways to improve the convergence rate of the Markov Chains used in the MCMC approach. Firstly, we further decrease the size of state space by classifying observations and predicting the moving area of an intruder. Secondly, we apply a proposed Ejection Chain algorithm to the optimal solution searching process. This approach partitions an optimization problem into relatively independent sub-optimization problems, and accelerates the optimization process. Comparing with a full MCMCDA, our algorithm needs fewer samples to reach an optimal solution.

III. THE DENDRITIC CELL APPROACH

Our previous work [5] proposed a Dendritic Cell inspired algorithm. This algorithm consists of three sub-functions: checking for dangerous messages, abnormal messages and harmful intruders. Identifying a harmful intruder is a process of changes among immature state, semi-mature state and mature state. When a sensor node receives a new message, the algorithm firstly checks a saved Harmful Intruder (HI) list. If the sender of this message is an identified harmful intruder with mature state, the message is considered to be a dangerous message. If the sender is an identified suspicious intruder with semi-mature state and exists in a Semi-Harmful Intruder (SHI) list, the algorithm implements the function of identifying harmful intruder to decide whether this sender is dangerous enough to be considered as a harmful intruder. Finally, the algorithm implements the function of checking for abnormal message. The sender is saved in the SHI list and its state changed from immature to semi-mature if the message is abnormal. This algorithm only consumes limited energy of a sensor node because of the simple calculation in each sub-function. The experiment results showed that this algorithm is capable of detecting static harmful nodes.

IV. K-NEIGHBOR QUERY ALGORITHM

In [1], the authors proposed the KNN Perimeter Tree (KPT) algorithm for dynamically processing KNN queries in location-

 ISBN: 978-1-61208-154-0

aware sensor networks. The algorithm KPT first adopts GPSR, which is a geographical routing algorithm [2], to find the Home Node (HN) that is the nearest sensor to the Query Point (QP). As by-products, the perimeter nodes around the QP are also determined when the HN is found. A circular boundary including at least K sensor nodes is determined. At each hop around the perimeter, the midpoint on the line between perimeter nodes is computed, and by plotting a line from the QP through the midpoint to the circular boundary the sub-tree boundaries are determined. In each boundary, a spanning tree is constructed, shown in Figure 2. Each node on the tree reports its position and ID to its parent node. Finally, all sensor information in the circular boundary is reported to the HN. The HN sorts these nodes by their distance to the QP to find the K nearest neighbors.

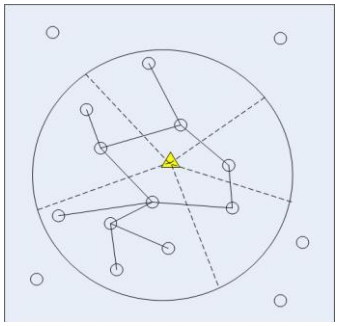


Figure 2. KNN Perimeter Tree

The perimeter boundaries can keep the spanning trees balanced and thus reduce the overall query latency. It is still possible that the K nearest neighbors are all or mostly in one spanning tree. Only querying some sensor nodes densely distributed in a small area around the QP cannot be trusted. In a given spanning tree, each node sends packages to its parent node instead of broadcasting. This reduces the number of totally transmitted message. But if a parent node in a spanning tree close to the root node has failed (e.g., because of energy exhaustion), this spanning tree will be useless for finding the K nearest neighbors.

To avoid these disadvantages and make the KNN query algorithm more efficient for detecting harmful intruders, we design the K- Neighbor Query Algorithm (KNQA) for finding K querying neighbors. Instead of constructing spanning trees in each sub-boundary, the KNQA finds K querying sensor nodes around a suspect harmful intruder, and these querying nodes tend to be closely and sparsely distributed around the suspect intruder.

Figure 3 shows an example of identifying 10 querying sensor nodes around a suspect harmful intruder. The first group querying nodes are those closest to the suspect node. We first, find the closest node to the suspect intruder, and then use a Right-Hand Rule to establish a perimeter [7], sequence of edges. The left querying nodes are searched clockwise, checking each boundary which has $\theta \geq 10^{\circ}$. In each boundary, we find the nearest node to the suspect intruder, and plot a line from the suspect intruder through the newly found node to the circular boundary. The partition of boundary continues until K nodes have been found or no boundary has $\theta \geq 10^{\circ}$.

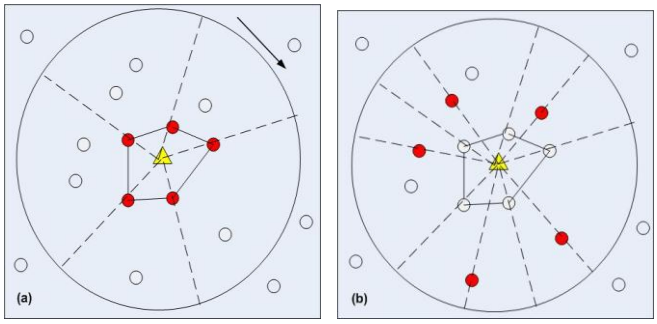


Figure 3. Querying K sensor nodes around a suspect intruder (K = 10) (a) the first 5 sensor nodes; (b) the second 5 sensor nodes

In [6], the authors described an Optimal Threshold Decision Scheme. According to the results in [6], the best policy for each node is to accept its own sensor reading if and only if at least half of its neighbors have the same reading. In this paper, we adopt this policy for deciding a harmful intruder.

V. MULTIPLE-TARGET TRACKING ALGORITHM

A. Markov Chain Monte Carlo

1) Problem formulation

In [3], the authors designed the MCMC data association (MCMCDA) algorithm for tracking an unknown number of targets that appear and disappear in the surveillance region during a surveillance period of time. The Markov chain Monte Carlo data association algorithm can initiate and terminate tracks autonomously and is robust to a high level of false alarms and missing measurements, a common problem in sensor networks [8]. During a surveillance period T, K targets appear in the surveillance region $\mathcal{R}$ for some duration $[t_a^k, t_b^k] \in [1, T]$. Each target moves in $\mathcal{R}$ at a random position at t_a^k, and moves out of $\mathcal{R}$ at t_b^k. At each time, a target disappears with probability p_z. The number of targets arriving at each time over $\mathcal{R}$ has a Poisson distribution with a parameter $\lambda_b V$ where λ_b is the birth rate of new targets per unit time, per unit volume V. Similarly, the number of false alarms has a Poisson distribution with a parameter $\lambda_f V$ where λ_f is the false alarm rate per unit time, per unit volume. The detecting probability of a noisy observation is p_d.The number of observations at time t is n(t). The purpose of MCMCDA is to find out the values K and $[t_a^k, t_b^k]$ (k = 1, 2, … K).

2) MCMC Algorithm

MCMCDA adopts the Metropolis-Hastings algorithm to generate samples from a distribution π on a solution space Ω by constructing a Markov chain with state $\omega \in \Omega$ and stationary distribution $\pi(\omega)$. The acceptance probability of a proposed state ω' is defined as:

$$A(\omega, \omega') = \min\{1, \frac{\pi(\omega')q(\omega'\omega)}{\pi(\omega)q(\omega,\omega')}\} \quad (1)$$

where ω is the current state and q is the proposal distribution.

Let $y(t) = \{y^i(t): i = 1, 2, \ldots, n(t)\}$ be all observations at time t and $Y = \{ y(t): 1 \leq t \leq T \}$ be all observations during the surveillance of T. The solution space Ω is defined to be a collection of partitions of observations Y, for $\omega \in \Omega$:

(1) $\omega = \{\tau_0, \tau_1, \ldots, \tau_K\}$;
(2) $Y = \cup_{k=0}^{K} \tau_k$ and $\tau_i \cap \tau_j = \phi$ for $i \neq j$;
(3) τ_0 is a set of false alarms;
(4) $|\tau_k \cap y(t)| \leq 1$ for $k = 1, 2, \ldots, K$ and $t = 1, 2, \ldots, T$; and
(5) $|\tau_k| \geq 2$ for $k = 1, 2, \ldots, K$.

The MCMCDA defines the stationary distribution $\pi(\omega)$ as:

$$P(\omega|Y) \propto P(Y|\omega) * P(\omega) \quad (2)$$

where

$$P(\omega) = \prod_{t=1}^{T} p_z^{z(t)}(1-p_z)^{c(t)} p_d^{d(t)}(1-p_d)^{g(t)} \lambda_b^{a(t)} \lambda_f^{f(t)}$$

$$P(Y|\omega) = \prod \tau \epsilon \omega \setminus \{\tau_0\} \prod_{i=1}^{|\tau|-1} \mathcal{N}(\tau_{i+1}|\mu, \sigma)$$

In this framework, the tracking problem is toto find a state ω^* with the maximum posterior among all of the checked states.

$$\omega^* = \arg\max(P(\omega|Y) \quad (3)$$

The Kalman filter is used to estimate the expected value μ and covariance σ. $P(Y|\omega)$ is the likelihood of observation; $z(t)$: the number of targets terminated at time t; $a(t)$: the number of new targets at time t; $d(t)$: the number of actual targets detected at time t; $e(t-1)$: the number of targets from time $t-1$; $c(t) = e(t-1) - z(t)$: the number of targets from time $t-1$ that has not terminated at time t; $g(t) = c(t) + a(t) - d(t)$: the number of undetected targets; $f(t) = n(t) - d(t)$: the number of false alarms.

B. Reduction of State Space Size

In [3] the authors make assumptions that any target has a maximal directional speed $\bar{v}$, and that the number of consecutive missing observations of any track is less than $\bar{d}$. To further accelerate the convergence rate of the Markov Chain, we distinguish abnormal from normal observations and identify the moving scope of an intruder at each monitoring time.

1) Distinguish abnormal observation from normal observation

When a sensor node receives a modified or fake message from an intruder it reports an abnormal observation to the cluster head. otherwise reports a normal observation. The state space is a collection of partitions of observations. Each partition has a number of tracks that consist of different observations. Each track can only include normal observation or abnormal observation, which reduces the size of the state space. Figure 4 shows an example of partition with classified observations.

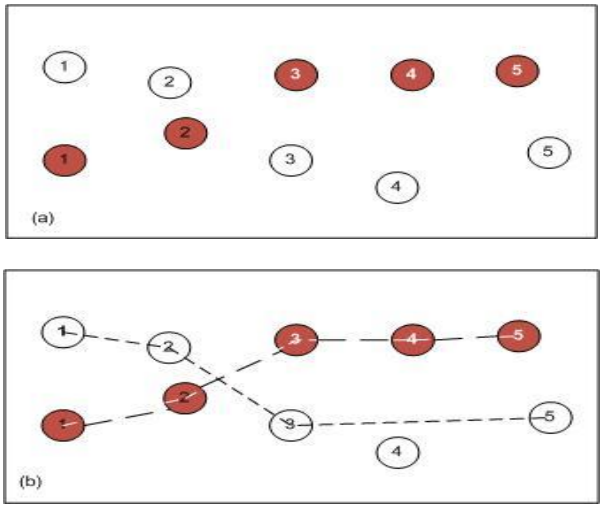


Figure 4. (a) an example of observation Y (solid circles represent abnormal observations and hollow circles represent normal observations. The numbers represent observation times); (b) an example of a partition ω of Y

2) Forecasting intruder position

We follow the method of [4], where it is assumed that a sensor node can sense an intruder approaching or moving away. We achieve this function by computing the energy level of signals, which requires small computational power [9].

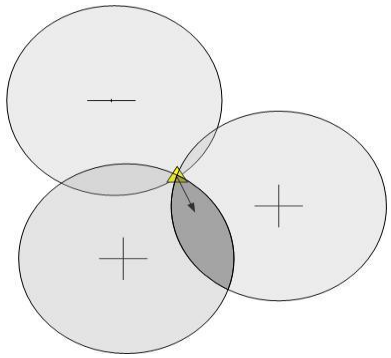


Figure 5. The future moving direction of an intruder within the shaded area

A sensor reports the movement trend of an intruder to the cluster head along with the normal or abnormal observation. The cluster predicts the movement of an intruder at a future time by collecting and analyzing received information from different sensors at a synchronized time and at the same position. Figure 5 illustrates three sensors. Two of them sense that an intruder is approaching them (+ symbol), and one senses that the intruder is moving away (- symbol). The triangle shows the intruder position. The prediction is that the future position of this intruder is in the shaded area.

VI. TABU SEARCH

We utilize a local search technique to sample from the obtained observations. The sampling process is divided into two steps. First, an initial feasible set of tracks is constructed. Second, an improved new neighboring set of tracks is found. We define a cost function for each track as below:

 ISBN: 978-1-61208-154-0

$$f(\tau_k) = \sum_{i=1}^{n-1}\left(\lambda d_{i,i+1} + \lambda^2 p_{i,i+1} + \lambda^3 t_{i,i+1}\right) \quad (k = 1 \ldots K) \tag{4}$$

We use τ_k for the kth track; n is the number of observations in track τ_k ; $d_{i,i+1}$, $t_{i,i+1}$ and $p_{i,i+1}$ are Boolean values representing if the distance of two sequential observations exceeds a threshold value; if the types of two sequential observations are identical; and if the relative position of two sequential observations is in the forecasted area (the shaded area in Figure 5).

An improved track must have cost that is no larger than the cost of the old track. The new neighboring set of tracks can include one or more than one improved tracks according to actual requirements. This new neighboring solution is used as a proposed state to calculate the acceptance probability in the MCMC algorithm.

A neighborhood structure for defining moves based on ejection chains is well-known for the traveling salesman problem [21]. We use neighborhoods defined by ejection chains to produce effective moves with reduced computational effort. This method is a local search optimization technique which tries to minimize a cost function F(x), where x represents a parameter vector, by iteratively moving from a solution x to a solution x0 in the neighborhood of x until a stopping criterion is satisfied or a predetermined number of iterations N is reached. Algorithm 1 and 2 show the ejection chain and Tabu search methodologies.

We developed an ejection chain algorithm for the Tabu Search (TS) framework to determine if a trial set of tracks from received observations during a surveillance period. We group the observations by time period, then identify trial solutions in each group of observations using a proposed ejection chain algorithm. A trial solution is a connection among observations in a group. Ultimately, we combine all these group trial solutions to get a final trial set of tracks. Figure 6 and 7 illustrate the method. This approach partitions an optimization problem into relatively independent sub-optimization problems, and accelerates the optimization process.

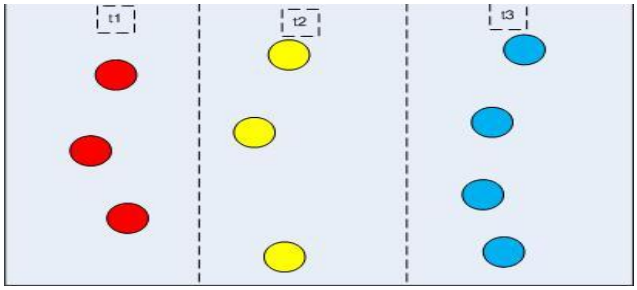
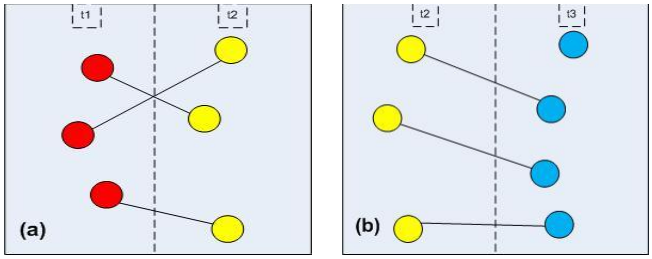


Figure 6. An example of observation Y (from time t1 to t3)



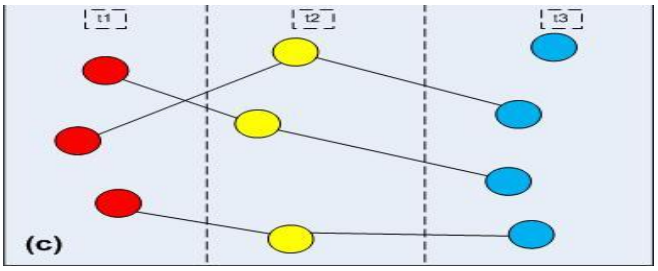


Figure 7. Illustration of searching for a trial set of tracks from observation Y (a) trial solution of group1; (b) trial solution of group2; (c) the whole trial solution

Algorithm 1 Ejection Chain Algorithm (ECA)

$n1$: Number of observation in the first column;
$n2$: Number of observation in the second column;
L: Current Ejection level;
L^*: The current best ejection level;

1: Set $L = 1$, $L^* = L$.
2: Create the first level of the ejection chain
(a) Start from the node which has the largest track cost;
(b) Try to generate a new trial solution with no larger cost;
(c) If no such trial solution, go to step 4;
(d) Update current trial solution;
(e) If no ejection occurred, go to step 4;
(f) Record the last ejection node $\boldsymbol{e^L}$.
3: Increase the chain to further levels (ref [19])
(a) Set $L = L + 1$;
(b) Start from $\boldsymbol{e^L}$, determine a new element that doesn't increase the cost;
(c) Update current trial solution and $\boldsymbol{e^L}$;
(d) If $L < Max\_LEVEL$ and L $< Min(n1, n2)$ go back to step 3. Otherwise go to step 4;
4: Get a new trial solution S'; Exit.

Algorithm 2 Tabu Search Algorithm (TSA)

Temp: flag;
TM: the maximum value of iteration

1: Generate a starting solution in S randomly, let $S^* = S$;
2: Call Ejection Chain Algorithm.
3: If improving: set Temp = 0, update the best solution with the new current solution, $S^* = S'$; otherwise Temp++;
If Temp < TM return to step2, otherwise Exit.

VII. SIMULATION RESULTS

We wrote the MCMC algorithm and Tabu Search algorithm in the C++ language with Matlab interfaces, and implemented the DCA algorithm in the Java Agent DEvelopment Framework (JADE). The results of running the DCA algorithm are saved in a file that is called by the MCMC algorithm, KNQ algorithm and Ejection Chain algorithm. The surveillance area

 ISBN: 978-1-61208-154-0

is a $\mathcal{R} = [0, 100] \times [0, 100] \in \mathbb{R}^2$ rectangular region, and 200 sensor nodes are randomly deployed in the area.

Four experiments are implemented. The first experiment is used to measure and analyze the efficiency of the DCA algorithm on identifying malfunctioning sensor nodes; the second assesses the efficiency of the KNQ algorithm implemented by cluster heads. The KNQA verifies the credibility of received abnormal observations. The last two experiments show that the local search algorithm accelerates convergence of the Markov chain.

A. *Experiment 1-DCA algorithm*

We randomly deploy 10, 20, 30, 40 and 50 intruders in the test area and perform experiments on each case. In each case we take 10 samples. Figure 8 shows the results for detecting malfunction sensor nodes. The results show the DCA algorithm can detect more than 90% of the attacked sensor nodes when fewer than 10% of the intruders are harmful. The algorithm needs an initial period of time and a cache to monitor and identify a malfunctioning node. This explains why there is a low detecting rate at the beginning of the sampling times, and decreased detecting rate when there are more harmful intruders.

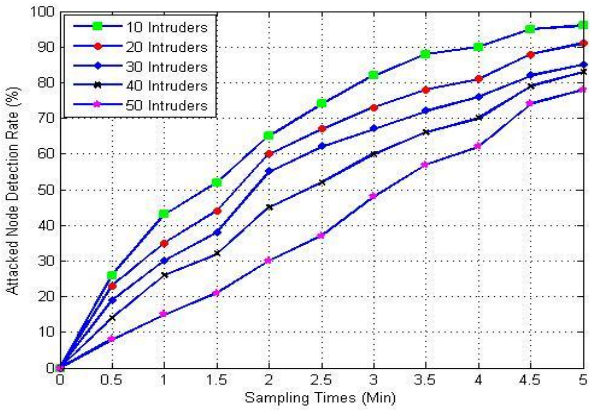


Figure 8. Sampling time vs. attacked node detection rate with different number of harmful intruders

B. *Experiment 2 – K-Neignbor Query Algorithm (KNQA)*

The accuracy probability of receiving a message from a suspect harmful intruder is defined as:

$$P_a = 1 - \frac{1}{n}\sum_{i=1}^{n} \cos\theta_i \quad (5)$$

where n is the number of neighbor nodes, and θ_i is the angle of the ith neighbor node, the suspect intruder and the current node. Only those neighbors that have $\theta_i \epsilon [0^0, 90^0]$ are considered. In this experiment, we analyze the effectiveness of the KNQ algorithm on detecting abnormal observation based on the data from the first experiment. There are five groups of data for each number of intruders in a WSN. The analysis is for the group with 20 intruders. The threshold value for the accuracy probability in (5) is 0.85. The number of querying node is 10.

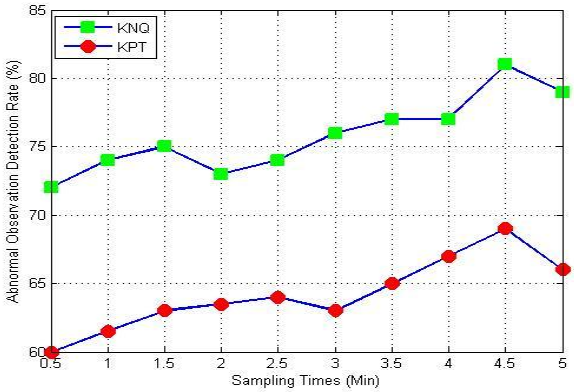


Figure 9. KNQ vs. KPT on detecting abnormal observations

Figure 9 indicates that the KNQ algorithm has higher detection rate than KPT algorithm when considering interference among neighbor nodes. At sampling time 4.5, the KNQ has an obvious higher detection rate than KPT, which is caused by some nodes densely distributed in a particular area.

C. *Experiment 3 –MCMC with Tabu Search*

In this experiment, we utilize the same simulation settings as in [3]. The surveillance area is a $\mathcal{R} = [0, 100] \times [0, 100] \in \mathbb{R}^2$ rectangular region. The number of mobile targets K varies from 10 to 50. The other parameters are: $T = 10$, $p_d = 0.8$, $\lambda_f V = 1.0$, $\lambda_b V = 1$, $p_z = 0.01$, $\bar{d} = 5$, $\bar{v} = 100$ unit lengths per unit time. The state vector is $x = [x, y, v_x, v_y]^T$ where (x, y) is a coordinate and (v_x, v_y) is a velocity vector. The Kalman filter is used to estimate the states of an target, and the models are:

$$x_t = Ax_{t-1} + Bw_k$$
$$y_t = Hx_t + v_t$$

where

$$A(\Delta) = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} B(\Delta) = \begin{bmatrix} \Delta^2 & 0 \\ 0 & \Delta^2 \\ \Delta & 0 \\ 0 & \Delta \end{bmatrix} H(\Delta) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}^T$$

w_k and v_k are white noises with Gaussian distributions $\mathcal{N}(0, \sigma_w^2)$ and $\mathcal{N}(0, \sigma_v^2)$ respectively, where $\sigma_w^2 = \text{diag}(100, 100)$ and $\sigma_v^2 = \text{diag}(20, 20)$. To estimate the efficiency of finding the optimal solution among a state space, we specify value 0.9 as a threshold for $P(\omega|Y)$ defined in equation (2).

1) *Number of samples vs Number of tracks*

A MCMC algorithm usually iterates some fixed number of times and the maximum posterior of a state is the optimal solution. In our case, the goal is to find the state defined in (3). We use the number of iterations required to find the state that has a larger posterior then a threshold as the criterion to evaluate convergence speed. We compare the algorithm with Markov Chain Monte Carlo Data Association (MCMCDA) by running each algorithm 10 times.

 ISBN: 978-1-61208-154-0

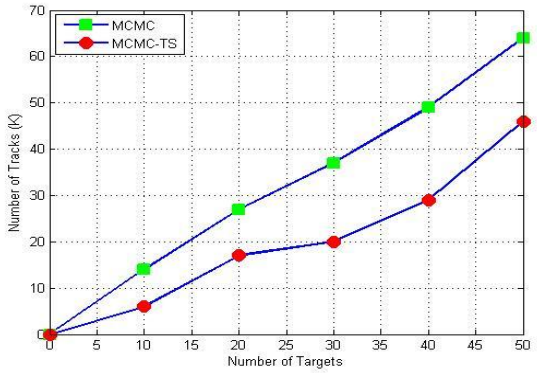


Figure 10. Number of targets vs. number of tracks

The number of tracks is significantly increased with the increase of targets. That means a longer time is needed to identify the real target tracks among the options. Figure 10 shows that the algorithm reduces the number of optional tracks.

2) Average running time vs number of tracks

The running times of the algorithms with and without state space reduction and ejection chains are shown in Figure 11. The improved MCMC procedure is significantly faster.

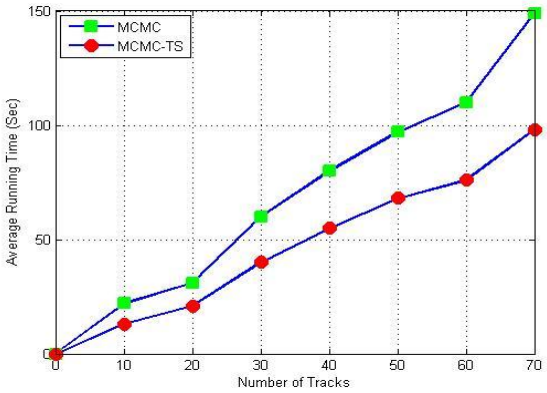


Figure 11. Average running time vs. number of tracks

VIII. CONCLUSION AND FUTURE WORK

A two-phase security mechanism that combines Dendritic Cell, MCMC and Ejection Chain algorithms was developed and tested. The DCA algorithm detects malfunctioning sensor nodes and prevents damage to a WSN by ceasing response to requests from these nodes. The MCMC procedure simultaneously tracks multiple harmful mobile intruders. Instead of randomly samplings from the reported observations, ejection chains and a Tabu Search are used together to selectively sample from the observations and accelerate the convergence rate. Our results establish that our new security mechanism promptly identifies trouble makers.

REFERENCES

[1] J.Winter and W. Lee. Kpt, "A Dynamic KNN Query Processing Algorithm for Location-aware Sensor Networks," Proc. of DMSN, 2004.

[2] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," In Proceedings of the 6th Annual Int. Conference on Mobile Computing and Networking, pp. 243–254, 2000.

[3] S. Oh, S. Russell, and S. Sastry, "Markov Chain Monte Carlo Data Association for General Multiple-Target Tracking Problems," in Proc. of the IEEE International Conference on Decision and Control, Paradise Island, Bahamas, Dec. 2004.

[4] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a Moving Object with Binary Sensors," Proc. ACM SenSys Conf., Nov. 2003.

[5] J. J. Zhao and K. E. Nygard, "A Dendritic Cell Inspired Security System in WSNs," FUTURE COMPUTING 2010, Int. Conference on Future Computational Technologies and Applications, Lisbon, Nov. 21, 2010.

[6] B. Krisnamachari and S. S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," IEEE Transactions on Computers, Vol. 53, No. 3, Mar. 1, 2004.

[7] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for wireless networks," in: Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), Boston, MA (August 2000).

[8] P. Chen, S. Oh, M. Manzo, B. Sinopoli, C. Sharp, K. Whitehouse, G. Tolle, J. Jeong, P. Dutta, J. Hui, S. Shaert, S. Kim, J. Taneja, B. Zhu, T. Roosta, M. Howard, D. Culler, and S. Sastry, "Experiments in instrumenting wireless sensor networks for real-time surveillance," In International Conference on Robotics and Automation (video), 2006.

[9] W. Chen, J. Hou, and L. Sha, "BDynamic clustering for acoustic target tracking in wireless sensor networks," in Proc. of the 11th IEEE Int. Conf. Network Protocols, Nov. 2003, pp. 284–294.

[10] F. L. Lewis, "Wireless sensor networks," In D. J. Cook and S. K. Das, editors, Smart Environments: Technology, Protocols, and Applications. Wiley, 2004.

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," IEEE Communications Magazine 40, 8 (Aug.), 102—114, 2002.

[12] J. P. Walthers, Z. Liang, W. Shi and V. Chaudhary, "Wireless Sensor Networks Security: a Survey," Report MIST-TR-2005-007, 2005.

[13] J. Kim and P. J. Bentley, "Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection," Proc. Genetic and Evolutionary Computation Conference (GECCO), 2001.

[14] S. Forrest, S. Hofmeyr, and A. Somayaji, "Computer Immunology," Communications of the ACM, 40(10):88-96, 1997.

[15] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod, "Danger theory: The link between AIS and IDS," Proceedings of the Second International Conference on Artificial Immune Systems (ICARIS 2003), vol. 2787 of LNCS, Springer-Verlag; pp. 147‑155, 2003.

[16] N. Mazhar and M. Farooq, "A sense of danger: dendritic cells inspired artificial immune system for MANET security," GECCO 2008: 63-70.

[17] A. P. da Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks (Q2SWinet '05). ACM Press, October 2005, pp. 16–23.

[18] D. B. Reid, "An Algorithm for Tracking Multiple Targets," IEEE Transaction on Automatic Control, 24(6):843–854, December 1979.

[19] C. Rego, T. James and F. Glover, " An Ejection Chain Algorithm for the Quadratic Assignment Problem," Networks, 2009.

[20] M. K. Cowles and B. P. Carlin, "Markov chain Monte Carlo convergence diagnostics: a comparative review," Journal of the American Statistical Association 91, 883–904, 1996.

[21] F. Glover, " Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem," Working paper, College of Business & Administration, University of Colorado, Boulder, CO, 1991.

[22] Y. Hu, A. Perrig and D. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks," IEEE Annual Conference on Computer Communications (INFOCOM), 2003, pp. 1976–1986.

[23] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," IEEE Computer Magazine, October 2002, pp. 54-62.