A TWO-PHASE SECURITY MECHANISM FOR ANOMALY DETECTION IN WIRELESS SENSOR NETWORKS

A Dissertation
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Jingjun Zhao

In Partial Fulfillment
for the Degree of
DOCTOR OF PHILOSOPHY

Major Department:
Computer Science

June 2012

Fargo, North Dakota

North Dakota State University
Graduate School

Title

A Two-phase Security Mechanism For Anomaly Detection in Wireless Sensor Networks

By

Jingjun Zhao

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

SUPERVISORY COMMITTEE:

Dr. Kendall Nygard
Chair

Dr. Juan Li

Dr. Changhui Yan

Dr. Chao You

Approved:

05/21/2013
Date

Dr. Brian M. Slator
Department Chair

ABSTRACT

Wireless Sensor Networks (WSNs) have been applied to a wide range of application areas, including battle fields, transportation systems, and hospitals. The security issues in WSNs are still hot research topics. The constrained capabilities of sensors and the environments in which sensors are deployed, such as hostile and non-reachable areas make the security more complicated.

This dissertation describes the development and testing of a novel two-phase security mechanism for hierarchical WSNs that is capable of defending both outside and inside attacks. For the outside attacks, the attackers are usually malicious intruders that entered the network. The computation and communication capabilities of the sensors restrict them from directly defending the harmful intruders by performing traditionally encryption, authentication, or other cryptographic operations. However, the sensors can assist the more powerful nodes in a hierarchical structured WSN to track down these intruders and thereby prevent further damage. To fundamentally improve the security of a WSN, a multi-target tracking algorithm is developed to track the intruders. For the inside attacks, the attackers are compromised insiders. The intruders manipulate these insiders to indirectly attack other sensors. Therefore, detecting these malicious insiders in a timely manner is important to improve the security of a network. In this dissertation, we mainly focus on detecting the malicious insiders that try to break the normal communication among sensors, which creates holes in the WSN. As the malicious insiders attempt to break the communication by actively using HELLO flooding attack, we apply an immune-inspired algorithm called Dendritic Cell Algorithm (DCA) to detect this type of attack. If the malicious insiders adopt a subtle way to break the communication by dropping received

packets, we implement another proposed technique, a short-and-safe routing (SSR) protocol to prevent this type of attack.

The designed security mechanism can be applied to different sizes of both static and dynamic WSNs. We adopt a popular simulation tool, ns-2, and a numerical computing environment, MATLAB, to analyze and compare the computational complexities of the proposed security mechanism. Simulation results demonstrate effective performance of the developed corrective and preventive security mechanisms on detecting malicious nodes and tracking the intruders.

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to Dr. Kendall E. Nygard, my academic advisor, for his time and expertise all these years. Without his guidance and persistent help this dissertation would not have been possible. I would also like to acknowledge the contributions of my committee, Dr. Juan Li, Dr. Changhui Yan, and Dr. Chao You.

I want to give special thanks to my parents, old sister, and friends for their continuous encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CHAPTER 1. INTRODUCTION

Improved wireless communication and electronics promote the development of low-power, low-cost and multifunctional sensor nodes [1]. 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 packet dropping, packet change, and energy-exhaustion. Traditional cryptographic security algorithms assume that all nodes are cooperative and trustworthy [2]. This assumption is not satisfied in most real-world WSN applications consequently traditional security approaches do not always apply to WSNs composed of large number of sensors with limited power and defense ability.

1.1. Motivation

Wireless Sensor Networks (WSNs) consist of a number of nodes that are capable of communicating with each other absent a fixed infrastructure. However, arbitrary node movements and lack of a centralized control make them vulnerable to a wide variety of attacks from inside as well as from outside. Therefore, providing effective security protection is important to ensure continued viability of WSNs. In general, two complementary approaches exist to protect a system: prevention and detection.

Intrusion prevention techniques, such as encryption and authentication, attempt to deter and block attackers. Unfortunately, prevention techniques can only reduce intrusions, not completely eliminate them [1, 2]. Despite the amount or quality of intrusion prevention

measures, an intelligent attack can exploit a single security hole to break into a system. Nothing is absolutely secure. Therefore, intrusion detection systems (IDSs) are indispensable for a reliable system. They serve as an important secondary line of defense. Intrusion detection can be based either on detecting malicious insiders or detecting harmful intruders.

A malicious nodes detection technique identifies those nodes that have been attacked by harmful intruders and behaved harmfully to the normal operation of the sensor network. The malicious nodes may drop received packets instead of transmitting them. Such behavior can create holes in the network and damage the communications between sensors and the central center. If a malfunctioning node is recognized, an alarm is generated. No more packets are sent to this node. But more normal sensor nodes can be attacked if harmful intruders still exist in the sensor network. Therefore, detecting harmful intruders is needed.

To prevent harmful intruders causing further damage, the first important step is to locate the intruders. In comparison with static intruders, mobile intruders commonly exist in practical applications, and are more difficult to track. Most algorithms use the sensor network to actively participate in the tracking process, sensing the target and propagating information regarding its position. The trade-off, in this case, is that there is a high message and energy consumption overhead. There are also issues concerning the scalability of those solutions and how they can be mapped into low density sensor networks; therefore a tracking algorithm that operates with low message overhead and low power consumption while scaling well is desired. A tracking technique based on Markov Chain Monte Carlo is developed in this dissertation to address these issues.

1.2. Definition of Problems

In this dissertation, we mainly explore three general but important security problems in wireless sensor networks (WSNs).

1.2.1. Misbehavior Detection Problem

In a wireless sensor network, sensors equipped with limited resources communicate wirelessly with each other. It is inevitable to have unexpected behavior happening. However, some sensors' behavior, like reporting messages with irregular high frequency or continually dropping packets, may seriously damage the normal operation of the network. Therefore, promptly detecting these sensors is necessary to ensure achieving the intended purpose of the network. In many cases, sensors are deployed in the environments where a central monitoring system is difficult or expensive to set up. It means that the sensors can only depend on themselves to identify misbehaving neighbors around them. The problem is how they distinguish misbehaving neighbors from normal ones with tolerable false positive rate. Deployed sensors are often used to monitor an environment and transmit collected information back to the base stations. So detecting misbehaving neighbors should not consume much of these sensors resources, even though it is an import task. What techniques can common sensors adopt to precisely and effectively detect misbehaving neighbors that mainly conduct flooding attack and packet dropping attack? This is the first problem addressed in this dissertation.

1.2.2. Misbehavior Monitoring Problem

Gray hole attack, in which the attackers only transmit routing packets but drop all or part of received data packets, is a common type of attack in WSNs. Comparing with other active attack like HELLO flooding attack, gray hole attack is more difficult to be detected because of its concealment. The longer the attackers exist in a WSN, the more damage they can make to the

network. Therefore detecting gray hole attackers can improve the security level of a WNS. In wired network, real-time monitoring is usually adopted to prevent attacks from outsiders or compromised insiders. Real-time monitoring is an ideal method, but it is impractical to use this method in WSNs because it will consume much of the sensors' energy. So the periodic monitoring method is naturally adopted by many researchers in the sensor network security area. The direct problem is what monitoring cycle should be chosen to maximize detection rate and minimize the energy consumption. To our knowledge, there are no such effective rules made for this decision yet. Besides these monitoring methods, are there any other techniques that we can use to meet a desired detection rate as well as to utilize energy effectively? This is another problem we try to solve in this work.

1.2.3. Multi-target Tracking Problem

When compromised insiders in a WSN are detected, they can't create further damage to the network. But that doesn't mean the WSN is safe. The malicious intruders are the true culprits. They can compromise more sensor nodes to keep damaging the WSN indirectly. Therefore, an active defense mechanism is required to maintain the security of a WNS. One of effective mechanisms is to track down the mobile malicious intruders in a WSN. This belongs to the location tracking problem, whose goal is to track the roaming paths of the moving objects in the area in which sensors are deployed. The essence of the target tracking problem is to find tracks from noisy measurements (observations), i.e., to associate measurements to the objects that really generate these measurements; more precisely, a partition of measurements is formed such that each element of a partition is a collection of measurements generated by a single target or cluster. It is not difficult to imagine that the number of optional partitions will rapidly increase with the number of mobile targets. So it is a challenge to quickly discover the true partition

among the numerous options. How to speed up the discovery of the objects' tracks is the last problem we try to explore in this work.

1.3. Contributions

The central challenge in security is determining the difference between normal and potentially harmful activity. An Artificial Immune System (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. A particular class of AIS methodologies called Negative Selection Algorithms (NSAs) has been applied to anomaly detection problems [4]. 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) [3]. Following another type of AIS, the work in [5] 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 [6] established a security system based on the behaviors of Dendritic Cells. The work reported in [7] 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 invaders.

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 [8]. The data association problem is to identify the tracks of targets from noisy observations of target positions at known points in time. The multiple hypothesis tracker (MHT) [10] algorithm is the prominent methodology for solving the data association problem. In this setting, a hypothesis is an association of a set of observations with a target. A set of hypotheses is developed over time as observations become available over a monitoring period. At the end of the monitoring, the hypothesis with highest posterior is identified as the best solution. MHT is effective in detecting variable numbers of targets, but has high computational complexity since the number of hypotheses grows exponentially over time. In [8], 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 MHT 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. Convergence rate is an important criterion on assessment of the MCMC algorithm ability [11]. The authors make two additional assumptions: (1) the maximal directional speed of any target is less than $\overline{v}$; and (2) the number of consecutive missing observations of any track is less than $\bar{d}$. These assumptions make the computations of the proposal distribution easier. In our work, we further improve the convergence rate of the Markov Chains used in the MCMC approach. We classify observations into two types: normal observations and abnormal observations. We assume that each sensor node can sense an intruder approaching or moving away. Since a sensor can use the received signal strength to infer an intruder's moving tendency, this assumption is realistic. This assumption is then used to predict the moving area of an

intruder at some time, calculated through the sensing of the energy level. We apply a Tabu Search technique [97] to the optimal solution searching process. These improvements decrease the size of the state space of the Markov Chain and speed up the convergence rate. Comparing with a full MCMCDA, our algorithm needs fewer samples to reach an optimal solution.

In this work, we focus on two types of attacks: flooding attack and packet-dropping attack. These two types of attacks can cause serious interference to the normal operation of a WSN. They can create communication holes in a WSN and increase delay of information transmission, which are intolerable for some practical applications. Figure 1 shows the types of concerned attacks. Each of these attacks can have different defense strategies and adopted techniques. We try to adopt and design some security methods that meet both the security needs of applied applications and sensors characteristics. Figure 2 and 3 show the used countermeasures and techniques. To evaluate the effectiveness of each designed method, we also set some performance metrics for each method. Figure 4 shows the chosen performance metrics.

The main contribution of our work is the development of a two-phase security mechanism for WSNs with hierarchical structure by combining a DCA with a multi-target tracking algorithm and a Tabu Search technique. This development is based on similar validated research we did in [39]. The two-level hierarchical sensor network that we adopt for our experiments has a backbone of sparsely placed static high-end sensor 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 [9]. The DCA works on the low-end nodes and identifies malfunctioning neighbors that have been attacked by harmful intruders. The primary malfunctions include packet dropping 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 multi-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.

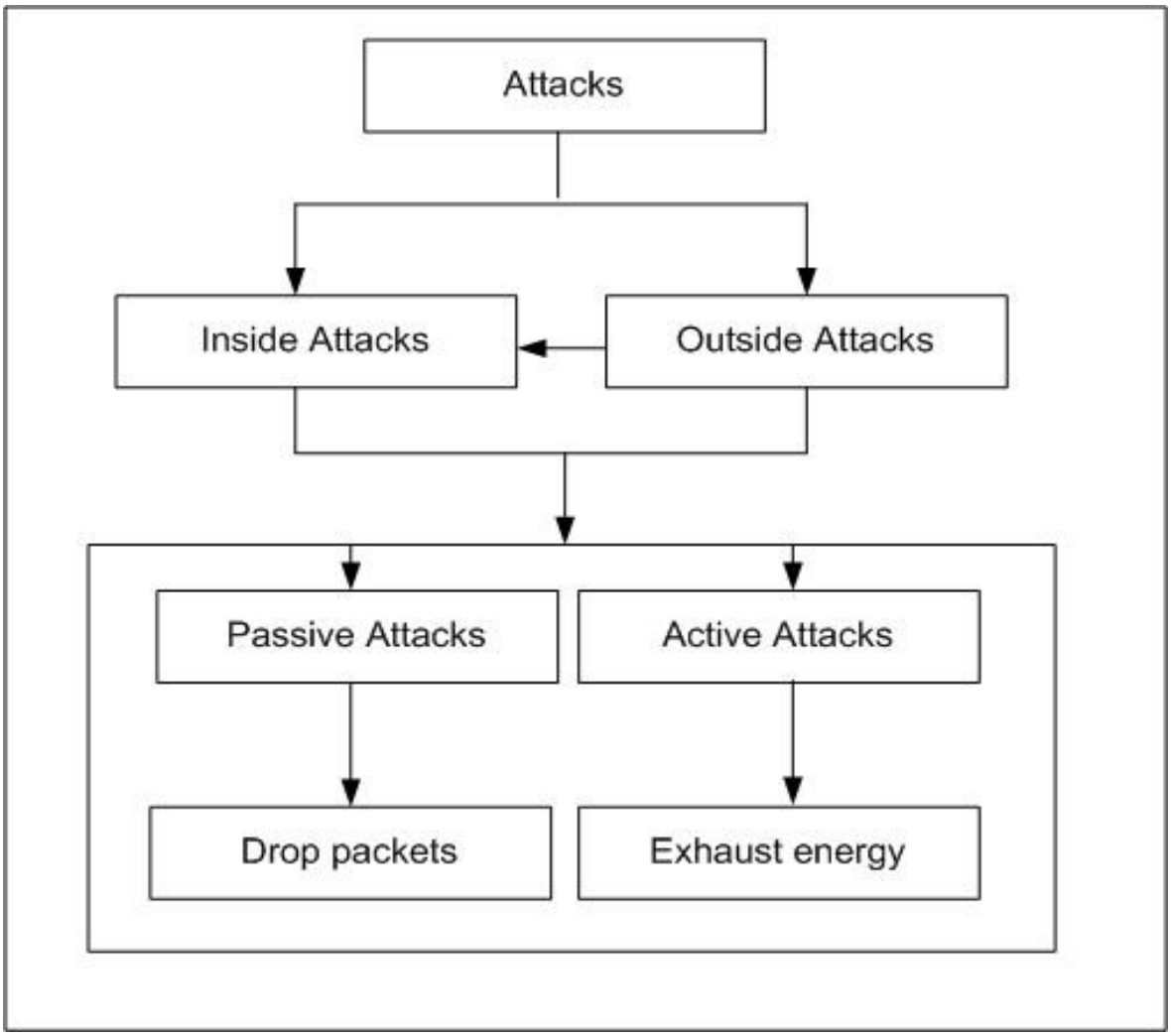


Figure 1. Types of attacks in WSNs

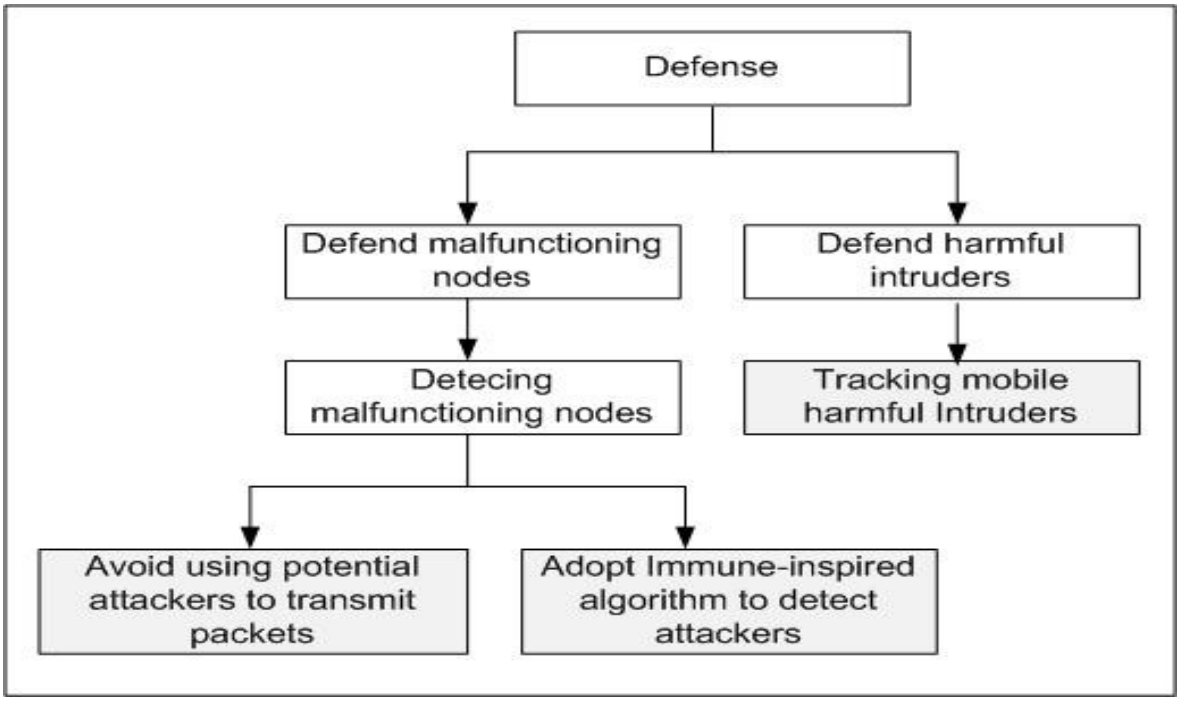


Figure 2. Defense strategies

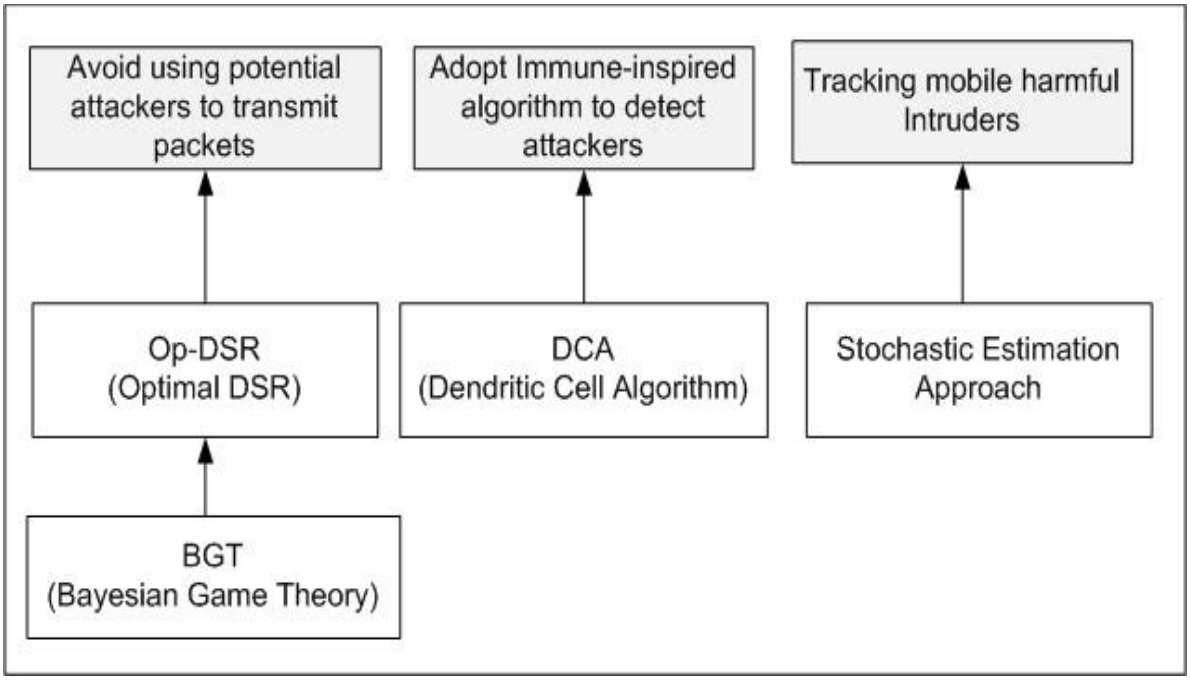


Figure 3. Adopted techniques for implementing defense strategies

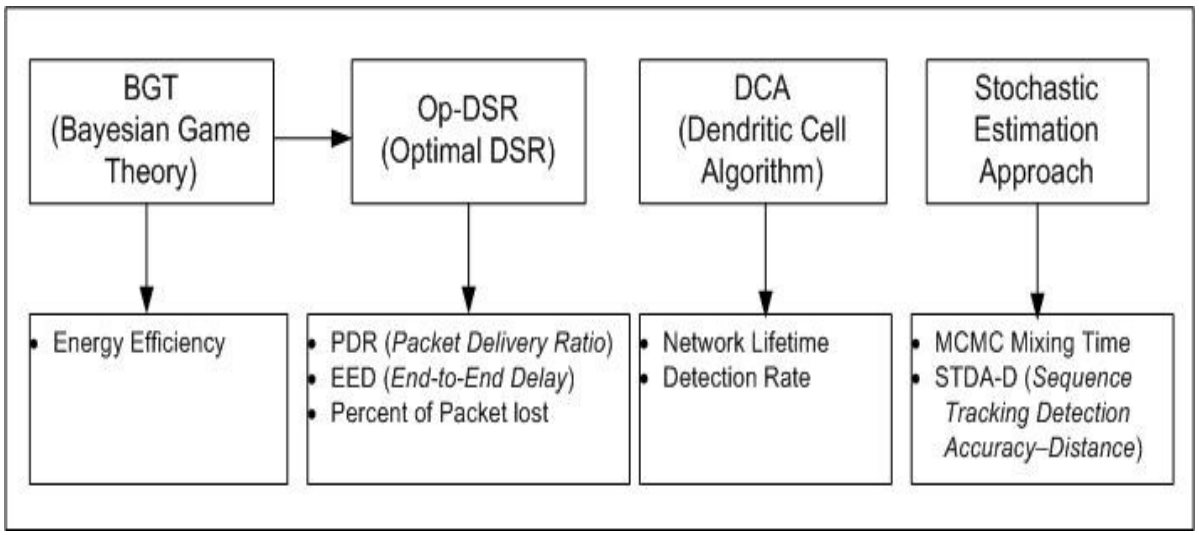


Figure 4. Evaluated performance metrics

Figure 5 shows the two-phase security architecture, in which hollow circles represent normal sensors, the solid circle represents a malfunctioning sensor, and the stars represent mobile intruders. Each low-end sensor node that runs the proposed DCA is responsible for detecting malicious neighbors. This DCA helps the sensor node identify deliberate misbehavior except occasional misbehavior such as a sensor node reports inaccurate data when its surrounding environment has sudden changes. An energy-consumption attack like HELLO flood attack can easily be detected by this immune-inspired algorithm. Another feature in the designed security mechanism is that Bayesian game theory based monitoring strategies are adopted to help a sensor node decide when to monitor a potential malicious node, which can effectively save energy without losing accuracy. The sensed target position is sent to the cluster head where the multi-target tracking algorithm is implemented. Finally, the tracking results are sent to the base station.

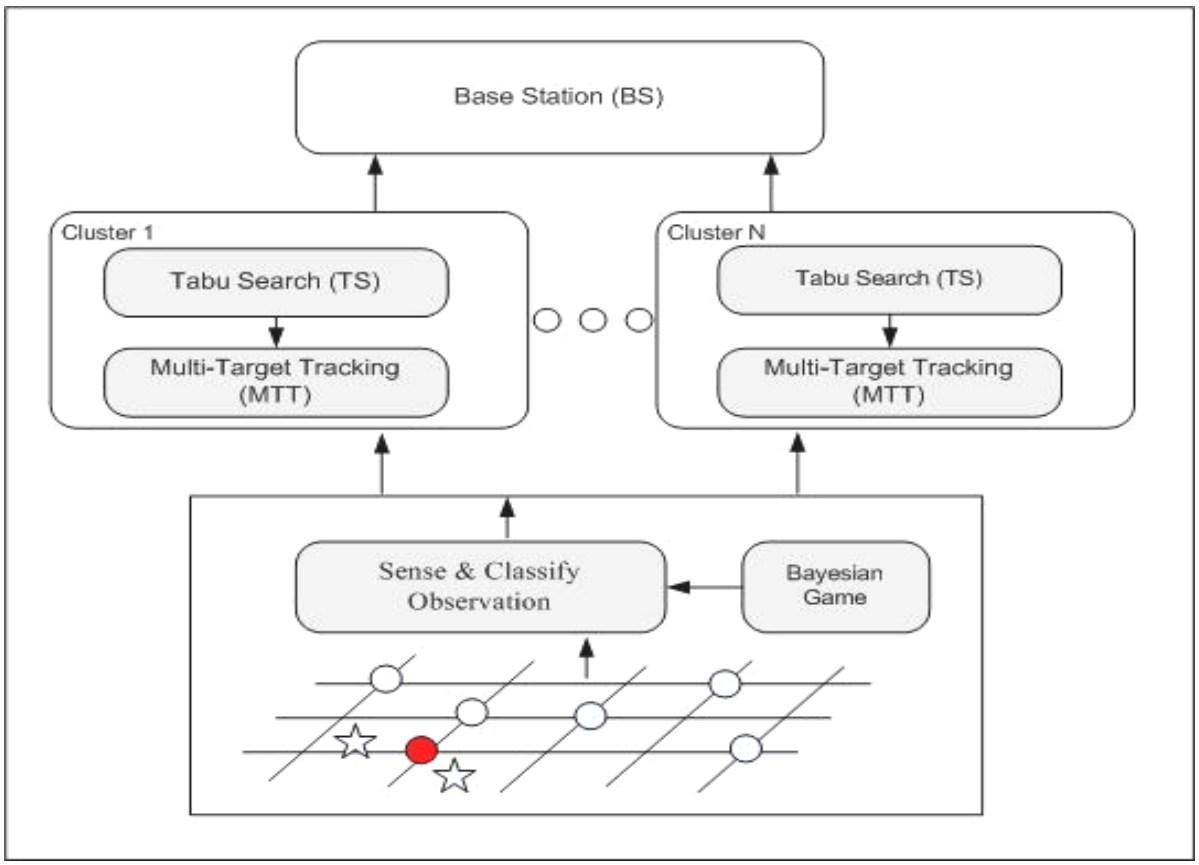


Figure 5. The two-phase security architecture

The rest of this dissertation is organized as following. Chapter 2 includes two sections of reviews: one for recent development in AIS in general; another for some popular techniques used for multi-target tracking. Chapter 3 explores some security problems in wireless sensor networks. Chapter 4 details the proposed security methods for preventing malicious insiders. Chapter 5 explores the basic techniques on target tracking, and details the improved multi-target tracking method for tracking malicious outsiders. Chapter 6 introduces some basic concepts of Tabu search, and details the proposed Tabu search method used for assisting the target tracking. Chapter 7 analyzes the simulation results and compares them with some other algorithms. Finally, the conclusions and future works are given in Chapter 8.

CHAPTER 2. BACKGROUND AND LITERATURE REVIEW

2.1. Artificial Immune System

Forrest and Hofmeyer [25], [26] use AIS for intrusion detection in wired local area networks. Their work is based on the negative selection part of the self - nonself model and some form of danger signal. TCP connections play the role of self and nonself cells. One connection is represented by a triplet encoding the sender's destination address, the receiver's destination address and the receiver's port number. A detector is a bit sequence of the same length as the triplet. A detector matches a triplet if both have M contiguous equal bits, where M is a fixed system parameter. Candidate detectors are generated randomly; in a learning phase, detectors that match any correct (i.e., self) triplets are eliminated. This is done offline, by presenting only correct TCP connections. Non-eliminated detectors have a finite lifetime and die unless they match a nonself triplet, as in the immune system. The danger signal is also used: it is sent by humans as confirmation in case of potential detection. This is a drawback, since human intervention is required to eliminate false positives, but it allows the system to learn changes in the self. With the terminology of statistical pattern classification, this use of the danger signal can be viewed as some form of supervised training. Similarly, Dasgupta and Gonz´alez [32] use an AIS approach to intrusion detection, based on negative selection and genetic algorithms. Kim and Bentley [3] show that straightforward mappings have computational problems and lead to poor performance, and they introduce a more efficient representation of self and nonself than in [25]. They show the computational weakness of negative selection and add clonal selection to address this problem [3]. In their subsequent papers, they examine clonal selection with negative selection as an operator [28], and dynamical clonal selection [29], showing how different

parameters impact detection results. For an overview of AIS, see the book by de Castro and Timmis [31] and the paper by de Castro and von Zuben [30].

2.2. Multi-target Tracking

Multi-object tracking algorithms depend on a single object tracking algorithm when objects are well-separated from one another. When objects are close to one another, the data association problem needs to be solved - there are exponentially many ways to associate measurements to known states. There are two well-known methods for dealing with the data association problem - the Multiple Hypothesis Tracker (MHT) in [66] and the Joint Probabilistic Data Association Filter (JPDA) in [67]. Figure 6 demonstrates how MHT works (the `X' marks denote new measurements, and triangles and circles denote tracks) - whenever new measurements are available from sensors, MHT generates hypotheses. In the worst case, the number of hypotheses can grow exponentially, and MHT uses heuristics such as gating (a measurement can be associated with a state only when it is within some distance from the state), pruning (one can compute the probabilities of the hypotheses and eliminate unlikely hypotheses), and N-scan-back (this heuristic considers only measurements within the recent N time steps). The main advantage of MHT is its ability to deal with the unknown number of objects even in a cluttered environment, where there could be many false and missing measurements. The flexibility of MHT comes at the cost of exponential storage and running time, and it is crucial to have good heuristics to prune unlikely hypotheses. Despite its worst case complexities, it has been used extensively in military applications due to its near-optimal performance and has been successfully implemented in the visual tracking community [69].

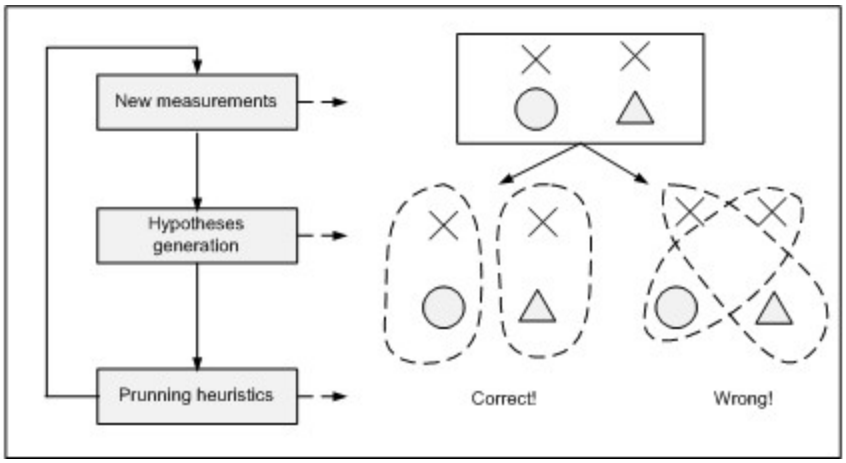


Figure 6. MHT illustration

JPDA is inherently a sub-optimal solution for the data association problem, since it considers only the most recent set of measurements. Given a state S_k^i at time kk, JPDA combines all the measurements$\{Z_k^1, \dots Z_k^N\}$ in a probabilistically consistent manner as follows,

$$p(S_k^i|Z_k) = \sum_j \alpha_{ij}\, p\left(Z_k^j|S_k\right) p(S_k^i|Z_{k-1}))$$

where α_{ij} is a probability of associating the jth measurements Z_k^j to the ith state S_k^i and $\sum_j \alpha_{ij} = 1$. As one can see from the above equation, JPDA has the polynomial running time and can be used with general dynamic models - JPDA does not specify how to compute a predicted posterior $p\left(S_k^i \middle| Z_{k-1}\right)$.

In [71], the authors proposed a formal probabilistic approach called Identity Uncertainty to reasoning about identity under the framework of first-order logics of probability. Their setting is very general in that their framework allows one to pose identity questions with varying number of objects, properties and relations.

Approximate inference for their formal language can be done using Markov Chain Monte Carlo (MCMC) algorithms [73], where the Markov chain is defined on a state space consisting of

the relational models of the first-order language. There are some notable differences between their approach and what we propose in this dissertation. First, their approach does not assume unique identities - in fact, the number of unique identities is an identity question posed in their language, while our approach assumes unique identities and exploits the exclusion among them to update global identity information. Second, their method does not seem to be easily distributable, while one of the focuses of this dissertation is to design a framework that can be implemented in a purely distributed fashion in a WSN. Finally, the two approaches are fundamentally different in their goals - their approach concerns inference on identity, while we use identity information to update probabilities of different identities in a WSN. In fact, their approach can be used to implement a local identity sensing module in our framework, whose outputs can then be used by our approach to update global identity information.

The general-purpose multi-target tracking algorithms such as the joint probabilistic data association filter (JPDAF) [74] and multiple hypothesis trackers (MHT) [75] are robust against the low detection probability and high false alarm rate. But they are not suitable for sensor networks since track initiation and termination is difficult with JPDAF and both JPDAF and MHT require large memory and computation cycles. Since MHT can initiate and terminate tracks, the tracking task can be easily distributed in a network of sensors. In [76], a distributed tracking algorithm based on MHT is developed for multiple sensors. But the approach is not suitable for sensor networks since it demands large computational power and a large amount of memory on each sensor.

In [77], the authors propose to use a classification algorithm to disambiguate closely located targets. But signals received from targets are correlated and we cannot recover the uncorrelated signals in all cases. Since we do not know in advance the number of targets around

each sensor, the problem is ill-posed and very challenging even for a high-end computer. In [79], distributed track initiation and maintenance methods are described. By electing a leader among the sensors by which a target is detected, unnecessary communication is reduced while tracking targets using the nearest neighbor method. But considering the complexity of the data association problem, the approach will suffer from incorrect associations when there are many targets crossing or moving close to each other. In addition, when the false alarm rate is high, the proposed approach will overflow the network with spurious tracks and it is unclear how the missing observations are handled. Recently, some multi-target tracking algorithms specifically designed for sensor networks have been proposed to solve the identity management problem [78]. They assume the availability of a classification algorithm as in [77] but the disambiguation is delayed until targets are sufficiently separated. As assumed in the simulations of [78], when the targets are of different classes, a target can be classified by the signature of its class. But, if all targets are of the same class, a target cannot be easily classified by its signature and, in the absence of reliable classification information, the proposed methods will behave like the naive nearest neighbor tracker. Our algorithm can complement the identity management algorithms when tracking targets within the same class or when reliable classification information is not available. A distributed particle filtering algorithm for sensor networks is presented in [80] and used to track a single maneuvering target, assuming the availability of super nodes and a hierarchical topology similar to ours. The paper assumes the availability of sensors which can measure an angle and distance to a target. But sensors with such capabilities are costly and they are not suitable for a large sensor network with inexpensive sensor nodes. The most widely used and realistic sensor model is based on the signal strength and this is the model we use in this work.

CHAPTER 3. WIRELESS SENSOR NETWORK SECURITY

One of the key issues rising from switching to wireless communication lies in security; while an air gap is among the most effective security measures in wired networks, wireless communication is not as easy to isolate from attack. The security issues in mobile ad hoc networks (MANETs) are more challenging than those in traditional wired computer networks and the Internet. Providing security in sensor networks is even more difficult than in MANETs due to the resource limitations of sensor nodes and security concerns remain a serious impediment to widespread adoption of these WSNs [12].

3.1. Wired & Wireless Networks

Wireless networks have offered attractive flexibility to both network operators and users. Ubiquitous network coverage, for both local and wide areas, is provided without the cost of deploying and maintaining the wires. This fact is extremely useful in several situations like network deployment in difficult to wire areas, prohibition of cable deployment and deployment of a temporary network. Mobility support is another salient feature of wireless networks. Though there are varieties of challenges in sensor networks, here we focus on different security issues and possible remedies of those. Though security is a very important issue in WSN, due to various resource limitations and the salient features of a WSN, the security design for such networks is significantly challenging. In this chapter, we explore the security issues and challenges for next generation WSNs and discuss the crucial parameters that require extensive investigations.. Although most, if not all, security threats against the TCP/IP stack in a wired network are equally applicable to an IP-based wireless network, the latter possesses a number of additional vulnerabilities; wireless medium unreliability, spectrum use, power management, security,

limited bandwidth, system complexity, routing, Interfacing with wired networks and health concern make it more challenging to secure [13].

3.2. Operation

A WSN is a large network of resource-constrained sensor nodes with multiple preset functions, such as sensing and processing, to fulfill different application objectives. The major elements of WSN are the sensor nodes and the base stations (BSs). In fact, they can be abstracted as the "sensing cells" and the "brain" of the network, respectively. Usually, sensor nodes are deployed in a designated area by an authority and then, automatically form a network through wireless communications. Sensor nodes of homogeneous or heterogeneous type can be deployed randomly or at pre-determined locations using a deterministic scheme. Sensor nodes are static most of the time, whereas mobile nodes can be deployed according to application requirements. One or several, static or mobile [15] BSs are deployed together with the network. Sensor nodes keep monitoring the network area after being deployed. After an event of interest occurs, one of the surrounding sensor nodes can detect it, generate a report, and transmit the report to a BS through multi-hop wireless links. Collaboration can be carried out if multiple surrounding nodes detect the same event. In this case, one of them generates a final report after collaborating with the other nodes. The BS can process the report and then forward it through either high-quality wireless or wired links to the external world for further processing. The WSN authority can send commands or queries to a BS, which spreads those commands or queries into the network. Hence, a BS acts as a gateway between the WSN and the external world. An example is illustrated in Figure 7.

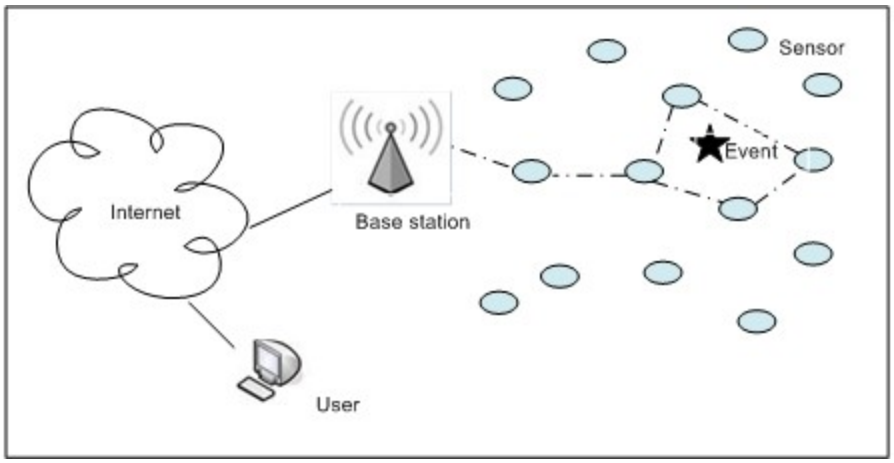


Figure 7. A wireless sensor network structure

3.3. Constraints

Individual sensor nodes in a WSN have the inherent limitations in resources, which make the design of security procedures more complicated. A typical sensor node processor is of 4-8 MHz, having 4KB of RAM, 128KB flash and ideally 916 MHz of radio frequency. Each of these limitations is due in part to the two greatest constraints — limited energy and physical size.

Energy: Sensor nodes typically have a small form factor with a limited amount of battery power. Therefore, protocols designed for sensor networks should utilize only a few control messages. Sensor transducer, communication among sensor nodes and microprocessor computation consumes energy in sensor nodes, in that communication consumes more energy in WSNs. Any message expansion caused by security mechanisms comes at a significant cost. Further, higher security levels in WSNs usually correspond to more energy consumption for cryptographic functions.

Memory: Sensor nodes usually have a small amount of memory. Hence, sensor network protocols should not require the storage of a large amount of information at the sensor node.

There is usually not enough space to run complicated algorithms after loading OS and application code. This makes it impractical to use the majority of current security algorithms.

Scalability: The number of sensor nodes in a sensor network can be in the order of hundreds or even thousands. Hence, protocols designed for sensor networks should be highly scalable.

Transmission Range: The communication range of sensor nodes is limited both technically and by the need to conserve energy.

Fault Tolerance: Sensor nodes are prone to failure. This may be due to a variety of reasons. Loss of battery power may lead to failure of the sensor nodes. Thus, protocols designers should build fault tolerance into their algorithms for improving the utility of sensor networks.

Self-Organization: Sensor nodes are often air-dropped in hostile or harmful environments. It is not possible for humans to reach these sensor nodes. Besides, it is not possible for humans to repair each sensor node, as often the number of sensor nodes is quite large. Hence, self- organization of sensor nodes to form a connected network is an essential requirement.

3.4. Security

3.4.1. Reasons for Needing Security

MANETs and sensor networks have many applications in military, homeland security, and other areas. In that many sensor networks have mission-critical tasks. Security is critical for such networks deployed in hostile environments, and security concerns remain a serious impediment to widespread adoption of these wireless networks. The security issues in MANETs are more challenging than those in traditional wired computer networks and the Internet. Providing security in sensor networks is even more difficult than in MANETs due to the resource

limitations of sensor nodes. Most sensor networks actively monitor their surroundings, and it is often easy to deduce information other than the data monitored. Such unwanted information leakage often results in privacy breaches of the people in the environment. Moreover, the wireless communication employed by sensor networks facilitates eavesdropping and packet injection by an adversary. The combination of these factors demands security for sensor networks at design time to ensure operation safety, secrecy of sensitive data, and privacy for people in sensor environments [14]. Significant efforts and research have been undertaken to enhance security levels of wireless networks. Currently, there are three levels of security available in wireless networking environments [16].

3.4.2. Factors Causing Complex Security in WSNs

Security in sensor networks is complicated by the constrained capabilities of sensor node hardware and the properties of the deployment [14] and [15].

(1)The overall cost of the WSN should be as low as possible.

(2)Sensor nodes use wireless communication, which is particularly easy to eavesdrop on.

(3)Similarly, an attacker can easily inject malicious messages into the wireless network.

(4)Advanced anti-jamming techniques such as frequency- hopping spread spectrum and physical tamper proofing of nodes are generally impossible in a sensor network due to the requirements of greater design complexity and higher energy consumption.

(5)The use of radio transmission, along with the constraints of small size, low cost, and limited energy, make WSNs more susceptible to denial-of-service attacks.

(6)Ad-hoc networking topology of WSN facilitates attackers for different types of link attacks ranging from passive eavesdropping to active interfering. Attacks on a WSN can come

from all directions and target at any node leading to leaking of secret information, interfering with message, impersonating nodes, etc.

(7) Security also needs to scale to large-scale deployments. Most current standard security protocols were designed for two-party settings and do not scale to a large number of participants.

(8) There is a conflicting interest between minimization of resource consumption and maximization of security level. A better solution actually gives a good compromise between these two.

(9) Since sensor nodes usually have severely constrained, asymmetric cryptography is often too expensive for many applications. Thus, a promising approach is to use more efficient symmetric cryptographic alternatives.

3.5. Taxonomy of Attacks

Wireless networks are vulnerable to security attacks due to the broadcast nature of the transmission medium. Furthermore, WSNs have an additional vulnerability because nodes are often placed in a hostile or dangerous environment where they are not physically protected. For a large-scale sensor network, it is impractical to monitor and protect each individual sensor from physical or logical attack. Attackers may devise different types of security threats to make the WSN system unstable. Here, in this section, we present a layer-based classification of WSN security threats and also based on the capability of the attacker and defenses.

3.5.1. Attacks Based on the Capability of the Attacker

Outsider versus insider (Node Compromise) attacks - Outside attacks [14], [17] are defined as attacks from nodes, which do not belong to a WSN; insider attacks occur when legitimate nodes of a WSN behave in unintended or unauthorized ways. To overcome these

attacks [14], we require robustness against Outsider Attacks, Resilience to Insider Attacks, Graceful Degradation with Respect to Node Compromise and Realistic Levels of Security.

Passive versus active attacks: Passive attacks include eavesdropping on or monitoring packets exchanged within a WSN; active attacks involve some modifications of the data steam or the creation of a false stream.

Mote-class versus laptop-class attacks: In mote-class attacks, an adversary attacks a WSN by using a few nodes with similar capabilities to the network nodes; in laptop-class attacks, an adversary can use more powerful devices (e.g., a laptop) to attack a WSN. These devices have greater transmission range, processing power, and energy reserves than the network nodes.

3.5.2. Attacks on Information in Transition

In a sensor network, sensors monitor the changes of specific parameters or values and report to the sink according to the requirement. While sending the report, the information in transit may be attacked to provide wrong information to the base stations or sinks. The authors [17] show some types of attack on information in transition. Interruption Communication link in sensor networks becomes lost or unavailable. This operation threatens service availability. The main purpose is to launch denial-of-service (DoS) attacks. From the layer-specific perspective, this is aimed at all layers. Interception Sensor network has been compromised by an adversary where the attacker gains unauthorized access to sensor node or data in it. Example of this type of attack is node capture attacks. This threatens message confidentiality. The main purpose is to eavesdrop on the information carried in the messages. From the layer-specific perspective, this operation is usually aimed at the application layer. Modification Unauthorized party not only accesses the data but also tampers with it. This threatens message integrity. The main purpose is to confuse or mislead the parties involved in the communication protocol. This is usually aimed

at the network layer and the application layer, because of the richer semantics of these layers. Fabrication An adversary injects false data and compromises the trustworthiness of information. This threatens message authenticity. The main purpose is to confuse or mislead the parties involved in the communication protocol. This operation can also facilitate DoS attacks, by flooding the network. Replaying existing messages this operation threatens message freshness. The main purpose of this operation is to confuse or mislead the parties involved in the communication protocol that is not time-aware.

3.6. Issues with High-Level Security Mechanisms

3.6.1. Cryptography and Key Management

To achieve security in WSNs, it is important to be able to perform various cryptographic operations, including encryption, authentication, and so on. Selecting the appropriate cryptography method for sensor nodes is fundamental to providing security services in WSNs. However, the decision depends on the computation and communication capability of the sensor nodes. Since sensor nodes usually have severely constrained resources, asymmetric cryptography is often too expensive for many applications. Thus, a promising approach is to use more efficient symmetric cryptographic alternatives. However, symmetric cryptography is not as versatile as public key cryptographic techniques, which complicates the design of secure applications. Applying any encryption scheme requires transmission of extra bits, hence extra processing, memory and battery power, which are very important resources for the sensors' longevity. Applying the security mechanisms such as encryption could also increase delay, jitter and packet loss in WSNs.

3.6.2. Intrusion Detection

The problem of intrusion detection is very important in the case of WSNs. Traditional approaches which do an anomaly analysis of the network at a few concentration points, are expensive in terms of network's memory and energy consumption. So there is a need for decentralized intrusion detection [18]. Intrusion detection in WSNs is still largely open to research. Key research issues are [17]:

(1) Due to the constraints in WSNs, intrusion detection has many aspects that are not of concern in other network types.

(2) The problem of intrusion detection needs to be well defined in WSNs.

(3) The proposed IDS protocols in literature focus on filtering injected false information only.

(4) These protocols need to be improved so as to address scalability issues.

(5) It is very difficult to integrate intrusion detection techniques into a uniform hardware platform due to cost and implementation considerations [15].

CHAPTER 4. DANGER THEORY INSPIRED SECURITY APPROACHES

In this chapter, we present proposed danger theory inspired security approaches. These approaches are primarily used to detect two types of attacks conducted by malicious insiders: active flooding attack and passive packet dropping attack. An effective approach on detecting malicious insiders should have a low false positive rate, the ratio representing the number of normal nodes mistakenly detected as malicious ones to the total number of normal nodes. The characteristics of Danger Theory, one of models of Artificial Immune System (AIS), make it as an appropriate choice on accurately detecting abnormal events.

4.1. AIS Background

An Artificial Immune System uses an analogy with the natural Immune System (IS) of vertebrates. As an approximation, the immune system can be described with the self-nonself model. The immune system is thought to be able to classify cells that are present in the body as self and non-self cells. The immune system is made of two distinct sets of components: the innate IS, and the adaptive IS. The innate immune system is hard-wired to detect (and destroy) nonself cells that contain, or do not contain, specific patterns on their surface. The adaptive immune system is more complex. It produces a large number of randomly created detectors. A negative selection mechanism eliminates detectors that match any cell present in a protected environment (bone marrow and the thymus) where only self cells are assumed to be present. Non-eliminated detectors become naive detectors; they die after some time, unless they match something (assumed to be a pathogen), in which case they become memory cells. Further, detectors that do match a pathogen are quickly multiplied (clonal selection.); this is used to accelerate the response to further attacks. Also, since the clones are not exact replicates, (they are

mutated, and mutation rate is an increasing function of affinity between detectors and the pathogen) this provides a more focused response to the pathogen (affinity maturation). This also provides adaptation to a changing non-self environment. The self-nonself model is only a very crude approximation of the adaptive IS. Another important aspect is the danger signal model [19]. With this model, matching by the innate or adaptive mechanism is not sufficient to cause detection; an additional danger signal is required. The danger signal is for example generated by a cell that dies before being old. The danger signal model better explains how the IS adapts not only to a changing non-self, but also to some changes in self. There are many more aspects to the IS, some of which are not yet fully understood.

Many algorithms in the field of AIS have been developed. One of the major algorithms developed within AIS is the negative selection algorithm, first proposed by Forrest et al. [20] and then subsequently developed over the years [21, 22]. This paper investigates the real-valued negative selection algorithm with variable-sized detectors [23] and its applicability to network intrusion traffic. The negative selection algorithm is often cited for its potential use in intrusion detection problems due to its ability to generate a set of detectors from a single class of data (usually the normal network traffic), that is capable of identifying possible intrusions. However, there remains little work in the literature regarding the application of the negative selection algorithm with variable-sized detectors to network intrusion detection.

4.2. Danger Theory

4.2.1. Introduction

We now examine the biological basis for the self-nonself metaphor, and the alternative Danger Theory (DT) hypothesis. The human immune system (HIS) is commonly thought to work at two levels: innate immunity including external barriers (skin, mucus), and the acquired

or adaptive immune system [36]. As part of the latter level, B-Lymphocytes secrete specific antibodies that recognize and react to stimuli. It is this matching between antibodies and antigens that lies at the heart of the HIS and most AIS implementations. The central tenet of the immune system is the ability to respond to foreign invaders or 'antigens' while not reacting to 'self' molecules. In order to undertake this role the immune system needs to be able to discern differences between foreign, and possibly pathogenic, invaders and non-foreign molecules. It is currently believed that this occurs through the utilization of the Major Histo-compatibility Complex (MHC). This complex is unique to each individual and therefore provides a marker of 'self'. In addition, the cells within the immune system are matured by becoming tolerated to self-molecules. Together, through the MHC and tolerance, the HIS is able to recognize foreign invaders and send the requisite signals to the key effecter cells involved with the immune response. The DT debates this and argues that there must be discrimination happening that goes beyond the self-nonself distinction because the HIS only discriminates 'some self' from 'some nonself.' It could therefore be proposed that it is not the 'foreignness' of the invaders that is important for immune recognition, but the relative 'danger' of these invaders. This theory was first proposed in 1994 [37] to explain current anomalies in our understanding of how the immune system recognizes foreign invaders. For instance, there is no immune reaction to foreign bacteria in the gut or to food. Conversely, some auto reactive processes exist, e.g. against self-molecules expressed by stressed cells. Furthermore, the human body (self) changes over its lifetime. Therefore, why do defenses against nonself learned early in life not become auto-reactive later? The DT suggests that foreign invaders, which are dangerous, will induce the generation of cellular molecules (danger signals) by initiating cellular stress or cell death [38]. These molecules are recognized by APCs, critical cells in the initiation of an immune response, which

become activated leading to protective immune interactions. Overall there are two classes of danger signal; those which are generated endogenously i.e., by the body itself, and exogenous signals which are derived from invading organisms e.g., bacteria [34]. Evidence is accruing as to the existence of myriad endogenous danger signals including cell receptors, intracellular molecules and cytokines. A commonality is their ability to activate APCs and thus drive an immune response.

We believe that the DT will provide a more suitable biological metaphor for IDS than the traditional self-nonself viewpoint, regardless whether the theory holds for the HIS, something that is currently hotly debated amongst immunologists ([35], [40]). In particular, the DT provides a way of grounding the response, i.e., linking it directly to the attacker and it removes the necessity to map self or nonself [33]. In our model, self-nonself discrimination will still be useful but it is no longer essential. This is because nonself no longer causes a response. Instead, danger signals will trigger a reaction. Actually, the response is more complicated than this, since it is believed that the APCs integrate necrotic ('danger') and apoptotic ('safe') signals in order to regulate the immune response. We intend to examine this integrative activity experimentally, which should provide useful inspiration for IDS.

In 2003, Aickelin et al outlined a project describing the application of a novel immunological theory, the Danger Theory to intrusion detection systems [5]. The authors of this work suggested that the Danger Theory encompassed pathogenic detection, where the basis for discrimination was not centered around 'self' or 'non-self', but to the presence or absence of danger signals. The paper described how danger signals are released from the body's own tissue cells as a result of necrotic cell death, triggered by an invading pathogen. The immune system was thought to be sensitive to changes in concentration of danger signals and hence an

appropriate response is generated. Aickelin et al propose that by differentiating between the chaotic process of necrotic cell death and the safe signals derived from regulated apoptotic cell death, pathogenic agents can be detected within an artificial immune system context. Currently, the majority of artificial immune systems (AIS) encompass two different types of immune inspired algorithms, namely negative selection (T-cell based), and clonal selection with somatic hyper-mutation (B-cell based). Exceptions to this include [42], where defined patterns of misbehavior were used to create danger signals within mobile ad-hoc networks. Danger signals are used in [41] to define the context for collaborative filtering. Implementations including Danger Theory, so far, have monitored danger signals directly and have not taken into account any of the cells responsible for signal detection. It is thought that danger signals are detected and processed through 'professional' antigen presenting cells known as dendritic cells (DCs). DCs are viewed as one of the major control mechanisms of the immune system, influencing and orchestrating T-cell responses, in addition to acting as a vital interface between the innate (initial detection) and adaptive (effecter response) immune systems. DCs are responsible for some of the initial pathogenic recognition process, sampling the environment and differentiating depending on the concentration of signals, or perceived misbehavior, in the host tissue cells. Strong parallels can be drawn from this process to the goal of successful anomaly detection. Current anomaly detection systems frequently rely on profiling 'normal' user behavior during a training period. Any subsequent observed behavior that does not match the normal profile (often based on a simple distance metric) is classed as anomalous. At this point an 'alert' is generated. However, these systems can have problems with high levels of false positive errors, as behavior of users on a system changes over a period of time. Anomaly detection systems remain a high research priority as their inherent properties allow for the detection of novel instances, which

could not be detected using a signature based approach. AIS featuring negative selection algorithms have been tried and tested for the purpose of anomaly detection [43]. They produced promising results, but were tarnished by issues surrounding false positives and scalability [44]. Some moderately successful non-AIS systems have been implemented, often involving adaptive sampling [45] and adaptive alert threshold modification. The aim of this research is to understand the Danger Theory and its implications and to be able to derive an anomaly detection system.

4.2.2. Nervous System

Muscles and bones are what move your body, but how does your body know when to move? The nervous system tells your body. Your nervous system is made up of the brain, spinal cord, nerves, and sense organs. Without this system you would not be able to speak, think, taste, hear, or see. The nervous system knows exactly what is going on both inside and outside your body. It is able to make sense of all the information it receives and respond to it.

The nervous system would not work without nerve cells called neurons. Neurons pass messages throughout your body. Each neuron (Figure 8) has a cell body with short branches sticking out on one side and a long branch on the other side. The short branches are called dendrites. Dendrites get messages from other neurons and give them to the cell body. The long branch, called the axon, moves messages away from that neuron to other nerve cells.

When the dendrite of a neuron gets a message, the chemicals in the neuron change. This change causes an impulse, or message, to move across the neuron. The impulse moves from the dendrite to the cell body. It leaves the neuron through the axon. The message then gets picked up by the dendrites of the next neuron and causes an impulse in that neuron. This is how messages move from one neuron to the next.

Most of these impulses move along neurons to your brain. Your brain controls almost everything you experience. When your brain gets a message from your nervous system, it makes sense of the message and tells your body how to react.

This message is sent through the spinal cord. The spinal cord is a long bundle of nerves that runs down your back. Some neurons of the spinal cord bring messages to the brain. Others carry messages away.

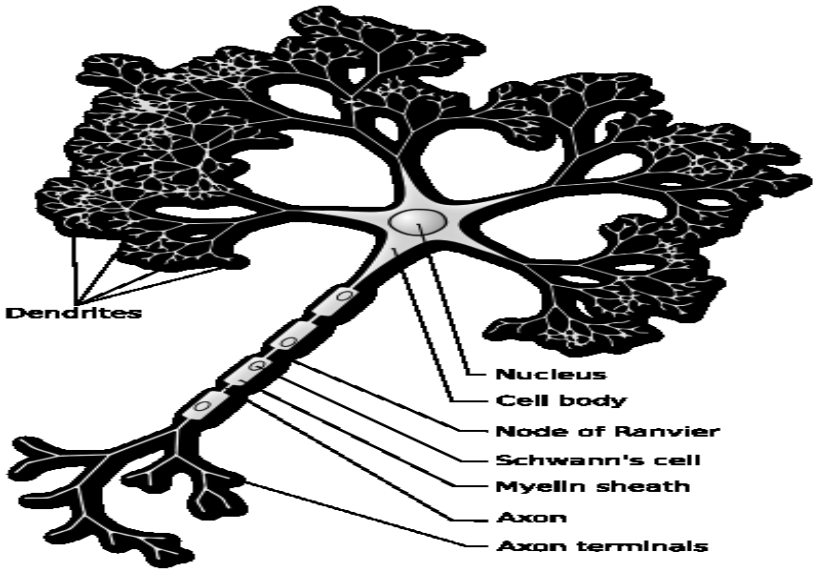


Figure 8. A neuron diagram [74]

The Danger Theory, proposed by Polly Matzinger in 1994 [46], also emphasizes the crucial role of the innate immune system for guiding the adaptive immune responses. However, unlike detecting exogenous signals, the Danger Theory rests on the detection of endogenous signals. Endogenous danger signals arise as a result of damage or stress to the tissue cells. The crucial point of the Danger Theory is that the only pathogens detected are the ones that induce necrosis and cause actual damage to the host tissue. The damage can be caused by invading micro-organisms or through defects in the host tissue or innate immune cells. Irrespective of the

cause, the danger signals released are always the same. These signals are thought to be derived from the internal contents of the cell [47] inclusive of heat shock proteins, fragmented DNA and uric acid. It is proposed that the exposure of antigen presenting cells to danger signals modulates the cells' behavior, ultimately leading to the activation of naive T-cells in the lymph nodes. Alternatively, the absence of danger signals and the presence of cytokines released as a result of apoptosis can lead to antigen presentation in a different context, deleting or energizing a matching T-cell [48]. The Danger Theory suggests that the tissue is in control of the immune response. In [49] it is suggested that DCs have the capability to combine signals from both endogenous and exogenous sources, and respond appropriately. Different combinations of input signals can ultimately lead to the differentiation and activation of T-cells. Both theories have implications for the function of DCs.

4.2.3. Dendritic Cells

Dendritic cells (DCs) are white blood cells, which have the capability to act in two different roles - as macrophages in peripheral tissues and organs and as a vehicle for antigen presentation within the secondary lymphoid organs. DCs can be sub-categorized dependent on their location within the body. For the purpose of this investigation and the subsequent algorithm, dermal or tissue resident DCs have been examined. Essentially, the DCs' function is to collect antigen from pathogens and host cells in tissues, and to present multiple antigen samples to naive T-cells in the lymph node. DCs exist in a number of different states of maturity, dependent on the type of environmental signals present in the surrounding fluid. They can exist in immature, semi-mature or mature forms.

Immature DCs: Immature DCs (iDCs) are cells found in their initial maturation state. They reside in the tissue where their primary function is to collect and remove debris from the

interstitial fluid. The ingested material is then processed by the cell. It is either metabolized for use by the cell, returned to the environment, or is repackaged for presentation to another immune cell. At this point the matter can be termed antigen, and could be a 'self' molecule or something foreign. The representation of antigenic material is performed by complexion the antigen with another molecule namely the MHC molecule family, necessary for binding to T-cell receptors. In order to present antigen to T-cells, DC needs sufficient antigen presented with MHC. However, the expression of inflammatory cytokines is needed in order to activate T-cells. Therefore a T-cell encounter with an iDC results in the deactivation of the T-cell. Differentiation of iDCs occurs in response to the receipt of various signals. This leads to full or partial maturation depending on the combination of signals received.

Semi-Mature DCs: During the antigen collection process, iDCs can experience other environmental conditions. This can affect the end-stage differentiation of a DC. These different conditions can give rise to semi-mature DCs (smDCs). The signals responsible for producing smDCs are also generated by the tissue - endogenous signals. During the process of apoptosis, a number of proteins are actively up-regulated and secreted by the dying cell. The release of TNF-_ (tumor necrosis factor) from apoptosing cells is thought to be one candidate responsible for creating semi-mature DCs [50]. As a result of exposure to apoptotic cytokines, an iDC also undergoes migration to the lymph node. Costimulatory molecules are up-regulated by a small yet significant amount and, after migration to the lymph node, the cell can present antigen to any matching T-cell. However, smDCs do not produce any great amount of pro-inflammatory cytokines, necessary for promoting activation of T-cells. Instead, smDCs can produce small quantities of IL-10 (anti-inflammatory cytokine), which acts to suppress matching T-cells.

Mature DCs: Due to the low levels of inflammatory cytokines expressed by iDCs, they are not able to activate T-cells on contact. In order to present antigen and activate T-cells, the increased expression (or up-regulation) of a number of proteins and cytokines is necessary. DCs which have the ability to activate naive T-cells are termed mature DCs (mDCs). For an iDC to differentiate and become a mDC, the iDC has to be exposed to a certain number of signals. This includes activation of toll-like receptors through exposure to both the exogenous and endogenous signals (previously described). On exposure to various combinations of these signals, the DC up-regulates a number of molecules vital for stimulating a T-cell response. Perhaps most importantly, it up-regulates a number of costimulatory molecules, pro-inflammatory cytokines (namely IL-12), and migrates from the tissue to the local draining lymph node. During this migration period, the iDC changes morphologically too. Instead of being compact, the DC develops finger-like projections - characterizing it as an mDC. The projections not only make it distinguishable from iDCs, but also increase the surface area of the cell, allowing it to present a greater quantity of antigen.

In brief, DCs can perform a number of functions, related to their state of maturation. Modulation between these states is facilitated by the release of endogenous and exogenous signals, produced by pathogens and the tissue itself. The state of maturity of a DC influences the response by T-cells, either immunogenic or tolerogenic, to specific presented antigen. Immature DCs reside in the tissue where they collect antigenic material and are exposed to exogenous and endogenous signals. Based on the combinations of signals, mature or semi-mature DCs are generated. Mature DCs have an activating effect while semi-mature DCs have a suppressive effect. The different cytokine output by the respective cells differ sufficiently to provide the

context for antigen presentation. In the following section this information is utilized to derive a signal processor based on the explored functionality of the DCs.

4.3. Preventing Flooding Attack

To identify malicious insiders that implement a flooding attack, we propose a new DCA. A DCA is a problem-solving method that is based on the behavior of DCs. An innovation of our method is that the monitoring period is dynamically adjusted by the base station in accordance with the current network status. Thus, we refer to the method as a Dynamic Dendritic Cell Algorithm (DDCA) [94]. Algorithm 1 illustrates the structure of this algorithm. This proposed DDCA is capable to recognize the following types of attacks in a WSN: the transmitting of modified messages; the reporting of messages with an abnormal frequency; and the reporting of fake messages. When a sensor node sends out a message, we assume that all the neighbor nodes of the sender can receive and interpret the message. Each sensor node has the ability to discern if a neighbor node is transmitting a suspicious message (such as a modified or false message). A sensor node flags a harmful intruder if the number of suspicious messages exceeds a pre-specified threshold value. Sensor nodes are typically battery powered and have limited energy that must be used efficiently. The energy consumption rate for communication greatly exceeds that for sensing. Thereby, the useful lifetime of a WSN is largely governed by the management of the transmitting and receiving of messages. Ignoring messages rather than communicating with harmful intruders conserves energy.

Algorithm 1: The dynamic dendritic cell algorithm (DDCA)

$\boldsymbol{m_n}$: New message

$\boldsymbol{T_1}$: The maximum time slice calculating abnormal frequency of reporting message

$\boldsymbol{T_2}$: The maximum time slice supervising a suspicious intruder

$\boldsymbol{HI}$, $\boldsymbol{SHI}$: Harmful Intruder, Semi-Harmful Intruder

$\boldsymbol{t_0, t_1, t_2, t_3}$: Thresholds for reporting message frequency during T_1, reporting message frequency during T_2, reporting false message rate during T_2, reporting changed message rate during T_2, respectively

(**Note**: to tolerate casual high frequency of reporting message, let $T_2 > T_1$)

bool **checkDangerMessage** (m_n)

1. ***if*** (m_n came from a HI) ***return*** true
2. ***if*** (m_n came from a SHI)
3. ***if*** (checkHarmfulIntruder(m_n) == true) ***return*** true
4. ***if*** (checkAbnormalMessage(m_n) == true) update SHI list
5. ***return*** false

bool **checkAbnormalMessage** (m_n)

1. ***if*** (m_n is a changed original message) ***return*** true
2. ***if*** (m_n is a fake message) ***return*** true
3. ***if*** (the frequency of the sender reporting message $> t_0$) ***return*** true
4. ***return*** false

bool **CheckHarmfulIntruder** (m_n)

1. Update the history message record of the sender
2. Estimate the frequency and percentage of the sender
3. ***if*** ((t_1 reached) or (t_2 reached) or (t_3 reached))
4. Clear the history record of the sender; Move the sender from SHI list to HI list
5. ***return*** true
6. ***if*** (T_2 reached) Remove the sender from SHI list
7. ***return*** false

This algorithm consists of three sub-functions: checking for dangerous messages, abnormal messages, and harmful intruders. When a sensor node receives a new message, the algorithm first checks the harmful intruder (*HI*) list. If the sender of this message is an identified harmful intruder, the message is considered to be a dangerous message. If the sender is an identified suspicious intruder that exists in the semi-harmful intruder (*SHI*) list, the algorithm runs the function for identifying harmful intruder to decide if this sender is dangerous enough to be considered harmful. Finally, the algorithm runs the function of checking for abnormal messages and possibly puts the sender on the *SHI* list. This algorithm will only consume limited energy of a sensor node because of the simple calculation in each sub-function. The dynamic adjustment of parameters increases the flexibility and accuracy of the DC-inspired algorithm in detecting dangerous messages.

During the process of identifying malicious insiders, a normal node maintains a danger level (d_l) for each sensed suspicious insider. This danger level is useful for the proposed monitoring scheme, which is presented in the next section.

4.4. Preventing Packet Dropping Attack

4.4.1. Bayesian Game Based Monitoring Scheme

To prevent packet dropping attack, we first need to label the malicious insiders. A malicious insider may implement gray hole attack mentioned in [3]. This type of attacker is called gray hole. Gray holes usually only transmit routing packets and drop data packets. In this work, we consider a more general gray hole attack, namely light-gray hole attack (LGHA), in wireless sensor networks. A malicious node selectively forwards packets instead of dropping all packets. This type of attack is harder to be detected than dropping all packets attack.

We consider a two-player static Bayesian game. One player is a potential attacker (a malicious insider, denoted by MI). The other player is a defender (a normal insider, denoted by NI). Player MI has private information about his type, which is either regular, denoted by $\theta_{\mathrm{i}} = 0$, or malicious, denoted by $\theta_i = 1$. In other words, the maliciousness of player i is unknown to the defender NI. Defender NI is of regular type denoted by $\theta_j = 0$. The type of defender NI is common knowledge to the two players.

The malicious type of player MI has two pure strategies: Attack (drop packets) and Non-attack (transmit packets). The regular type of player i has only one pure strategy: Non-attack. Defender NI has two pure strategies: monitor (query the destination) and Non-monitor (not query the destination). The two players choose their strategies simultaneously at the beginning of the game, assuming common knowledge about the game (costs). Here the belief μ_0 is the danger level calculated by the proposed DDCA during the process of identifying malicious insiders. It dynamically changes throughout the detecting process. Therefore, its value has high credibility.

A normal node can't predict when a malicious node decides to drop received packets. But the normal node can find if a packet is dropped by probing (querying) to the other normal node

right next to this malicious node on the route. This detecting method is also mentioned in [27]. The main difference between our probing method and the one in [27] is that we adopt Bayesian game based strategies to decide when to do the probe instead of the periodic probe used in [27]. As malicious nodes randomly or strategically drop received packets, the periodically probing method is inefficient. A shorter monitoring period has higher routing overhead, and longer monitoring period has lower detection rate. Therefore we adopt Bayesian game theory based strategies to decide the proper monitoring time. In this game, the two players are a normal node and a malicious node. Table 1 shows the payoffs for different combinations of strategies adopted by the two players. The symbol ω represents a loss of security whose value is equivalent to a degree of damage such as loss of reputation, loss of data integrity, or cost of damage; the symbol α represents the detection rate; the symbol β represents the false positive rate, and $\alpha, \beta \in [0, 1]$; the symbol c_a represents the cost of attacking; and the symbol c_m represents the cost of monitoring.

In the case that player NI adopts the strategy monitor and player MI adopts the strategy non-attack, the payoff of player NI is $(-\beta\omega - c_m)$, which is the sum of the cost of false decision and the cost of monitoring. Since player MI doesn't have any loss, its payoff is zero. If the strategy combination is non-monitor and attack, the payoff of player NI is $(-\omega)$, the cost of damage caused by the missed attack. And the payoff of player MI is $(\omega - c_a)$, which is the gain from the loss of player NI excluding the cost on the attack. Similarly, if the strategy combination is monitor and attack, the payoff of play NI is the sum of the gain $(\alpha\omega)$ caused by the right decision, the cost $(1-\alpha)\omega$ caused by the false negative rate, and the cost on monitoring (c_m), i.e., totally $((2\alpha - 1)\omega - c_m)$. And the payoff of player MI is $((1-2\alpha)\omega - c_a)$, the gain from

the loss of player NI excluding the cost on the attack. In the last case that nobody adopts monitor or attack, their payoffs are both zero.

Table 1. Strategic Form of Static Bayesian Game

		A Malicious Insider (MI)	
		Non- Attack	Attack
A Normal Insider (NI)	Monitor	$-(\beta\omega + c_m)$, 0	$(2\alpha - 1)\omega - c_m$, $(1 - 2\alpha)\omega - c_a$
	Non- Monitor	0, 0	$-\omega$, $\omega - c_a$

The objective of the two players is to maximize their own payoff with incomplete information, i.e., without knowing another player's decision before making their own decision. How can the players make their best strategies which can help them achieve maximal payoff whatever strategies their opponent use? Bayesian game theory makes it possible to solve this problem. According to this theory, two players in a game can achieve their respective maximal payoff under Bayesian Nash Equilibrium (BNE) [52] situation, in which everybody is doing the best they can, given what everybody else is doing. So, we next try to find this equilibrium for the two players.

We use symbol μ_0 to represent the danger level of player MI in this game, which can be obtained from the proposed immune-inspired algorithm. This value is a common prior for both players, i.e., player MI knows defender NI's belief of μ_0 .

If player MI plays his strategy attack, the expected payoff of defender NI playing his strategy monitor is

$$E_{NI}(monitor) = \mu_0\big((2\alpha - 1)\omega - c_m\big) + (1 - \mu_0)(-\beta\omega - c_m),$$

and its payoff playing the strategy not monitor is

$$E_{NI}(non - monitor) = -\mu_0\omega .$$

If $E_{NI}(monitor) > E_{NI}(non - monitor)$, or $\mu_0 > \frac{(1+\beta)\omega + c_m}{(2\alpha+\beta-1)\omega}$, strategy monitor seems to be the best strategy for player NI because of its greater payoff. However, when player NI play monitor, player MI will change its strategy to non-attack from attack. Hence, (μ_0, monitor, attack) is not a BNE when $\mu_0 > \frac{(1+\beta)\omega + c_m}{(2\alpha+\beta-1)\omega}$. Since using pure-strategy can't achieve the BNE, we need to adopt a mix-strategy for this case, which will be explained later. If $\mu_0 < \frac{(1+\beta)\omega + c_m}{(2\alpha+\beta-1)\omega}$, player NI prefers to use strategy non-monitor. Player MI will keep its strategy attack. Hence (μ_0 , non-monitor, attack) is a BNE when $\mu_0 < \frac{(1+\beta)\omega + c_m}{(2\alpha+\beta-1)\omega}$.

If player MI plays its strategy non-attack, strategy non-monitor seems to be the best strategy for player NI whatever the value of μ_0 is. But, player MI would change its strategy to attack if player NI plays non-monitor. Hence, the final stable state will be (μ_0, non-monitor, attack), which is a BNE as we analyzed before.

Now, we analyze how the players adopt a mix-strategy to maximize their payoff when $\mu_0 > \frac{(1+\beta)\omega + c_m}{(2\alpha+\beta-1)\omega}$. Let p be the probability with which player MI plays attack, and q be the

probability with which player NI plays monitor. The new expected payoff of player NI playing monitor is

$$E_{NI}^{*}(monitor) = p\mu_0\big((2\alpha-1)\omega - c_m\big) + (1-p\mu_0)(-\beta\omega - c_m),$$

and its new expected payoff playing non-monitor is

$$E_{NI}^{*}(non-monitor) = -p\mu_0\omega .$$

When $E_{NI}^{*}(monitor) = E_{NI}^{*}(non-monitor)$, we get $p^{*} = \frac{\beta\omega + c_m}{(2\alpha+\beta)\omega\mu_0}$. This is the probability that player MI plays equilibrium strategy attack. Similarly, we can calculate the probability that player NI plays equilibrium strategy monitor. Let $E_{MI}^{*}(attack) = E_{MI}^{*}(non-attack)$, i.e.,

$$q\mu_0\big((1-2\alpha)\omega - c_a\big) + (1-q\mu_0)(\omega - c_a) = q\mu_0 * 0 + (1-q\mu_0) * 0 = 0.$$

We get $q^{*} = \frac{\omega - c_a}{2\alpha\mu_0\omega}$, which is the best monitoring strategy for the player NI based on the value of μ_0.

According to the above analysis, both players can choose proper strategies under different situations. Here we are concerned that player NI chooses proper strategies for monitoring attack. Besides using μ_0 as the belief, player NI can also refer to the maintained reputation table of its neighbor nodes to make monitoring decision.

Each normal node maintains a neighbor table that lists all neighbors and records if it has data packets communication with them recently. For example, node x has some neighbors, and for each neighbor i:

$$< x, i > \begin{cases} 1 & If\ x\ sends\ data\ packets\ to\ i\ recently \\ 0 & Otherwise \end{cases}$$

$$< i, x > \begin{cases} 1 & If\ x\ receives\ data\ packets\ from\ i\ recently \\ 0 & Otherwise \end{cases}.$$

It only needs small storage to save this information, but it is very useful on evaluating each neighbor's reputation.

As the packet dropping rate exceeds the predefined threshold, which is set to be slightly above the normal packet dropping rate, the malicious node is recognized as a real attacker. The normal node also put the new detected malicious node in its malicious node list.

4.4.2. Short-and-Safe Routing Protocol

After malicious insiders are recognized, the direct preventive measure is not using the routes that contain malicious insiders to transmit packets. However, as we assumed, malicious nodes conducting a packet dropping attack don't attack routing traffic, i.e., the routing information that the source node received is real. Can the source node depend on the routing information and the detected malicious insiders to choose a safe and short route? Here we propose a new routing protocol that is capable of mitigating the packet dropping attack and maintaining network performance, i.e., find a short-and-safe route for the source node to transmit packets.

We propose a short-and-safe routing (SSR) protocol to find a both safe and short route for transmitting data packets. Before running this protocol, the source node should be able to distinguish safe routes from unsafe ones. A route including suspicious nodes is considered to be an unsafe route. Actually, the safety information has been included in the route before sending back to the source node. The source node only needs to check the safety flag to see if it is a safe route or not. This flag is set by the intermediate normal node on the route.

As an intermediate normal node receives a Route Response (RREP) from a suspicious neighbor, it sets the unsafe route flag in RREP as true before forwarding it back to the source node. It is possible that there is more than one malicious node on a RREP route, and some of

them try to reset the flag to off if they find the flag has been set on, which is a kind of cooperation attack. However this kind of cooperation attack can't be successful, because the malicious node nearest to the source node can always be detected by its upstream node, which is an intermediate node or the source node itself. So the safety flag can always be correctly set, and help the source node to make a right decision.

In paper [26], the authors proposed an effective algorithm to identify multi-cooperative black holes. That algorithm assumes that a source node (SN) considers a node is believable if it has routed data through this node recently, otherwise the node is unbelievable. The process of finding cooperative black holes is as below: (1) the SN broadcasts a RREQ to find a rout to the destination node; (2) if an intermediate node has a route to the destination, it sends back a RREP to the SN, otherwise appends itself to the end of the route then forward this updated RREQ; (3) later, the SN receives the first RREP from a node. If this node is believable, the SN sends packets to it, otherwise ask for its next hop node (NHN); (4) if this NHN is still unbelievable, the SN asks for its NHN again, this process keeps going until a believable NHN is reached; (5) if the last unbelievable NHN provides different data routing information (DRI) from the one of the believable NHN, the SN can judge that all these queried unbelievable NHNs are cooperative black holes.

This algorithm can effectively disclose cooperative black hole attack. But in practical sensor network applications, malicious nodes are likely to adopt more flexible strategies to avoid being easily detected, for example, initiating gray hole attack by transmitting routing packets and dropping data packets. To solve this problem, we propose a short-and-safe routing protocol indicated in algorithm 2.

Figure 9 shows an example of a gray hole attack. There are ten nodes in this graph. Node S represents the source node, node D represents the destination node, node 3 represents a gray hole, which is a malicious node conducing a gray hole attack, and others are normal nodes. When the gray hole receives a route request (RREQ) broadcasted by the source node, it forwards the RREQ to its neighbor nodes instead of dropping it. When this request reaches to the intermediate nodes that have a fresh enough route to the destination node, in this case nodes 5 and 6, they initiate a route response (RREP) and unicast back to the origination of the RREQ. A fresh enough route is a valid route entry for the destination whose associated sequence number is at least as great as that contained in the RREQ. Because each node receiving the request caches a route back to the originator of the request, the RREP can be unicast from any intermediate node which is able to satisfy the request to the source node. These intermediate nodes also append the route from them to the destination in the RREPs. Later the source node receives some optional routes to the destination node. Some routes consisting of all normal nodes may be longer than those consisting of both normal nodes and gray holes. The source node should choose or recombine a route that is both short and safe based on received optional routes.

In Figure 9, there are two RREPs initialed by normal nodes 5 and 6 respectively. The two original routes and the merged route are:

(1) The first original route is $5 \rightarrow 3 \rightarrow 2 \rightarrow S$. The entire route is $D \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow S$. The number of hops of this route is five, and it is an unsafe route because a gray hole is in this route.

(2) The second original route is $6 \rightarrow 4 \rightarrow S$. The entire route is $D \rightarrow 9 \rightarrow 8 \rightarrow 5 \rightarrow 6 \rightarrow 4 \rightarrow S$. The number of hops of this route is six, and it is a safe route. Since the shorter route is

unsafe and the safe route is longer, we can merge these two routes and discover a new route that is safe and shorter than any received safe routes. The merged route is below.

(3) The merged route is $D \rightarrow 7 \rightarrow 5 \rightarrow 6 \rightarrow 4 \rightarrow S$. This route has five hops, the same as the first route. But it doesn't have any malicious nodes, it is a safe route. The source node will choose the merged route to send data packets.

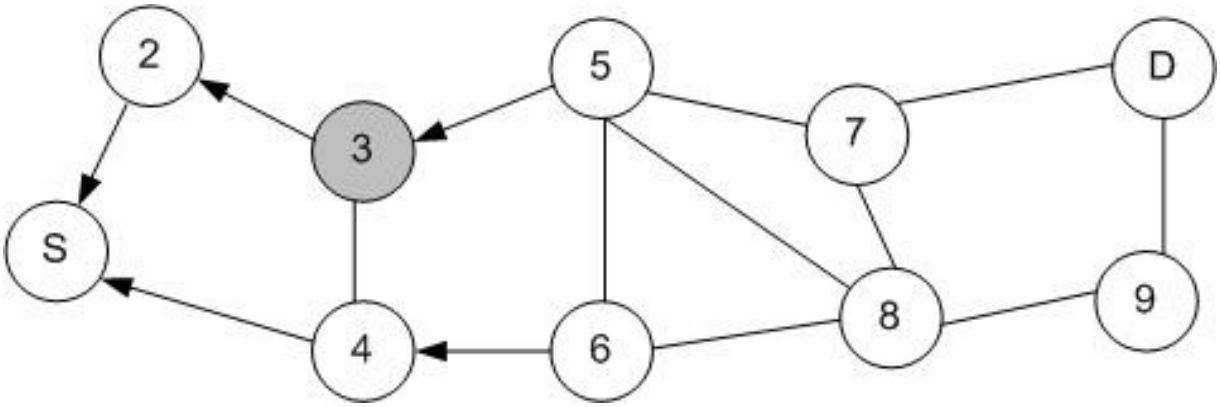


Figure 9. Propagation of RREP messages (node 3 is a gray hole attacker)

Algorithm 2 first sorts received safe routes. Then it saves the rout with shortest path length in a variable. Next, each intermediate node (IN) that initiates routing request response is checked to see if a shorter route can be found based on received safe routes. If a safe route includes this IN, a new fresh route is discovered. It consists of the first part of this safe route, this IN, and the second part of the route that the IN is in. Figure 10 shows an example of discovering a new route from an unsafe route and a safe route, in which (a) is an unsafe route with a suspect node X, (b) is a safe route including node N that initiates a RREP in the unsafe route, and (c) is the merged route that is safe and shorter than the route in (b). If this new safe route is shorter than the previously saved one, it is saved as the new shortest route. Finally, as all intermediate nodes are checked, the shortest route is found.

Algorithm 2: Short-and-safe routing (SSR) protocol

Input: the unsafe route and all optional safe routes

Output: *SS*

1. Sort all safe routes and save the shortest one to the variable *SS*
2. ***for*** (all intermediate nodes -*INs* that initiate RREP)
3. ***for*** (all safe routes)
4. ***if*** (the considering *IN* is in the considering safe route)
5. Merge these two routes into a new safe route that consists of the first part of the safe route, the *IN*, and the second part of the unsafe route
6. ***if*** (this new route is shorter than *SS*)
7. Save this route to *SS*
8. ***end if***
9. ***end if***
10. ***end for***
11. ***end for***

Since a normal node maintains a reputation table of its neighbors, it is easy to decide if a neighbor is safe or not for transmitting packets. When each intermediate node transmits a RREP route back to its upstream node, it sets the flag indicating the safety of a node "on" if it thinks this upstream node is unsafe based on its reputation table. Otherwise, it sets the flag "off". At last, the source node can check this flag to decide if it is a safe RREP

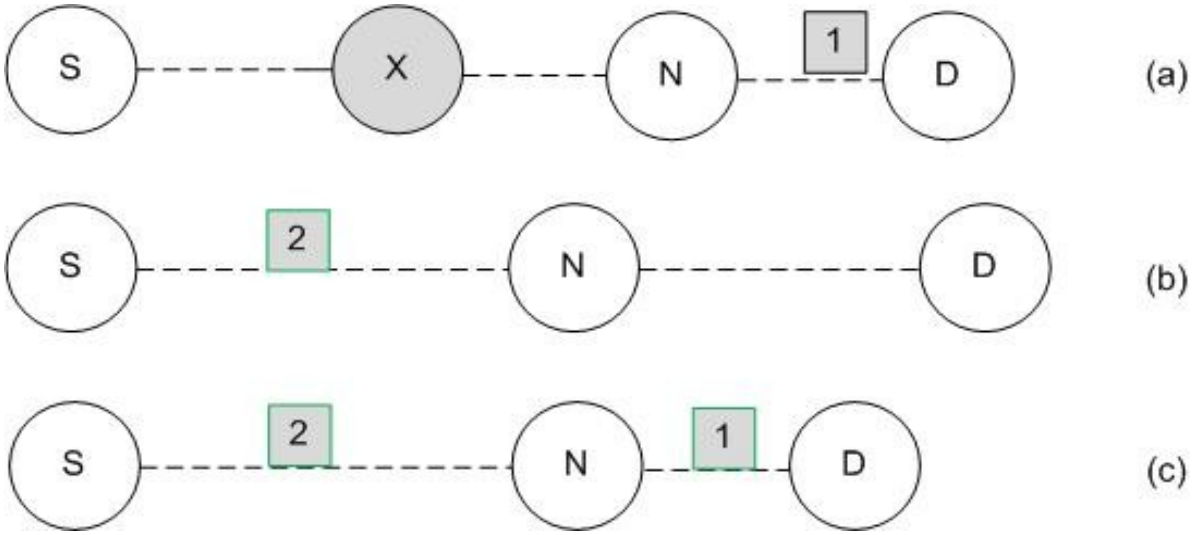


Figure 10. An example of merging two routes

CHAPTER 5. MULTI-TARGET TRACKING

The proposed techniques for detecting malicious insiders were introduced in the previous chapter. Now we focus on the designed techniques on detecting malicious outsiders (malicious intruders). Malicious intruders may attack a sensor network through their agents (malicious insiders) or themselves. No matter what methods malicious intruders adopt to attack a sensor network, the most effective way to prevent their attack is to track them down. In this chapter, we fist introduce some theoretical knowledge on mobile target tracking, and then present our proposed multi-target tracking techniques.

5.1. Bayesian Theory

5.1.1. The Bayes Formula

We will here review the terminology used for the Bayesian analysis of uncertainties in modeling. To simplify the notations in this section, let θ stand for all the unknowns in our model. The inference concerning θ is performed using the conditional distribution of the parameter, given the model and the observations $p(\theta|y)$. This probability distribution is called the posterior distribution. By the rules of conditional probabilities, we can invert the posterior and arrive at the Bayes formula:

$$p(\theta|y) = \frac{p\ (y|\theta)p\ (\theta)}{p\ (y)}.$$

The right hand side of the Bayes formula has the following ingredients. The likelihood function, $p(y|\theta)$, is the density function of the observations y y given θ. When the observed data y are held fixed, $p(y|\theta)$ is considered as a function of the parameters θ. If for two possible parameter values θ and θ' we have $p(y|\theta') > p\ (y|\theta)$, then the observation y is more likely to happen under θ' than under θ, hence the term "likelihood."

The unconditional distribution of the unknowns, $p(\theta)$, is called the prior distribution. This describes the knowledge about the unknowns that existed before, or exists independent of, the current observations. Again, when the data y are fixed, the term in the denominator, $p(y)$, is a single number, which can be seen as a constant that normalizes the likelihood-times-prior part into a proper probability distribution. When comparing different models for the same data, this unconditional probability of observations can also be seen as a (prior) predictive probability of the observations given the model. For this reason it is sometimes called the evidence. By the rule of total probability, we can calculate its value by integrating the product $p(y|\theta)p\ (\theta)$ over all values of θ, i.e.,

$$p(y) = \int p(y|\theta)p(\theta)d\theta.$$

This integral makes the actual computation of the posterior distribution a challenging task in all but some special cases. For most cases we need special numerical algorithms, e.g., the MCMC algorithm described below.

5.1.2. Predictive Distribution

An important element in Bayesian inference is the prediction of future observations y^* given the current ones y. Given the observations y and the model $p(y|\theta)$, the predictions are naturally based on the posterior distributions of θ. If we assume that y and y^* are conditionally independent given the value of θ, we can write

$$p(y^*|y) = \int p(y^*, \theta|y)d\theta = \int p(\ y^*|\theta)p(\theta|y)d\theta.$$

Here the predictive distribution of y^* is given as the expectations of $p(y^*|\theta)$ with respect to the posterior distribution $p(\theta|y)$. In this work we use the simulated MCMC chain as a sample from $p(\theta|y)$. If we sample θ from the chain and y^* from $p(y^*|\theta)$, we obtain samples of the predictive distribution $p(y^*|y)$. The predictive distribution is the basis for the validation of the

model against reality. This is not possible using only the posterior distribution of the model parameters θ, as the parameters are not directly observable.

5.1.3. Prior Information

Next, we consider the role of prior information and the problem of assigning the prior. In a straightforward classical interpretation of Bayesian inference the prior signifies the modeler's honest opinion about the unknown. In most modeling activities dealing with scientific inference, however, the idea of subjective probability seems restrictive. A common concern has to do with the problem of formulating the prior as "objectively" as possible. The solutions proposed include such concepts as non informative [51], hierarchical [52], reference [53], vague, or diffuse [54] priors. We will encounter three uses of the prior distribution in the course of this work: uninformative priors, describing a lack of, or unwillingness to provide prior information on the unknowns, priors that provide structural information about the solution, and finally priors that are based on empirical evidence from previous experiments. A modeler would usually be happy if the data were to produce informative posteriors "without a prior". Even if we had information about the unknowns from previous similar experiments, we would like the data to verify our conclusions. This usually means using very wide priors in the Bayes formula for some or all of the components of the unknown. Care must be taken that the posterior becomes a proper distribution.

Sometimes the prior gives the structure of the solution. In inverse problems where the state of the system is estimated, for example, we can have information on some of the properties of the system. In these cases the solution might not exist, or it could very unstable, unless some prior information is used [55]. If this regularization of the problem is done via a prior

distribution, we can have an intuitive interpretation for the restrictions and a better view of the effects of the premises on the conclusions.

When the data are sparse a great deal of effort should be put into selecting the prior distribution. When possible, the prior should be based on real information. In large-scale problems with a large number of unknowns it is usually not possible to formulate detailed priors. If we assume that the unknowns are a priori independent, we can independently assign one-dimensional densities to each of them.

If model parameters can be given physical interpretations, it is a lot easier to assess the prior. A natural prior restriction in many models is a requirement of positivity. The use of priors makes it possible (for good and evil) to consider models that are more complicated than would be necessary for predictive purposes. Assessing priors can in practice be seen as an iterative process of the same kind as the building of a model. For example, if a model is described by differential equations, the initial values of the state variables might be unknown. If we have observations on the state variables at the beginning of the period, then the prior for the initial values should contain information from these observations, and also information about the accuracy of the observations. But the latter is probably available only after a preliminary fit, from the residual variance.

There exists a longstanding philosophical controversy over the nature of statistical inference and its relation to the "real world", as in the question of whether it is appropriate, or even possible, to assign a statistical prior distribution to the value of an unknown parameter in a model, or for an unknown state in nature, if these are not random variables. In science we must be able to criticize all aspects of our methods and results. This includes the statistical paradigm used. Applied Bayesian statistics works very well in practice. Thanks to the computational

Markov Chain Monte Carlo methods, we are able to work with more complicated and realistic model than before, pool prior information from various sources, and easily produce probability statements about the predictions of the model that can be verified by future observations.

5.2. Markov Chain Monte Carlo (MCMC)

High-dimensional estimation problems pose a computational challenge that can be solved only by simulation methods. The normalizing constant makes computation of the posterior distribution in difficult problem, especially in multidimensional cases. A clever algorithm, Metropolis-Hastings Algorithm, provides a simple method for simulating values from a distribution that can be calculated only up to a normalizing constant.

We will use the notation $\pi(\theta)$ for the target distribution of interest. This is common in the MCMC literature. In most cases, the target will be the posterior distribution for the model unknown, $\pi(\theta) = p(\theta|y)$.

In MCMC simulation we produce a sequence of values which are not independent but instead follow a stochastic process called a Markov chain. The algorithm used in the simulation ensures that the chain will take values in the domain of the unknown θ and that its limiting distribution will be the target distribution $\pi(\theta)$. This means that we have a method of sampling values from the posterior distribution and therefore of making Monte Carlo inferences about θ in the form of sample averages and by means of histograms and kernel density estimates.

The MCMC algorithm produces a chain of values in which each value can depend on the previous value in the sequence. For example, a random walk MCMC algorithm advances the sequence by proposing and conditionally accepting or rejecting new values by means of a proposal distribution centered at the current position of the MCMC chain. The values are not independent but instead have some positive autocorrelation. Because of this, the sample averages

used as estimates for the corresponding posterior values have error due to the sampling procedure, so called Monte Carlo error, that is larger than in the i.i.d. (independent identically distributed) case. There is no general way to get rid of this problem, however. On the other hand, random walk sampling in a high-dimensional space has advantages over i.i.d. sampling. As paper [54] points out, high-dimensional spaces are very sparse. If we consider the unit hyper sphere that is located inside a unit hyper cube, we see that the total space consists almost entirely of corners and a very small part of the volume of the space is contained inside the sphere. We are confronted with problems of finding the regions of statistical significant probability and of exploring those regions. Random walk-type methods try to offer solutions to the problem of getting lost in the space. A remedy for larger Monte Carlo errors is to perform longer simulations than would be needed in the i.i.d. sampler case, and also to try to make the MCMC methods as efficient as possible. The latter is one of the main motivations behind the present work.

5.3. The Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm is currently the most general algorithm for MCMC simulation. Its basic form is easy to explain and implement and it has several useful generalizations and special cases for different purposes. The basic idea depends on the fact that, if instead of computing the values $\pi(\theta)$ we need only compute the ratio of the target at two distinct parameter values $\pi(\theta)/\pi(\theta^*)$, the integral in the Bayes formula cancels out.

With an MCMC algorithm we are generating a chain of values $\theta^0, \theta^1, \ldots, \theta^N$ in such a way that it can be used as a sample of the target density $\pi(\theta)$. In terms of the Markov chain theory [59], when using the Metropolis-Hastings algorithm we generate a Markov chain that has a transition kernel according to

$$p(\theta, \theta^*) = q(\theta, \theta^*)\alpha(\theta, \theta^*), \ \theta \neq \theta^*,$$

$$p(\theta,\theta) = 1 - \int q(\theta,\theta^*)\alpha(\theta,\theta^*)\,d\theta$$

for some transition density q , and for an acceptance probability α. The density $q(\theta, \cdot)$, with θ being the current location of the chain, is called the proposal density. The chain is said to be reversible if we have

$$\pi(\theta)q(\theta,\theta^*)\alpha(\theta,\theta^*) = \pi(\theta^*)q(\theta^*,\theta)\alpha(\theta^*,\theta).$$

Reversibility is a sufficient condition for the density π to be the stationary distribution of the chain,

$$\int \pi(\theta)p(\theta,\theta^*)d\theta = \pi(\theta^*)$$

meaning that if the chain were to reach π, it would also follow this distribution for the rest of the simulation. This leads to the choice of the Metropolis-Hastings acceptance probability α as

$$\alpha(\theta,\theta^*) = min(1, \frac{\pi(\theta^*)q(\theta^*,\theta)}{\pi(\theta)q(\theta,\theta^*)}).$$

The parameter space Θ is usually a subset of $\mathbb{R}^d$, but the reversibility condition can be formulated for more general state spaces.

The proposal distribution from which we choose new values for the chain can be quite arbitrary, but choosing a distribution that most closely resembles the true target distribution can dramatically speed up the convergence of the values generated to the right distribution. The closer the proposal distribution q is to the actual target $\pi(\theta)$, the better the chain mixes and the better a short sequence represents a random draw from the posterior. This is especially true in multidimensional cases and when there is correlation between the components of the parameter vector. In the applications described in this work the proposal density is taken to be the multidimensional Gaussian density.

The algorithm is constructed in such a way that the target distribution π is the stationary distribution of the Markov chain. This means that the generated values will eventually follow the posterior distribution π. In practise, we must allow some burn-in time to let the chain become close enough to the limiting distribution. The MH algorithm can be thought of as travelling uphill towards the peak of the posterior distribution, but occasionally taking steps downhill. The percentage of time spent in each region of the hill corresponding to the probabilities of the target distribution.

For the convergence results we need some theory of Markov chains, although with one important simplification: it is known by construction that the stationary distribution π exists. Also, we are able to choose the initial distribution arbitrarily. This provides simple ways of proving important ergodic properties of the MH chain: The Law of Large Numbers type of theorem that says that we can use sample averages as estimates and apply the Central Limit Theorem, which gives us the convergence rate for the algorithms.

5.4. Multi-target Tacking Problem

The tracking problem can be formulated as a stochastic estimation problem, where the goal is to estimate the mostly likely state S_k at time $t = k$, given a history of sensor measurements $\{S_1, \dots, S_k\}$. This formulation requires two models - the dynamic model $S_k = f(S_{k-1}, n_{k-1})$ and the measurement model $Z_k = g(S_k, w_k)$, where n_k and w_k are the stochastic transition noise and the measurement noise, respectively. The seminal work by Kalman [60] provides an optimal, iterative solution of this formulation under rather strict assumptions of linear, Dynamic and Gaussian measurement models given as follows:

$$S_k = A_k S_{k-1} + n_{k-1},$$

$$Z_k = H_k S_k + w_k.$$

For sensing modalities like acoustic sensors or laser range sensors, however, the Gaussian assumption is not satisfied and a more general mathematical framework is required to deal with the general noise characteristics. Bayesian filtering [61] is a mathematical framework, where one can iteratively compute a posterior distribution given a history of measurements. Bayesian filtering is named for English mathematician Thomas Bayes, who developed a theory of probability inference. The main theorem of Bayesian Filtering says that the posterior density $p\ (S_k\ |Z_k\ )$ can be computed recursively as follows:

$$p\ (S_k\ |Z_k\ ) \propto p\ (Z_k|\ S_k\ ) \int p(S_{K-1}|Z_{k-1})p(\ S_K|S_{K-1})dS_{K-1},$$

where $p\ (Z_k|\ S_k\ )$ is a measurement model given as a likelihood function and $p(S_K|S_{K-1})$ is a dynamic model. Figure 11 illustrates the finding of posterior distribution $p\ (S_{0:n}\ |Z_{0:n}\ )$, where $S_{0:n}$ are the states that can't be obersed and have to be estimated, and $Z_{0:n}$ are the nosiy obervations. Nonparametric methods based on the Monte-Carlo sampling have been successfully used to solve the above Bayesian filtering equation in many different applications [62, 63]. It is relatively straightforward to apply the aforementioned Bayesian filtering framework to the single-object tracking problem and, as a result, it is considered more or less a solved problem in the tracking community. The multi-object tracking problem, however, is not a simple extension of the single-object tracking problem due to the data association problem - which measurements are associated with which states in the multi-object setting? Figure 12 illustrates the main challenge of the data association problem for a simple case of two objects. Given N states S_k and N measurements Z_k, there are $N!$ ways to associate the measurements with the states. One can see that, in the worst case, there are exponentially many ways to associating measurements to states.

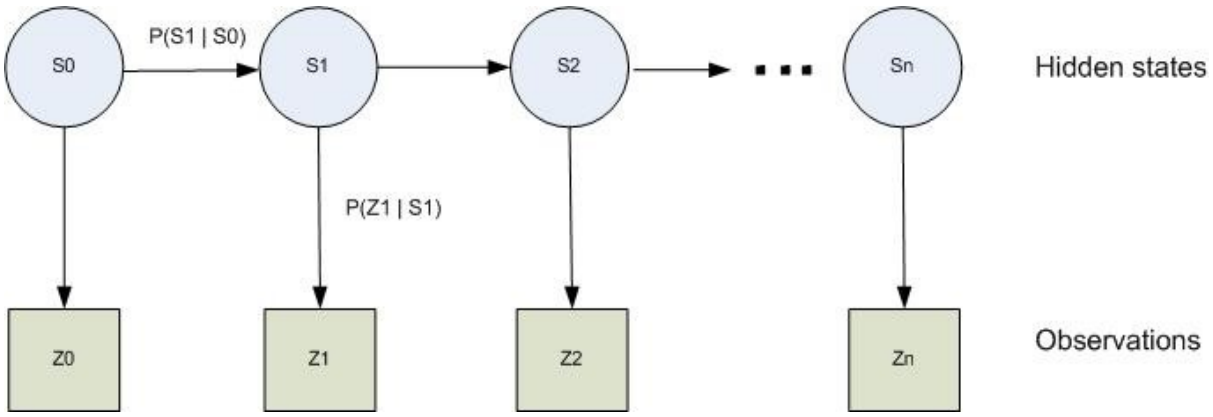


Figure 11. Posterior distribution calculation

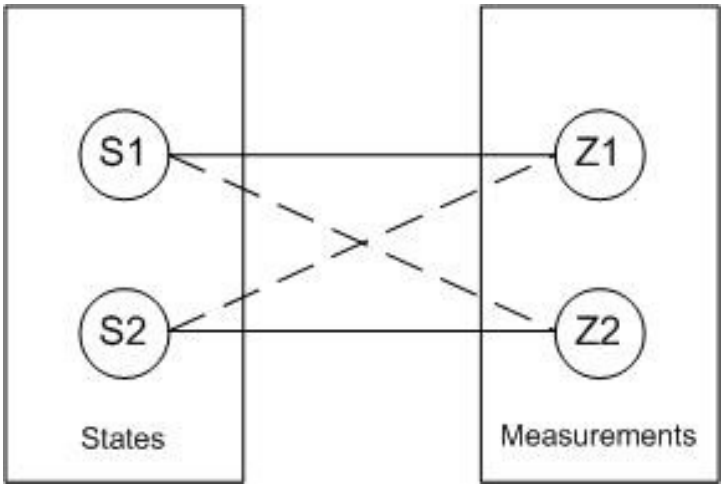


Figure 12. A problem of states and measurements association

There is no any computationally efficient algorithm for this problem. This problem is known to be NP-hard, since one can formulate the data association problem as the multi-dimensional assignment (MDA) problem [64, 65].

As a result of the NP-hardness, all the existing data association algorithms are heuristics. In other words, there always exists a finite probability of the tracking system being confused

about the identities of objects after a data association algorithm is applied, which we call identity swapping. Figure 13 illustrates the identity swapping. In the figure, two objects moving along curly lines. A sub-optimal data association algorithm, however, could make a wrong association and concludes that the lower object is the object 1 and the upper object is the object 2. When there are many objects, these could lead to many instances of identity swapping and it is not possible to design reliable high-level applications based on the outputs of the tracking systems.

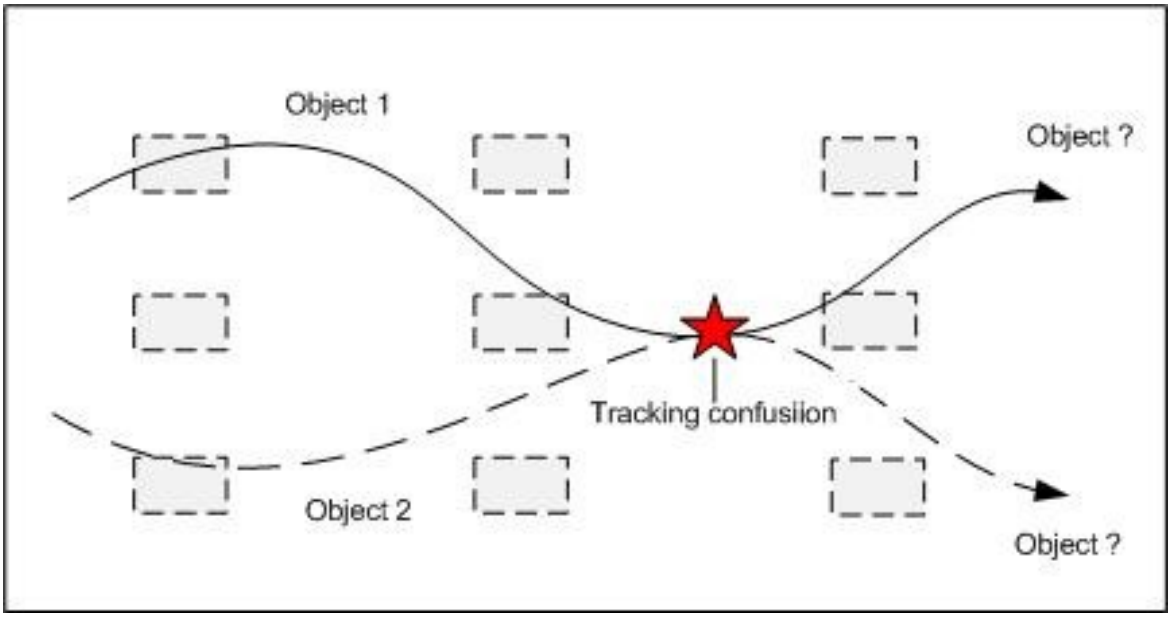


Figure 13. Entity swapping example

Given the nature of the data association problem, it seems that we cannot avoid the identity swapping problem. In a WSN, however, it is likely that some sensors are very close to moving objects and thus able to sense attributes of objects that can be used to fix the identity swapping problem. For example, suppose a sensor near the bottom object in Figure 13 reports the nearby object is actually the object 2. In this simple case, this local evidence implies that the upper object is the object 1and we can effectively fix the identity swapping problem using only

local evidence on object identity. We use the exclusion among the two identities to update both of the identities - global information - using only local evidence.

5.5. Multi-target Tracking in WSNs

Multiple-target tracking is a representative real-time application of sensor networks as it exhibits different aspects of sensor networks such as event detection, sensor information fusion, multi-hop communication, sensor management and real-time decision making. The task of tracking multiple objects in a sensor network is challenging due to constraints on a sensor node such as short communication and sensing ranges, a limited amount of memory and limited computational power. In addition, since a sensor network surveillance system needs to operate autonomously without human operators, it requires an autonomous real-time tracking algorithm which can track an unknown number of targets. In this work, we develop a scalable real-time multiple-target tracking algorithm that is autonomous and robust against transmission failures, communication delays and sensor localization error. In particular, there is no performance loss up to the average localization error of .7 times the separation between sensors and the algorithm tolerates up to 50% lost-to-total packet ratio and 90% delayed-to-total packet ratio.

In wireless ad-hoc sensor networks, many inexpensive and small sensor-rich devices are deployed to monitor and control our environment [81]. Each device, called a sensor node, is capable of sensing, computation and communication. Sensor nodes form a wireless ad hoc network for communication. The limited supply of power and other constraints, such as manufacturing costs and limited package sizes, limit the capabilities of each sensor node. For example, a typical sensor node has short communication and sensing ranges, a limited amount of memory and limited computational power. However, the abundant number of spatially spread sensors will enable us to monitor changes in our environment accurately despite the inaccuracy

of each sensor node. Multiple-target tracking is a representative real-time application of sensor networks as it exhibits different aspects of sensor networks such as event detection, sensor information fusion, communication, sensor management, and real-time decision making. The applications of tracking using sensor networks include surveillance, search and rescue, disaster response system, pursuit evasion games [83], distributed control [84], spatial-temporal data collection, and other location based services [82]. However, the task of tracking multiple objects in a sensor network is challenging due to the following issues. Each sensor node has a limited supply of power, leading to low detection probability and high false alarm rate. The presence of false alarms and missing observations complicate the problems of track initiation and termination. These important issues are ignored by many tracking algorithm designed for sensor networks. For example, when the false alarm rate is high, a naive track initiation algorithm will overflow the network with spurious tracks. Hence, an algorithm for sensor networks must be robust against the low detection probability and high false alarm rate. To conserve power and reduce interference, multi-hop routing is used in sensor networks. In many cases, the communication links are not reliable, causing transmission failures. In addition, due to the low communication bandwidth and a limited amount of memory, communication delays can occur frequently. Moreover, the localization of sensor nodes in an ad-hoc wireless sensor network without expensive hardware such as the global positioning system (GPS) is a challenging problem [85]. Since the position of a target is reported with respect to the location of the reporting sensor, the algorithm must be robust against the sensor localization error. It is well known that communication is costlier than computation in sensor networks in terms of power usage [86]. Hence, it is essential to fuse local observations before the transmission. However, since the data association problem of multiple-target tracking is NP-hard [87], we cannot expect

to solve it with only local information. But at the same time we cannot afford to have a centralized algorithm since such a solution is not scalable. Lastly, in sensor networks, we seek for an autonomous tracking algorithm which does not require a continuous monitoring by a human operator. In summary, we need a real-time tracking algorithm that is robust against the low detection probability and high false alarm rates; capable of initiating and terminating tracks; uses less memory; combines local information to reduce the communication load; and is scalable. Also it must be robust against transmission failures, communication delays and sensor localization error. But at the same time we want an algorithm that provides a good solution which approaches the optimum given enough computation time.

In [8], Markov chain Monte Carlo data association (MCMCDA) is presented. MCMCDA can track an unknown number of targets in real-time and is an approximation to the optimal Bayesian filter. It has been shown that MCMCDA is computationally efficient compared to the multiple hypothesis tracker (MHT) [75] and outperforms MHT under extreme conditions, such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates [8]. MCMCDA is suitable for sensor networks since it can autonomously initiate and terminate tracks. Since transmission failure is another form of a missing observation, MCMCDA is robust against transmission failures. MCMCDA performs data association based on both current and past observations, so delayed observations, i.e., out-of-sequence measurements, can be easily combined with previously received observations to improve the accuracy of estimates. Furthermore, MCMCDA requires less memory as it maintains only the current hypothesis and the hypothesis with the highest posterior. It does not require the enumeration of all or some of hypotheses as in [75].

MCMCDA uses MCMC sampling instead of summary over all possible associations like in MHT. This sampling method outperforms MHT on running time and memory requirement, especially under dense environment. However, MCMCDA still has a high rejection rate during sampling process. How to decrease the rejection rate, thereby further accelerate the tracing speed, is the main solving problem of our tracking algorithm.

In this work, we extend the MCMCDA algorithm to sensor networks in a hierarchical manner so that the algorithm becomes scalable, and is able to systematically track an unknown number of targets in the presence of false alarms and missing observations and is robust against transmission failures, communication delays and sensor localization error. We consider a simple shortest-path routing scheme on a sensor network. The transmission failures and communication delays of the network are characterized probabilistically. We assume the availability of a small number of special nodes, super-nodes that are more capable than regular nodes in terms of computational power and communication range. Each node is assigned to its nearest super-node and nodes are grouped by super-nodes. We call the group of sensor nodes formed around a super-node as a "tracking group". When a node detects a possible target, it communicates with its neighbors and observations from the neighboring sensors are fused and sent to its super-node. Each super-node receives the fused observations from its tracking group and executes the tracking algorithm. Each super-node communicates with neighboring super-nodes when a target moves away from its range. Lastly, the tracks estimated by super-nodes are combined hierarchically. Although a specific sensor network model is used for the performance evaluation, the algorithm is applicable for different routing algorithms and sensor models, e.g., distributed air traffic control [88].

5.6. Multi-Target Tracking Algorithm

5.6.1. Problem Formulation

In [8], 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 [89]. 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 of 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 of V. 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 the values K and $[t_a^k, t_b^k]$ (k = 1, 2, …, K).

5.6.2. MCMC Based 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:

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

where parameter 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:

$$P(\omega | Y) \propto P(Y|\omega) * P(\omega),$$

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 to find a state ω^* with the maximum posterior among all of the checked states, i.e.,

$$\omega^* = arg\, max(P(\omega|Y).$$

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 have not been 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.

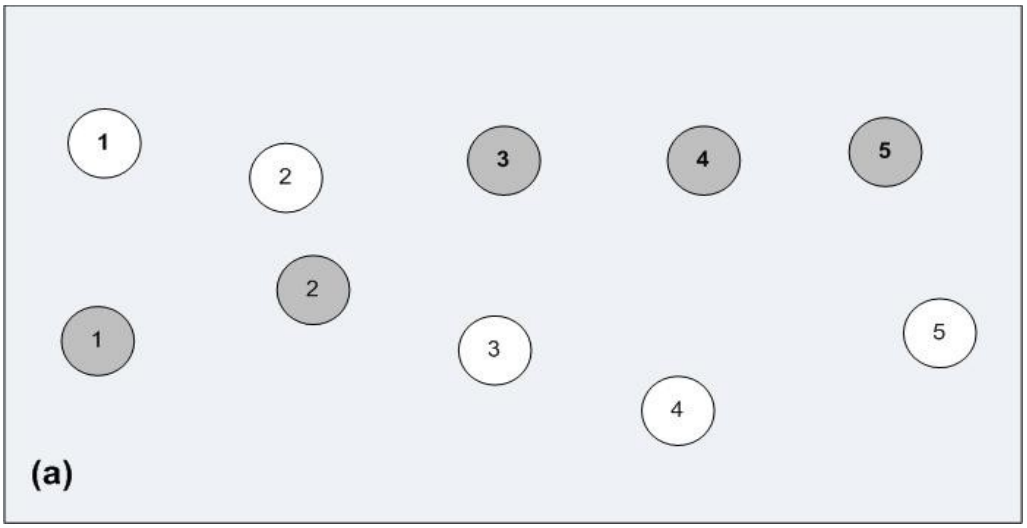
5.6.3. State Space Size Reduction

In [3] the authors make assumptions that any target has a maximal directional speed $\overline{v}$, and that the number of consecutive missing observations of any track is less than $\bar{d}$. In a sensor network, more powerful nodes (e.g., cluster heads) or the control center is the place to implement the tracking algorithm based on received observations from normal sensor nodes. Instead of only reporting a moving target's state, a sensor node could attach a little more information like the target's danger level in the reporting data. This attachment won't overload the network traffic but provide valuable clues on getting correct data associations. 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.

5.6.3.1. Distinguishing Normal and Abnormal Observations

When a sensor node receives a modified or fake message from an intruder, it reports an abnormal observation to the cluster head. Otherwise it 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 14 shows an example of partitioning tracks with classified observations. Figure 14 (a) is an example of observation *Y*, in which dark circles represent abnormal observations, hollow circles represent normal observations, and the numbers represent observation times. Figure 14 (b) represents an example of a partition ω of *Y*.

In this example, there are two observations in each of the five time steps. Without distinguishing the observations, there are thirty-two optional associations or tracks, assuming each track with five observations. However, there are only two associations with distinguished observation. It's obvious that distinguishing observations helps decrease searching correct associations scope.



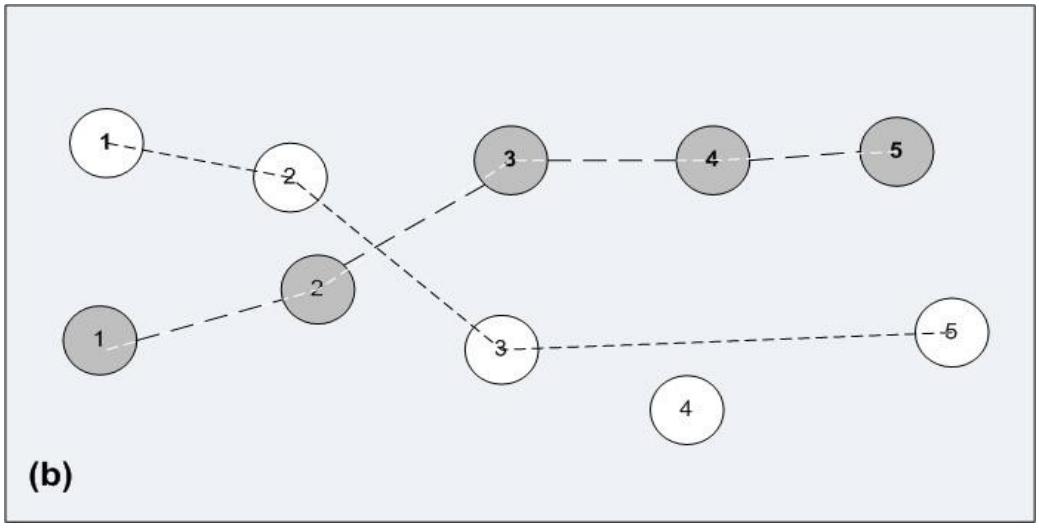


Figure 14. An example of track partition

5.6.3.2. Forecasting Intruder Moving Direction

We follow the method of [90], 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 [91].

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 15 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.

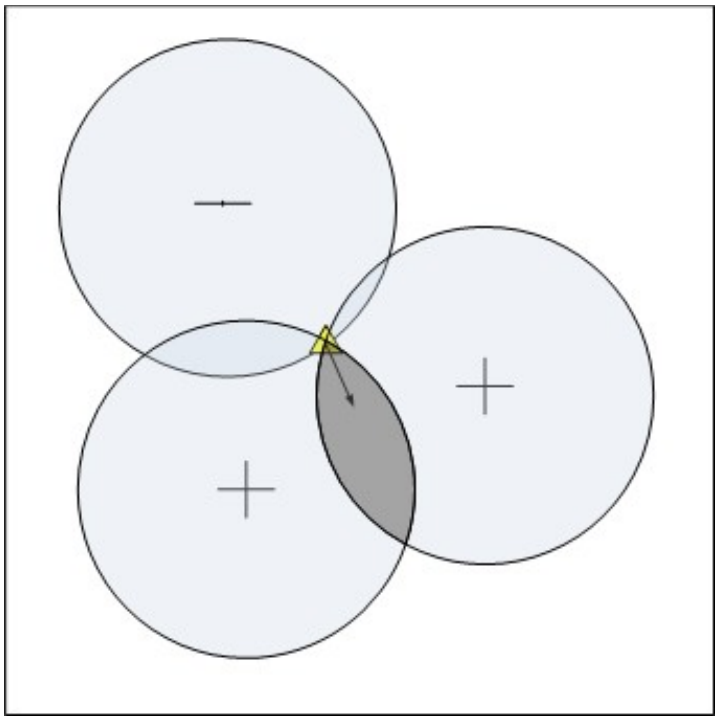


Figure 15. The future moving direction of an intruder within the shaded area

Traditionally, the joint particle filter is used for sampling on target tracking. However, this filter method suffers from exponential complexity in the number of tracked targets. As a result, we replace this traditional importance sampling step in the particle filter with an MCMC sampling step. This approach has the appealing property that the filter behaves as a set of individual particle filters when the targets are not interacting, but efficiently deals with complicated interactions when targets approach each other. The simulation results will show that the MCMC based tracking method is efficient on tracking multi-targets.

Detailing observations like distinguishing normal observations from abnormal ones and sensing targets' moving direction is one approach we adopted to accelerate tracking speed. Another approach is to use Tabu Search technique to find an optimal data association among observations. Since the proposed Tabu Search method has a general function on searching for an optimal solution based on received data, we devote a separate chapter to introduce it.

CHAPTER 6. TABU SEARCH

Although the MCMCDA give good results, it has one major disadvantage in that it has a high rejection rate during the sampling process. Motivated by this, we decide to adopt Tabu search (TS) technique to mitigate the high rejection rate. First, some basic concepts of Tabu search are introduced, and then is our designed Tabu search algorithm.

6.1. Basic Concepts

6.1.1. Historical Background

Before introducing the basic concepts of Tabu search, we believe it is useful to go back in time to try to better understand the genesis of the method and how it relates to previous work. Heuristics, i.e., approximate solution techniques, have been used since the beginnings of operations research to tackle difficult combinatorial problems. With the development of complexity theory in the early 1970s, it became clear that, since most of these problems were indeed NP-hard, there was hope of ever finding efficient exact solution procedures for them. This realization emphasized the role of heuristics for solving the combinatorial problems that were encountered in real-life applications and that needed to be tackled, whether or not they were NP-hard. While many different approaches were proposed and experimented with, the most popular ones were based on hill climbing. The latter can roughly be summarized as an iterative search procedure that, starting from an initial feasible solution, progressively improves it by applying a series of local modifications or moves (for this reason, hill climbing is in the family of local search methods). At each iteration, the search moves to an improving feasible solution that differs only slightly from the current one. In fact, the difference between the previous and the new solution amounts to one of the local modifications mentioned above. The search terminates when no more improvement is possible. At this point, we have a local optimum with regard to

the local modifications considered by the hill climbing method. Clearly, this is an important limitation of the method: unless one is extremely lucky, this local optimum will often be a fairly mediocre solution. The quality of the solution obtained and computing times are usually highly dependent upon the "richness" of the set of transformations (moves) at each iteration. In 1983, a new heuristic approach called simulated annealing [95] was shown to converge to an optimal solution of a combinatorial problem, albeit in infinite computing time. Based on analogy with statistical mechanics, simulated annealing could be interpreted as a form of controlled random walk in the space of feasible solutions. The emergence of simulated annealing indicated that one could look for other ways to tackle combinatorial optimization problems and spurred the interest of the research community. In the following years, many other new approaches, mostly based on analogies with natural phenomena, were proposed such as Tabu search, ant systems and threshold methods. Together with some older ones, in particular genetic algorithms, they gained an increasing popularity. Now collectively known under the name of meta-heuristics, a term originally coined by [93], these methods have become, over the last 15 years, the leading edge of heuristic approaches for solving combinatorial optimization problems.

6.1.2. Tabu Search

Fred Glover proposed in 1986 a new approach, which he called Tabu search, to allow hill climbing to overcome local optima. In fact, many elements of this first Tabu search proposal, and some elements of later elaborations, had already been introduced in [96] including short-term memory to prevent the reversal of recent moves, and longer-term frequency memory to reinforce attractive components. The basic principle of Tabu search is to pursue the search whenever a local optimum is encountered by allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories (called tabu lists) that record the recent

history of the search. The key idea to exploit information to guide the search can be linked to the informed search methods proposed in the late 1970s in the field of artificial intelligence. It is also important to remark that Glover did not see Tabu search as a proper heuristic, but rather as a meta-heuristic, i.e., a general strategy for guiding and controlling "inner" heuristics specifically tailored to the problems at hand.

6.1.3. Search Space and Neighborhood Structure

As just mentioned, Tabu search extends hill climbing methods. In fact, the basic Tabu search can be seen as simply the combination of hill climbing with short-term memories. It follows that the two first basic elements of any Tabu search heuristic are the definition of its search space and its neighborhood structure. The search space is simply the space of all possible solutions that can be considered (visited) during the search. An attractive search space is the set of feasible vectors of location variables, i.e., feasible vectors in $\{0,1\}^J$ (where $|J|$ is the cardinality of set J), any solution in that space being "completed" to yield a feasible solution to the original problem by computing the associated optimal flow variables. It is important to note that it is not always a good idea to restrict the search space to feasible solutions. In many cases, allowing the search to move to infeasible solutions is desirable and sometimes necessary closely linked to the definition of the search space is that of the neighborhood structure. At each iteration of Tabu search, the local transformations that can be applied to the current solution, denoted SS, define a set of neighboring solutions in the search space, denoted $N(S)$ (the neighborhood of S). Formally, $N(S)$ is a subset of the search space defined by:

$$N(S) = \textit{solutions obtained by applying a single local transformation to } S.$$

In general, for any specific problem at hand, there are many more possible (and even, attractive) neighborhood structures than search space definitions. This follows from the fact that

there may be several plausible neighborhood structures for a given definition of the search space. This is easily illustrated on our job shop scheduling problem. In order to simplify the discussion, we assume in the following that the search space is the feasible space. Simple neighborhood structures for the job shop scheduling problem are obtained by considering the sequence of jobs associated with a machine schedule, where the position of a job in the sequence corresponds to its processing order on the machine. For example, one can move a job at another position in the sequence or interchange the position of two jobs. While these neighborhood structures involve only one or two jobs, the neighborhoods they define contain all the feasible schedules that can be obtained from the current one either by moving any single job at any other position or by interchanging any two jobs. Examining these neighborhoods can thus be fairly demanding. In practice, it is often possible to reduce the computational burden, by identifying a restricted subset of moves that are feasible and can lead to improvements. When different definitions of the search space are considered for a given problem, neighborhood structures will inevitably differ to a considerable degree. This can be illustrated on our capacitated plant location problem. If the search space is defined with respect to the location variables, neighborhood structures will usually involve the so-called "Add/Drop" and "Swap" moves that respectively change the status of one site (i.e., either opening a closed facility or closing an open one) and move an open facility from one site to another (this move amounts to performing simultaneously an Add move and a Drop move). If, however, the search space is the set of extreme points associated with feasible flow vectors, these moves become meaningless. One should instead consider moves defined by the application of pivots to the linear programming formulation of the transportation problem, where each pivot operation modifies the flow structure to move the current solution to an adjacent extreme point. The preceding discussion should have clarified a major point:

choosing a search space and a neighborhood structure is by far the most critical step in the design of any Tabu search heuristic. It is at this step that one must make the best use of the understanding and knowledge one has of the problem at hand.

6.1.4. Tabus

Tabus are one of the distinctive elements of Tabu search when compared to hill climbing. As we already mentioned, tabus are used to prevent cycling when moving away from local optima through non-improving moves. The key realization here is that when this situation occurs, something needs to be done to prevent the search from tracing back its steps to where it came from. This is achieved by making certain actions Tabu. This might mean not allowing the search to return to a recently visited point in the search space or not allowing a recent move to be reversed. For example, in the job shop scheduling problem, if a job j has been moved to a new position in a machine schedule, one could declare tabu moving that job back to its previous position for some number of iterations (this number is called the tabu tenure of the move). Tabus are stored in a short-term memory of the search (the tabu list) and usually only a fixed and fairly limited quantity of information is recorded. In any given context, there are several possibilities regarding the recorded information. One could record complete solutions, but this requires a lot of storage and makes it expensive to check whether a potential move is tabu or not; it is therefore seldom used. The most commonly used tabus involve recording the last few transformations performed on the current solution and prohibiting reverse transformations (as in the example above); others are based on key characteristics of the solutions themselves or of the moves. To better understand how tabus work, let us go back to our reference problems. In the job shop scheduling problem, one could define tabus in several ways. To continue our example where a job j has just been moved from position p_1 to position p_2, one could declare tabu specifically

moving back j to position p_1 from position p_2, and record this in the short-term memory as the triplet (j, p_2, p_1). Note that this type of tabu will not constrain the search much, but that cycling may occur if j is then moved to another position p_3, and then from p_3, to p_1. A stronger tabu would involve prohibiting moving back j to p_1 (without consideration for its current position) and be recorded as (j, p_1). An even stronger tabu would be to disallow moving j at all, and would simply be noted as j.

In the capacitated plant location problem, tabus on Add/Drop moves should prohibit changing the status of the affected location variable and can be recorded by noting its index. Tabus for Swap moves are more complex. They could be declared with respect to the site where the facility was closed, to the site where the facility was opened, to both locations (i.e., changing the status of both location variables is tabu), or to the specific swapping operation. Multiple tabu lists can be used simultaneously and are sometimes advisable. For example, in the capacitated plant location problem, if one uses a neighborhood structure that contains both Add/Drop and Swap moves, it might be a good idea to keep a separate tabu list for each type of move. Standard tabu lists are usually implemented as circular lists of fixed length. It has been shown, however, that fixed-length tabus cannot always prevent cycling, and some authors have proposed varying the tabu list length during the search [97]. Another solution is to randomly generate the tabu tenure of each move within some specified interval. Using this approach requires a somewhat different scheme for recording tabus, which are usually stored as tags in an array. The entries in this array typically record the iteration number until which a move is tabu.

6.1.5. Termination Criteria

In theory, the search could go on forever, unless the optimal value of the problem at hand is known beforehand. In practice, obviously, the search has to be stopped at some point. The

most commonly used stopping criteria in Tabu search are: after a fixed number of iterations (or a fixed amount of CPU time); after some number of consecutive iterations without an improvement in the objective function value (the criterion used in most implementations); or when the objective function reaches a pre-specified threshold value.

6.1.6. Probabilistic Tabu Search and Candidate Lists

Normally, one must evaluate the objective function for every element of the neighborhood $N(S)$ of the current solution. This can be extremely expensive from a computational standpoint. In probabilistic Tabu search, only a random sample $N'(S)$ of $N(S)$ is considered, thus significantly reducing the computational overhead. Another attractive feature is that the added randomness can act as an anti-cycling mechanism. This allows one to use shorter tabu fists than would be necessary if a full exploration of the neighborhood was performed. One the negative side, it is possible to miss excellent solutions. It is also possible to probabilistically select when to apply tabu criteria. Another way to control the number of moves examined is by means of candidate list strategies, which provide more strategic ways of generating a useful subset $N'(S)$ of $N(S)$. In fact, the probabilistic approach can be considered to be one instance of a candidate list strategy, and may also be used to modify such a strategy. Failure to adequately address the issues involved in creating effective candidate lists is one of the more conspicuous shortcomings that differentiate a naive Tabu search implementation from one that is more solidly grounded.

6.2. The Proposed Tabu Search Method

Over the last ten years, many researchers committed to the research of Tabu search, and published hundreds of paper presenting application of Tabu search, a heuristic method originally

proposed in [93]. In several cases, the methods described provide solutions very close to optimality and are among the most effective, if not the best, to track the difficult problems at hand. These successes have made Tabu search extremely popular among those interested in finding good solutions to the large combinatorial problems encountered in many practical settings. In this work, we apply Tabu search to a new setting – optimizing sampling rate.

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:

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

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 15).

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, was introduced in the context of the traveling salesman problem [92]. The neighborhoods defined by ejection chains provide the foundation for the more advanced levels of the solution method. Such neighborhoods are designed to produce moves of greater power with an efficient investment of computer 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 x_0 in the neighborhood of x (according to a neighborhood function $H(x)$) until a stopping criterion is satisfied or a predetermined number N of iterations is reached.

We developed an ejection chain algorithm (algorithm 3) for use it in the Tabu search framework to determine if a trial set of tracks from received observations during a surveillance period. We group the observations by time period, and 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. Figures 16 and 17 illustrate the method. Figure 16 is a set of observations Y between time $t1$ and $t3$. Figure 17 illustrates the process of searching for a trial set of tracks from the observations. The observations are divided into two groups. Figure 17(a) is a trial solution on the first group observations, Figure 17(b) is another trial solution on the second group observations, and Figure 17(c) is the final combined trial solution. This Divide-and-Conquer approach partitions an optimization problem into relatively independent sub-optimization problems, and accelerates the optimization process.

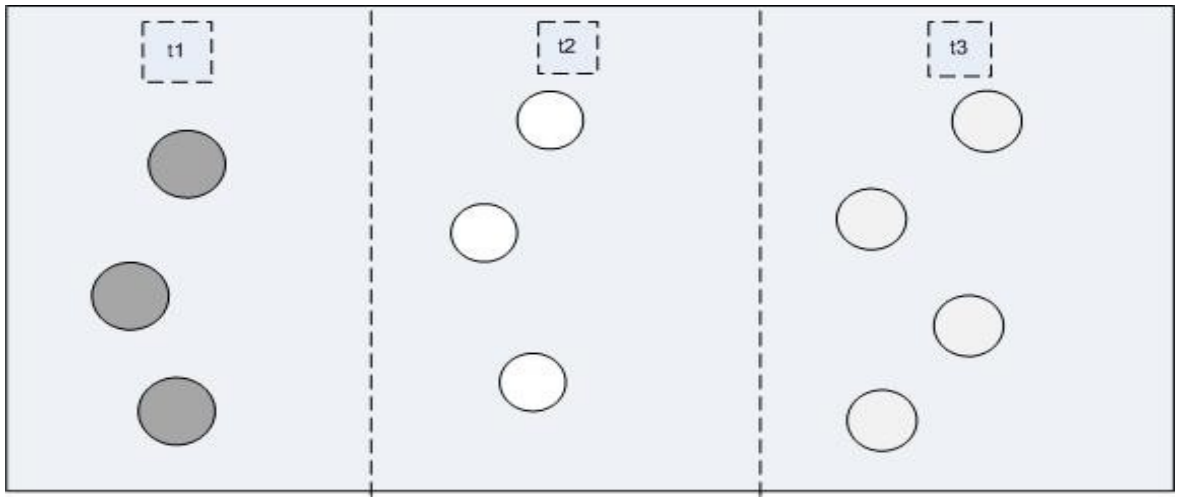


Figure 16. An example of a set of observations Y

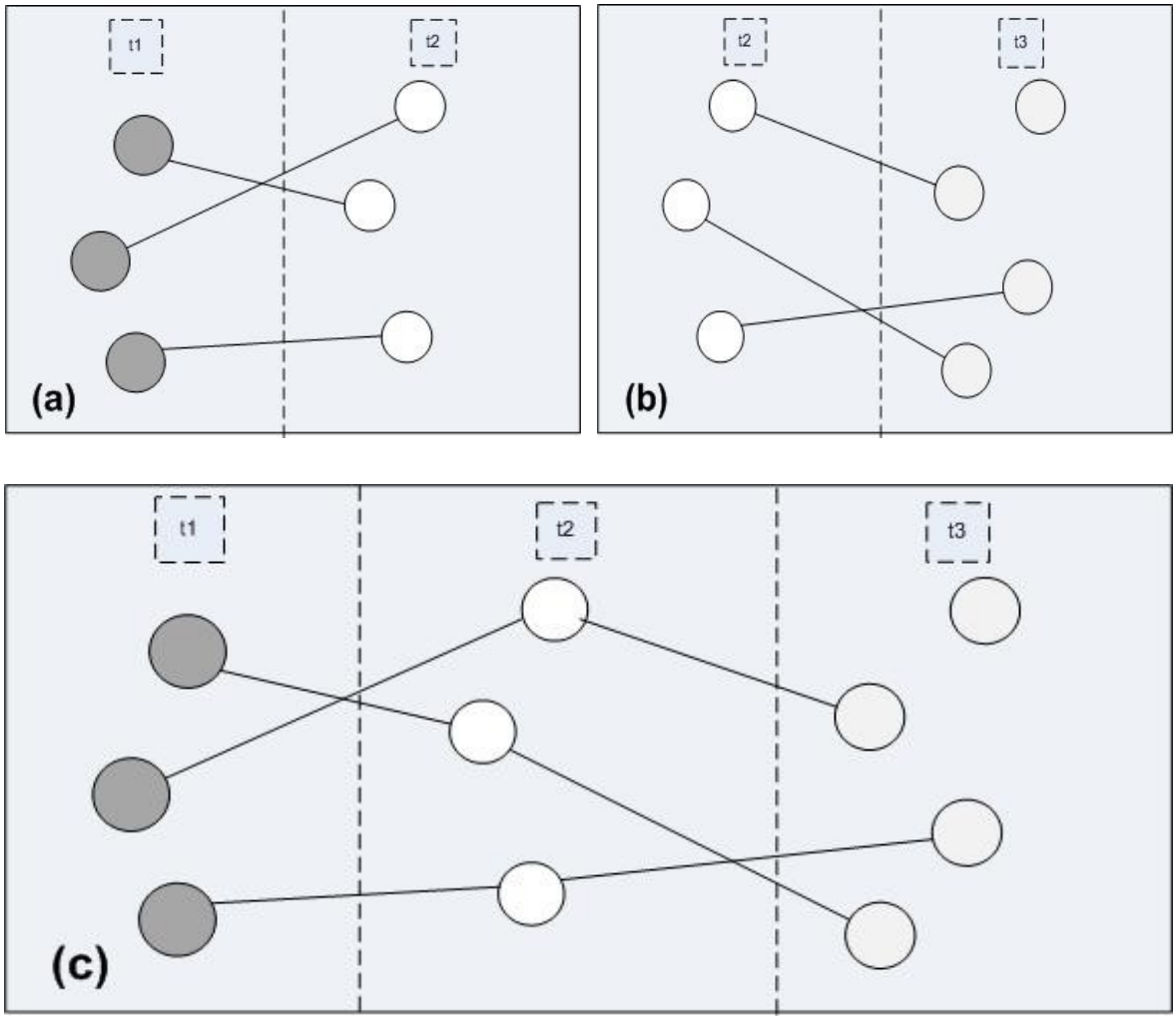


Figure 17. Illustration of searching for an optimal trial set of tracks

Combing MCMC with Tabu search on multi-target tracking is a significant feature on our work, and the simulation results will show that Tabu search is useful on keeping a low rejection rate.

Algorithm 3: Ejection chain algorithm (ECA)

$\boldsymbol{n1}$: number of observation in the first column

$\boldsymbol{n2}$: number of observation in the second column

$\boldsymbol{L}$: current Ejection level

$\boldsymbol{L^*}$: 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 e^L.
3. Increase the chain to further levels
    a. Set $L = L + 1$;
    b. Start from e^L, determine a new element that doesn't increase the cost
    c. Update current trial solution and e^L;
    d. ***if*** $L < Max\_LEVEL$ & $L < Min(n1, n2)$ go back to step 3, otherwise go to step 4
4. Get a new trial solution S';
5. ***Exit***

Algorithm 4: 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 ECA (Algorithm 3)
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***

CHAPTER 7. SIMULATION RESULTS

We performed experiments on Ns-2 (v2.33), a popular simulation tool, and the numerical computing environment MATLAB 7.0. The test area is a 750 by 750 square region, and a set of sensor nodes is randomly deployed in this area in each experiment. We conducted two groups of simulations to evaluate the efficiency of the proposed security scheme. The first group of simulations includes two experiments: one experiment to estimate the efficiency of the proposed immune-inspired algorithm on detecting malfunctioning nodes and another experiment to test the ability of defending Gray Hole attacks in a sensor network. The second group of simulations is conducted to assess the performance of the proposed multiple-target tracking algorithm on tracking intruders in a sensor network.

7.1. Misbehavior Detection Results

7.1.1. Flooding Attack Detection Results

7.1.1.1. The Network Lifetime Analysis

First, we generate a small size network to show how malicious nodes affect normal nodes' energy consumption. This network has 19 normal nodes and one malicious node, which attacks the network by sending high frequency packets to its neighboring nodes. The initial energy of each node is 500 (Joules), and the simulation time is 500 (seconds). Table 2 shows the average energy consumption of one normal node with and without this malicious node. The data show that this malicious node conducting a flooding attack can rapidly deplete a normal node's battery life.

Next, we generate a larger size network that has 50 nodes in the test area. When a sensor node runs out of battery life, normally or abnormally, it becomes a failed sensor node. We assume that a prescribed number of failed nodes will cause network failure and treat this as the

only factor affecting the network lifetime. We set the number of failed nodes that cause the network failure to 30 (60%). To estimate the efficiency of the DC-inspired algorithm, we sample the values of failed sensor nodes at different time slots, and compare the results with an experiment that simulates the proposed DCA and a parallel experiment in which no security mechanism is employed. We randomly deploy 5, 10 and 15 intruders in the test area and perform experiments on each case.

Table 2. Energy Consumption of a Normal Node

Time (Second)	Energy (Joules)	
	Zero malicious node	One malicious node
2.556954	497.442198	497.435126
50.016782	449.982866	412.499193
100.059589	399.940107	286.113860
150.009188	349.990460	151.504656
200.040950	299.958698	1.010425
250.002041	249.997607	0
300.012181	199.987515	0
350.024937	149.974711	0
400.008223	99.991473	0
450.000509	49.998771	0

Table 3 and Table 4 demonstrate the percent of failed nodes under different sampling times. Table 3 summarizes the results without security mechanism applied to the network and Table 4 shows the results when the network is equipped with the proposed DCA. It only needs about 80 seconds to make the percent of failed nodes reach 60% if the network has no security mechanism. But the percent of failed nodes only reaches about 36% when the sampling time is at 300 seconds. The network lifetime is obviously improved by the proposed security algorithm. Figures 18, 19, and 20 visually show the comparative data under different number of intruders in the network.

Table 3. Percent of Failed Nodes (without Security)

No. of Intruders	Sampling Time (Sec.)					
	20	40	60	80	100	120
5	2	6	13	18	79	96
10	1	27	40	62	89	97
15	2	30	51	69	93	98

Table 4. Percent of Failed Nodes (with Security)

No. of Intruders	Sampling Time (Sec.)					
	50	100	150	200	250	300
5	3	8	17	22	27	28
10	3	10	22	31	37	37
15	2	12	25	34	42	43

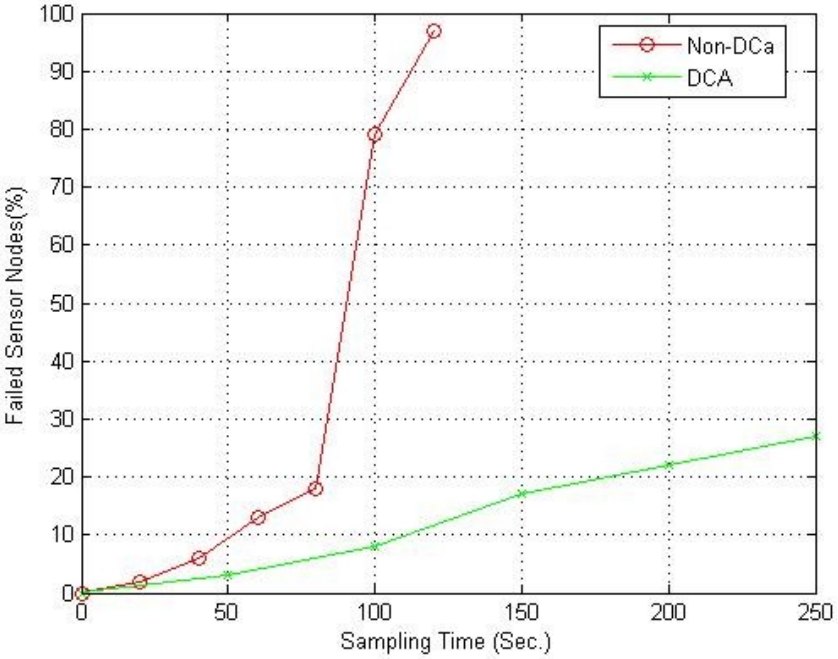


Figure 18. Sampling time vs. failed rate with 5 intruders

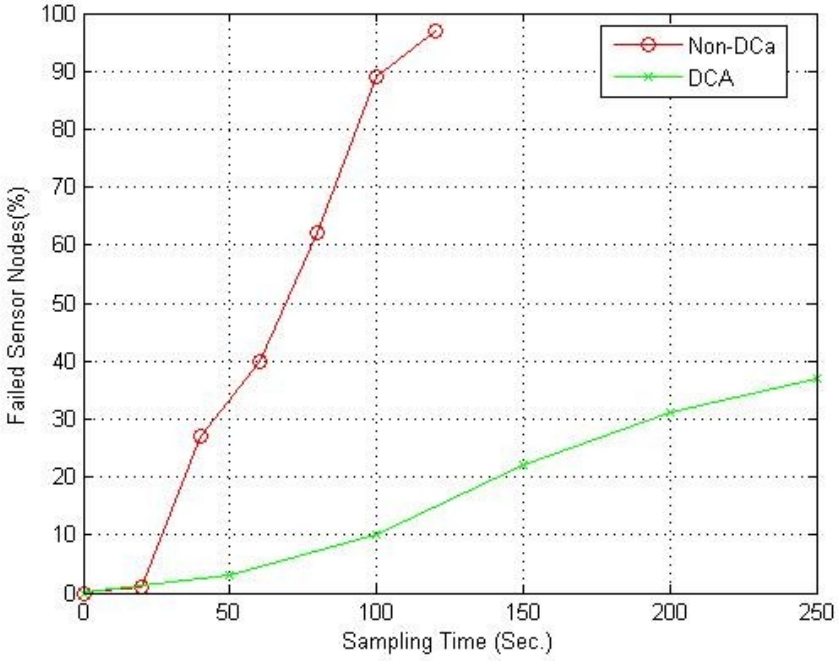


Figure 19. Sampling time vs. failed rate with 10 intruders

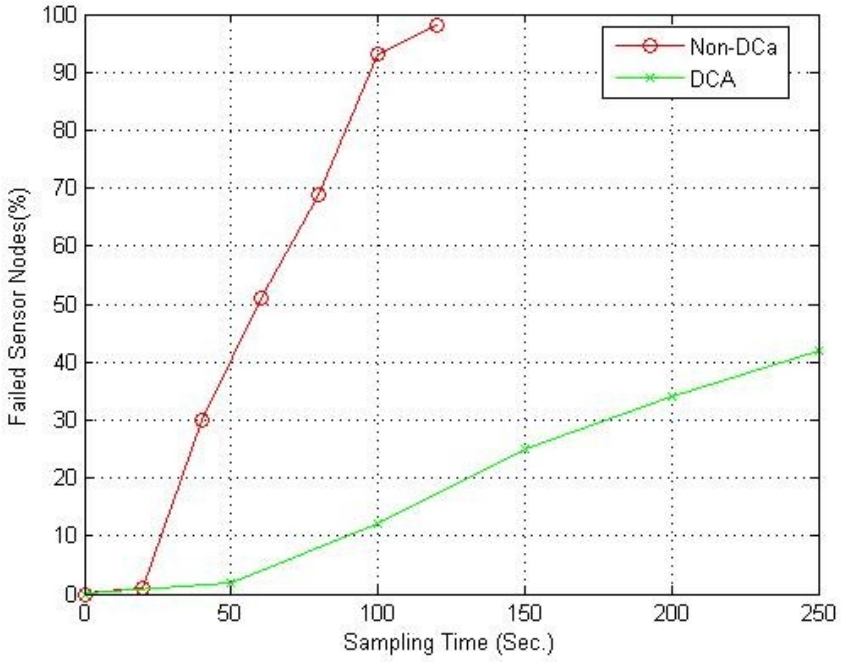


Figure 20. Sampling time vs. failed rate with 15 intruders

7.1.1.2. Impact of Cache Size

In the second group of experiments, we keep the same size of the test area and the network size as in the first group of experiments. Our aim is to measure the effect of the cache size on the detection rate of the DC-inspired algorithm. To detect a harmful intruder, a sensor node uses a cache to store the abnormal packets received from an intruder, and to monitor the intruder for a period of time to assess whether or not this is a harmful intruder. We set the cache size to 20, 40, 100, 200 and 300 units and perform experiments on each cache size.

Table 5 and Figure 21 show the experimental results. As expected, a larger size cache uniformly has a higher detection rate than a small cache. There is a flattening of the curves after a cache size of 150, indicating only marginal benefits of caches of size 200 or larger. When there are 5 intruders, the algorithm detects almost all of the harmful intruders with a cache size of 300.

Table 5. Intruder Detection Rate (%) vs. Cache Size

No. of Intruders	Cache Size (unit)				
	20	40	100	200	300
5	30	60	73	88	90
10	26	37	60	71	82
15	20	33	51	63	72

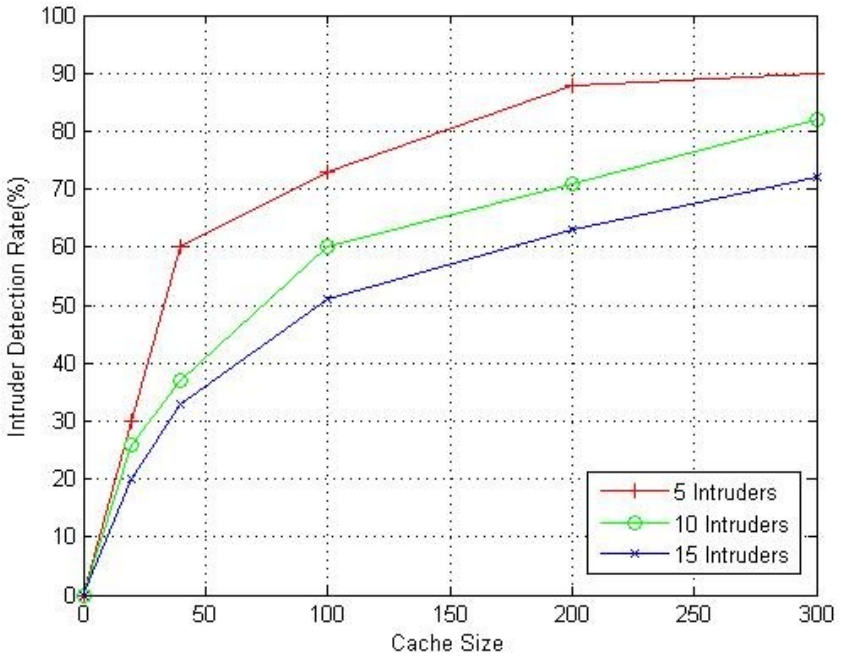


Figure 21. Cache size vs. detection rate

The above results also show that the procedure achieves a uniformly better detection rate for smaller numbers of intruders than for larger numbers, for any of the cache sizes. This is because the packets that a sensor node receives from harmful intruders are saved in the shared cache, and, when the cache is full, old packets will be removed from the cache, even though these packets may still be useful for detecting an intruder. Hence, the algorithm has a lower detection

rate when more intruders exist in the network. These experiment results indicate that cache size is an important factor for detecting harmful intruders. However, large cache sizes may be unrealistic in resource-limited low-end sensors.

7.1.1.3. Malicious Nodes Detection Rate

We randomly deploy 5, 10, and 15 intruders in the test area and perform experiments on each case. In each case, we take 10 samples. Table 6 and Figure 22 show the results for detecting malfunctioning sensor nodes.

Table 6. Intruder Detection Rate (%) vs. Sampling Time

No. of Intruders	Sampling Time (Sec.)					
	30	90	150	210	270	300
5	26	52	74	88	95	96
10	19	38	62	72	82	85
15	8	21	37	57	74	78

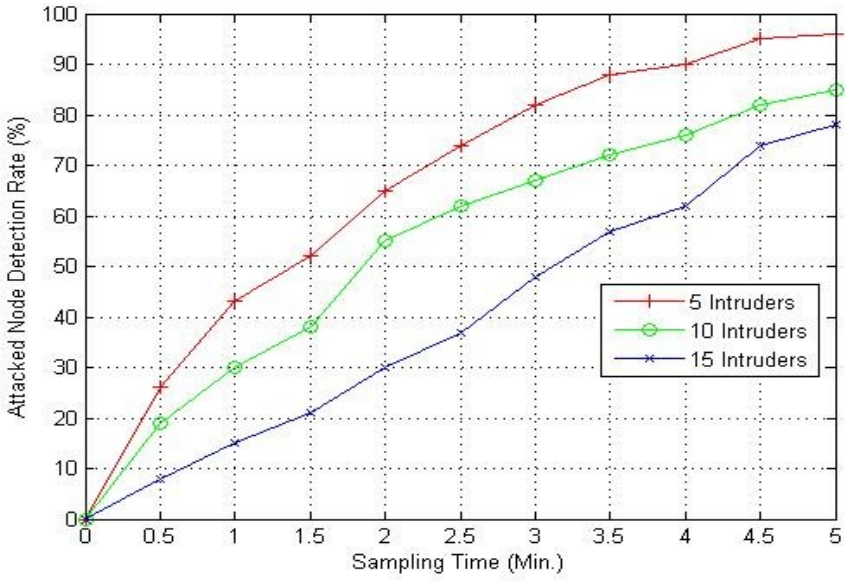


Figure 22. Sampling time vs. attacked node detection rate

The above results show the DCA 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 lower detecting rate at the beginning of the sampling times, and slightly decreased detecting rate when there are more harmful intruders.

7.1.2. Packet Dropping Attack Detection Results

After the malicious insiders that conduct packet dropping attack are detected using our proposed monitoring technique, we need to take further measures to maintain the normal operation of the network. We designed an improved packet routing method for preventing the packet dropping attack. In this section, we'll evaluate the performance of this method. The performance metrics include:

(1) Packet Delivery Ratio - the ratio representing the number of delivered data packet to the destination. This illustrates the level of delivered data to the destination. The greater value of packet delivery ratio means the better performance of the protocol. Its calculation equation is:

$$PDR = \sum Number\ of\ packet\ receivd / \sum Number\ of\ packet\ sent$$

(2) End-to-End Delay - the average time taken by a data packet to arrive in the destination. It also includes the delay caused by route discovery process and the queue in data packet transmission. Only the data packets that successfully are delivered to destinations are counted. Its calculation equation is:

$$EED = \sum (Arriv\ time - Send\ time) / \sum Number\ of\ connections$$

To investigate the effects of gray holes we simulated the wireless sensor network scenarios with and without gray hole nodes present in the network. The routing protocol we use

in the simulation is Dynamic Source Routing (DSR). We call our improved routing method as Optimal DSR (Op-DSR). To test the protocol DSR, we used two simulations. In the first scenario, we did not use any gray hole nodes and in the second scenario we added a gray hole node to the simulation. We then compared the results of the simulations.

We used UDP protocol in both simulations and attached CBR (Constant Bit Rate) application that generates constant packets through the UDP connection. CBR packet size is chosen to be 512 bytes, and data rate is set to 1 MB. Duration of the scenarios is 20 seconds and the CBR connections started at time equals to 1.0 seconds and continued until the end of the simulation. We manually defined appropriate positions of the nodes to show the data flow and also introduce a movement only to Node 1 to show the changes of the data flow in the network. A gray hole node is included in the network for the second simulation.

We used 20 nodes in the test networks and UDP connections are established between even and odd numbered nodes. In this setup the even numbered nodes are the sending nodes and odd numbered nodes are the receiving nodes. Each connection is represented by $r_{i \to j}$ (i the sending node, j the receiving node). For example, $r_{0 \to 1}$ represents that Node 0 is transmitting to Node 1. Node 18 and Node 19 are used as gray holes during the simulations as needed. Thus, we could count the sent and received packets between any two nodes. We could also count the number of packets dropped at each node including the gray hole nodes. In all the 20 scenarios we tested, the same nodes are acting as a source and sending to the same destination but in each scenario, every single node is placed at different coordinates and exhibits different movements. Node positions and movements are randomly generated. For each scenario, nodes move from a random starting point to a random destination with a speed that is randomly chosen. Total simulation time is set to 500 seconds and the CBR connections started at the first second of the

scenario and lasts for 450 seconds. We allowed 50 seconds for the buffers to be emptied after the transmission ends. In our scenarios CBR parameters are set to have the packet sizes of 512 bytes, and data rates of 10 KB/second.

For each scenario we performed two simulations. In the first one every node is working in cooperation with each other to keep the network in communication. The packet loss in an ad-hoc network without any malicious nodes is presented in Table 7. In the second, we introduced one malicious node that carries out the gray hole attack in the network. In this case Node 18 acted as a gray hole and Node 19 was silent. We measured the number of packets sent by the source node and received by the destination node. We also tried to evaluate how many packets that could not reach the destination node are absorbed in the gray hole. These data are also shown in Table 8.

Table 7. Average Packet Loss Percentage without Gray Holes (Using DSR)

Path	Packet sent	Packet received	% of packets lost
$r_{0\to1}$	974.2	931.7	4.3
$r_{2\to3}$	1012.07	983.2	3.2
$r_{4\to5}$	1020.67	976.21	4.3
$r_{6\to7}$	1007.2	956.2	5.0
$r_{8\to9}$	987.4	955.1	3.2
$r_{10\to11}$	995.3	968.24	2.7
$r_{12\to13}$	1029.7	982.8	4.6
$r_{14\to15}$	1009.78	984.3	2.6
$r_{16\to17}$	988.2	956.8	3.1
Total	9024.5	8695.0	3.6

We can see from Table 7 that DSR network has 3.6% packet loss without gray holes exist. But the percentage increases to 98.6%, shown in Table 8, when one gray hole exists in a network with 20 sensor nodes. Table 9 shows the results of packet loss when we use proposed routing method instead of the DSR protocol the average packet loss is 7.3%. Though it is still higher than 3.6%, it is much better than 98.6%.

Table 8. Average Packet Loss Percentage with Gray Holes (Using DSR)

path	Packet sent	Packet received	Packets dropped at gray hole 1	Packets dropped at gray hole 2	% of packets lost	% of packets lost at the gray holes
$r_{0\to1}$	1097	6	246	253	99.5	45.5
$r_{2\to3}$	1110	49	294	578	95.6	78.6
$r_{4\to5}$	1072	2	693	80	99.8	72.1
$r_{6\to7}$	1111	1	311	42	99.9	31.8
$r_{8\to9}$	1089	2	421	502	99.8	84.8
$r_{10\to11}$	1130	6	460	519	99.4	86.6
$r_{12\to13}$	1128	52	302	672	95.6	86.3
$r_{14\to15}$	1113	18	158	578	98.2	66.1
$r_{16\to17}$	1112	2	414	337	99.8	67.5
Total	9962	138	3299	3561	98.6	68.8

Table 9. Average Packet Loss Percentage with Gray Holes (Using Op-DSR)

path	Packet sent	Packet received	Packets dropped at gray hole 1	Packets dropped at gray hole 2	% of packets lost	% of packets lost at the gray holes
$r_{0\to1}$	978	898	33	27	8.1	75
$r_{2\to3}$	1006	930	27	20	7.5	61.8
$r_{4\to5}$	1009	950	17	22	5.6	66.1
$r_{6\to7}$	1008	923	30	18	8.6	56.5
$r_{8\to9}$	1028	952	23	22	7.4	59.1
$r_{10\to11}$	993	927	33	12	6.7	68.1
$r_{12\to13}$	988	916	20	13	7.3	45.9
$r_{14\to15}$	986	908	22	25	7.9	60.1
$r_{16\to17}$	985	917	14	24	6.8	55.9
Total	8981	8321	219	183	7.3	60.1

Next, we'll analyze the performance of the Op-DSR on the packet delivery ratio and end-to-end using DSR as the baseline. We did five groups of simulation in different sizes of network. Each network has 10% gray holes. The results are shown in Table 10 and Table 11.

Table 10. Packet Delivery Ratio (PDR)

No. of Nodes	10	20	30	40	50
DSR	81.5	95.2	97.6	98.2	98.7
Op-DSR	78.2	88.6	83.7	93.2	89.8

Table 11. End-to-End Delay (Sec.)

No. of Nodes	10	20	30	40	50
DSR	3.3645	9.43356	2.64367	15.4327	5.37654
Op-DSR	3.4471	8.76216	7.52374	9.64372	3.26583

Table 10 shows the results of packet delivery ratio. The Op-DSR can maintain an acceptable packet delivery ratio. Table 11 shows the results of end-to-end delay. It also can maintain a similar level of EED as DSR does.

7.2. Misbehavior Monitoring Results

To investigate the impact of Bayesian game theory based monitoring strategies on detecting malicious insiders. We simulate a network with the same size testing space and 50 mobile nodes. The routing protocol we use is Dynamic Source Routing (DSR) and the routing cache is path cache with a primary and a secondary FIFO cache. The probing (querying) technique is implemented as a part of DSR. The simulation time is 100 seconds. The mobile nodes move within the network space with a maximum speed of 20.0 m/s. The pause time is 50 seconds, which assures that the topology moderately changes. The communication patterns are 10 constant bit rate (CBR) connections with a data rate of 4 packets per second. We randomly choose 0, 3, 6, 9, 12, and 15 malicious nodes in each of the simulations.

The five metrics we choose for measuring the proposed monitoring strategies technique are: (1) malicious Node Detection Rate, the ratio of the number of detected malicious nodes and the total number of actual malicious nodes; (2) false Position Rate, the ratio of number of normal nodes mistakenly detected as malicious nodes and the total number of normal nodes; (3) packet

Delivery Rate, the ratio of total number of data packets received and the total number of data packets sent in application level; (4) network Overhead, the ratio of total number of routing related transmissions and the total number of packet transmissions. Each packet hop is counted as one transmission; and (5) extra Energy Consumption Rate, the ratio of the amount of energy used for probing other nodes and the amount of energy used for normal network communication.

We study the proposed monitoring technique using the chosen metrics. The standard DSR (Standard_DSR) is used as baselines to compare with our proposed dynamic probe DSR (DSR_DProbe). We run the simulation three times and the averaged data are shown in the flowing tables and figures. Table 12 and Figure 23 show the detection rate. Table 13 and Figure 24 show the false positive rate. Table 14 and Figure 25 show the packet delivery ratio. Table 15 and Figure 26 show the percentage of network overhead.

Table 12. Detection Rate

No. of Malicious Node (%)	0	0.06	0.12	0.18	0.24	0.30
Detection Rate (%)	100	93	91	83	84	80

Table 13. False Positive Rate

No. of Malicious Node (%)	0	0.06	0.12	0.18	0.24	0.30
False Positive Rate (%)	2	7	7	6	8	7

Table 14. Packet Delivery Ratio (PDR)

	No. of Malicious Node (%)					
	0	0.06	0.12	0.18	0.24	0.30
DSR-DProbe	100	92	90	85	78	76
Standard-DSR	100	82	83	72	68	60

Table 15. Network Overhead (%)

	No. of Malicious Node (%)					
	0	0.06	0.12	0.18	0.24	0.30
DSR-DProbe	17	16	15	20	24	26
Standard-DSR	4	5	7	6	8	5

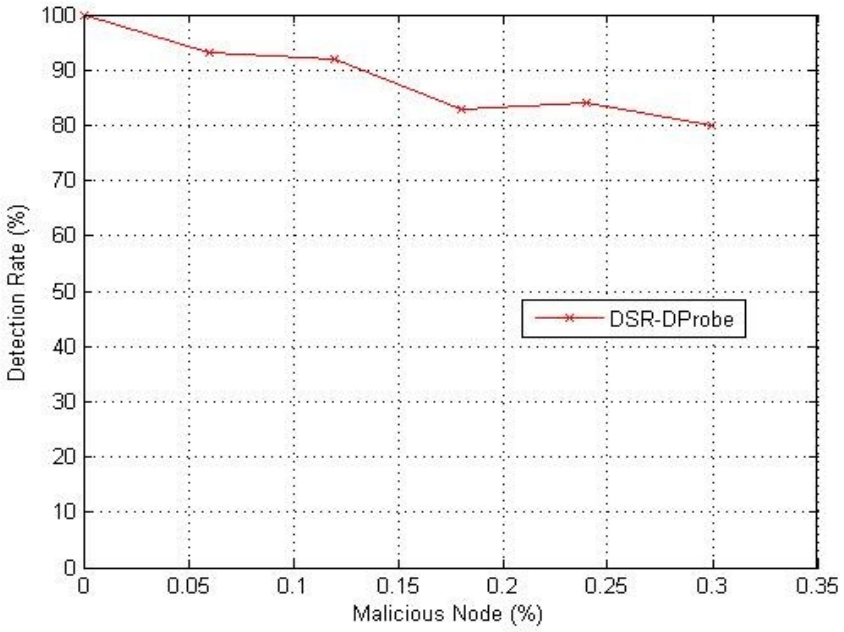


Figure 23. Detection rate

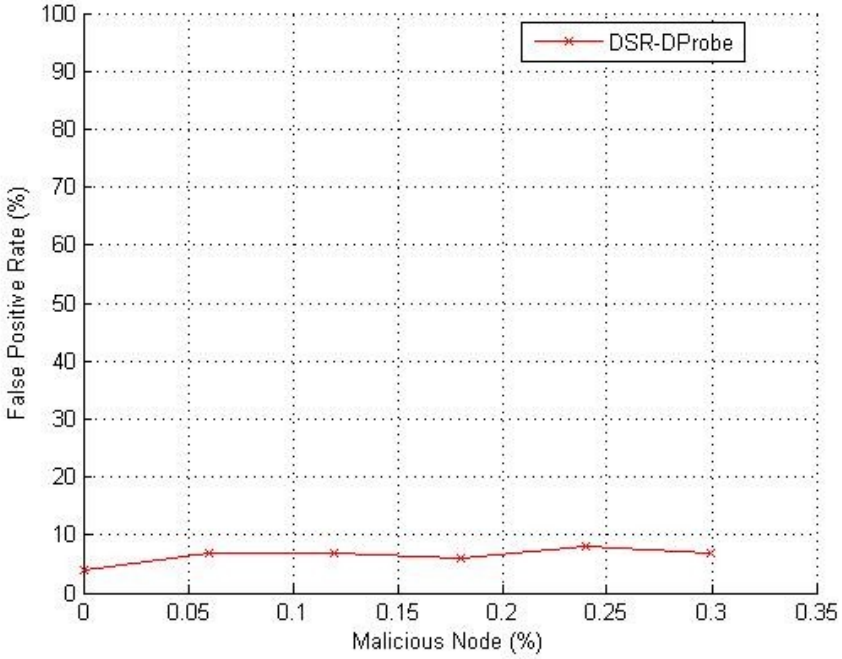


Figure 24. False positive rate

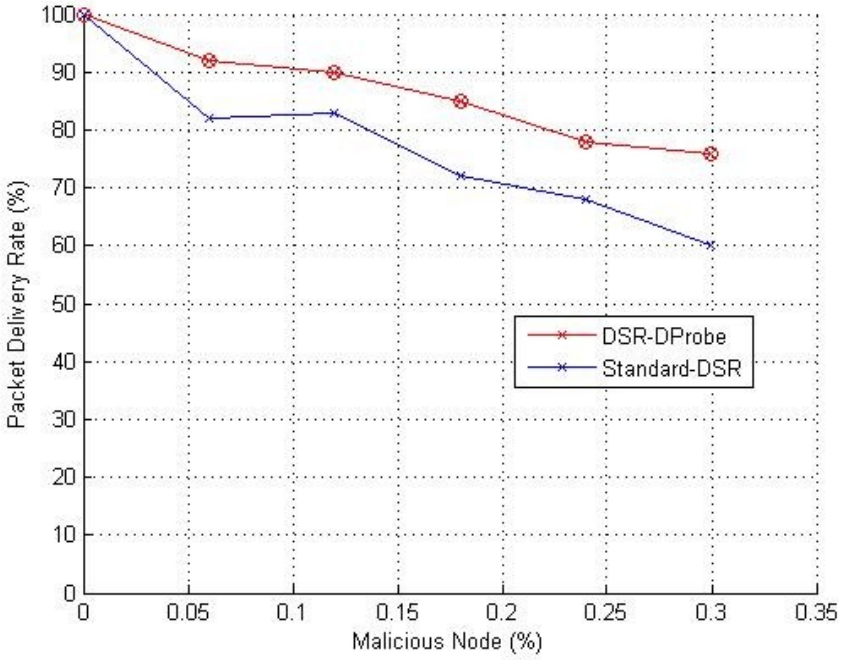


Figure 25. Packet delivery rate (PDR)

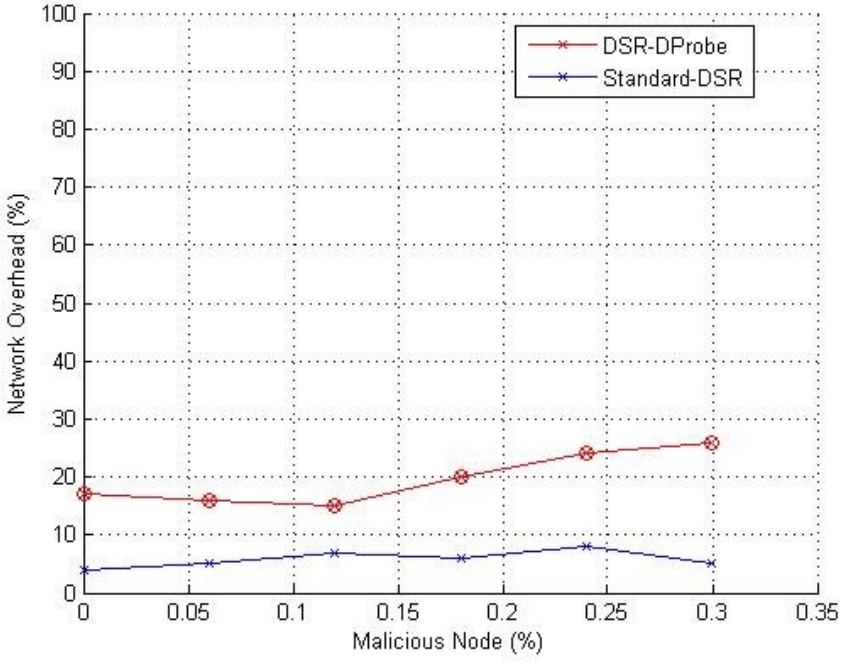


Figure 26. Network overhead

Table 12 shows the malicious node detection rate, which is averagely about 86 percent. Since malicious nodes don't drop packets with fixed frequency, it will have higher detection rate if the simulation time is longer. The false positive rate shown in Table 13 is less than 10 percent. It is mainly caused by the movement of nodes. Table 14 shows the packet delivery rate. Since malicious nodes are timely detected, our proposed method has better performance than the standard DSR. Table 15 shows the network overhead. The change speed of the network topology has obvious effect on the network overhead. When the network topology changes faster, more probe messages have to be sent out to identify dropped packets.

The extra energy consumption is proportional to the number of probe packets, so we count how many probe packets are used for detecting malicious nodes to study this metric. In this simulation, we choose 9 malicious nodes in the network, and compare our dynamic probe

technique with the periodic probe technique in [27] on the extra energy consumption during the process of detection. The extra energy consumption results are shown in Table 16 and Figure 27.

Table 16. Extra Energy Consumption Rate

	Sampling Time (Sec.)					
	0	20	40	60	80	100
DSR-Probe	0	10	20	30	40	50
DSR-DProbe	0	4	5	18	22	15

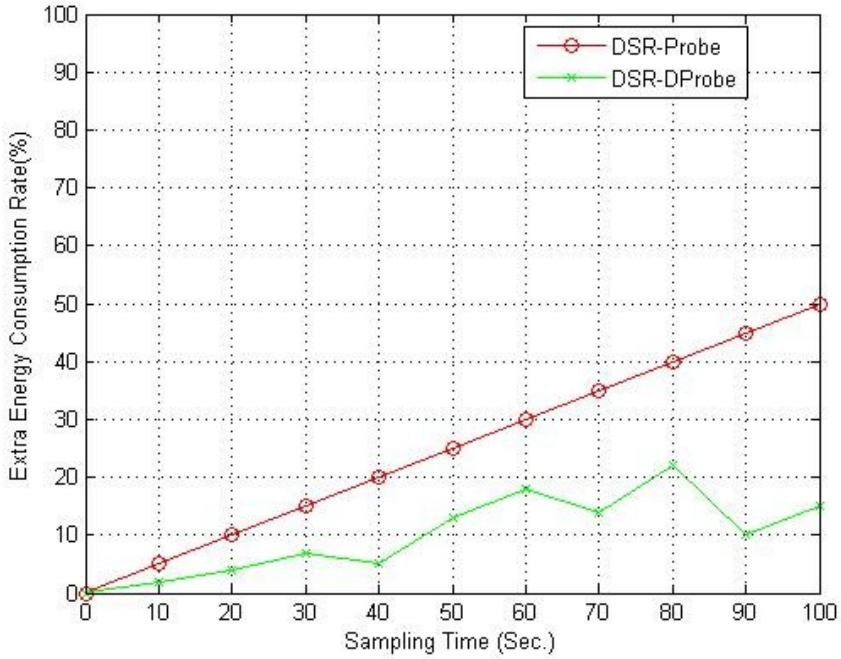


Figure 27. Extra energy consumption rate

The dynamic probe technique has similar detection rate with the probe technique in [27]. However, our method has obvious higher energy efficiency than the periodic probe technique.

Table XVIII shows that DSR-Prob method consumes 10% energy on the probing message, and our DSR-DProb method only consumes 4% energy at the sampling time 20sec. It's about 60% improvement on energy efficiency. And the average energy efficiency is about 65%.

7.3. Multi-target Tracking Results

In this experiment, we utilize the similar simulation settings as in [8]. 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 100. The other parameters are: $T = 50$, $p_d = 0.85$, $\lambda_f V = 1.0$, $\lambda_b V = 1$, $p_z = 0.01, \overline{d} = 5, \overline{v} = 5$ moving 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 a target, and the models are:

$$x_t = Ax_{t-1} + Bw_k$$

$$y_t = Hx_t + v_t$$

where

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

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

We adopt some algorithm-free metrics to evaluate the efficiency of the improved tracking algorithm. In real scenarios truths, or real targets' tracking, are not available. In this situation, the consistency of tracking results may be checked. Our work focuses on the other situation in which truths are available.

7.3.1. Trajectories Generation & Scenario Setup

Figure 29-32 shows a scenario sample, in which 10 mobile targets appear during a period of supervision. The moving duration of each target under the surveillance period is shown in Table 17, and Figure 28 shows the number of appeared targets at each time step. The trajectories of these mobile targets functioning as available truths are shown in Figure 29. The received measurements or observations over the monitoring period are presented in Figure 30. Figure 31 is a snapshot taken as tracking is performing, in which the solid lines represent obtained target tracking at the moment when the snapshot was taken, and the dotted lines represent received new measurements. The final tracking results are shown in Figure 32.

Table 17. Moving Duration of Each Target under a Surveillance Period [1, 50]

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
Initial (t_a)	15	16	2	6	2	14	7	12	21	8
Terminal (t_b)	45	50	36	41	36	50	50	50	50	37

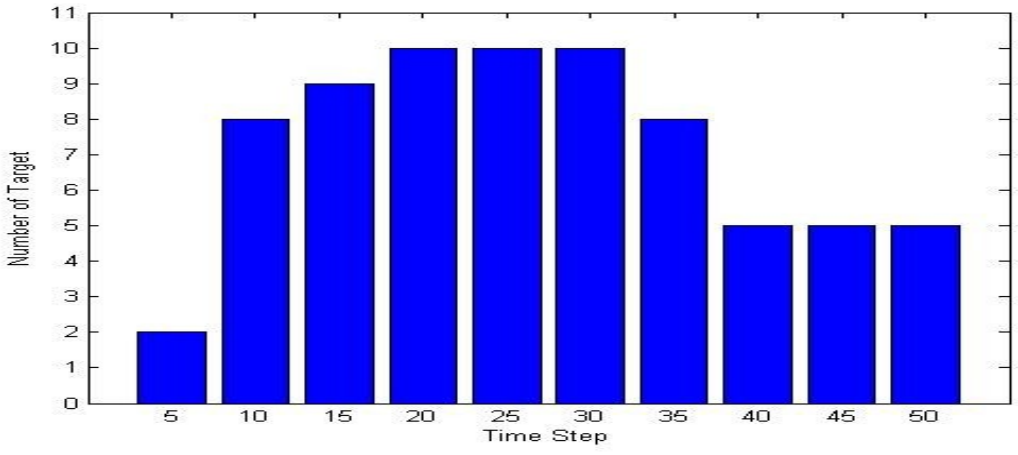


Figure 28. Appeared targets at each time step

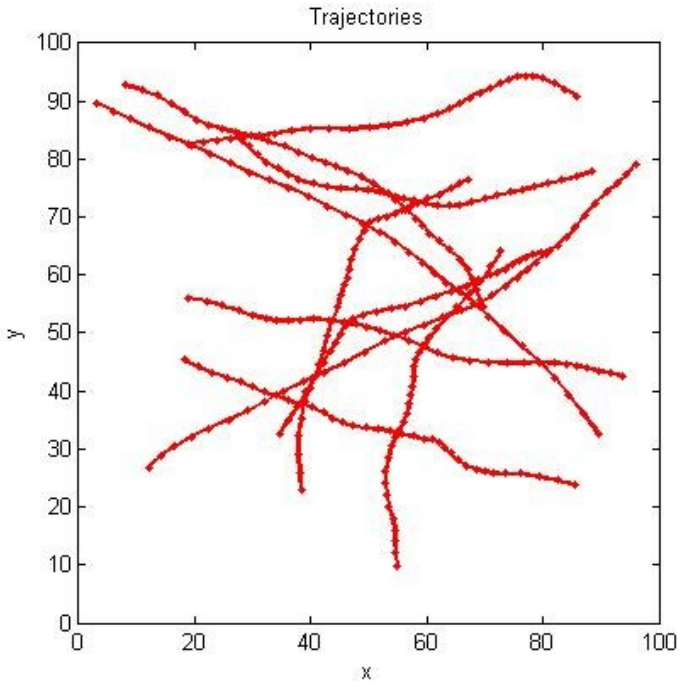


Figure 29. Trajectories (K = 10)

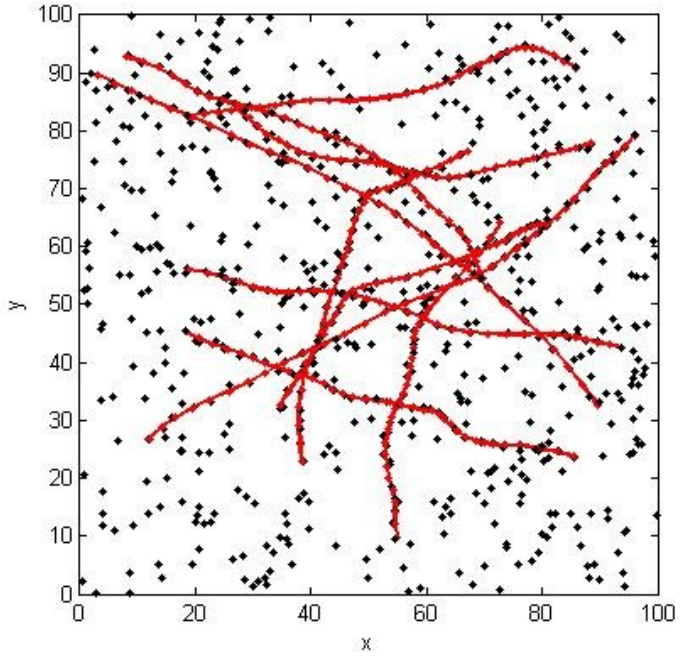


Figure 30. Cluttered measurements based on the trajectories

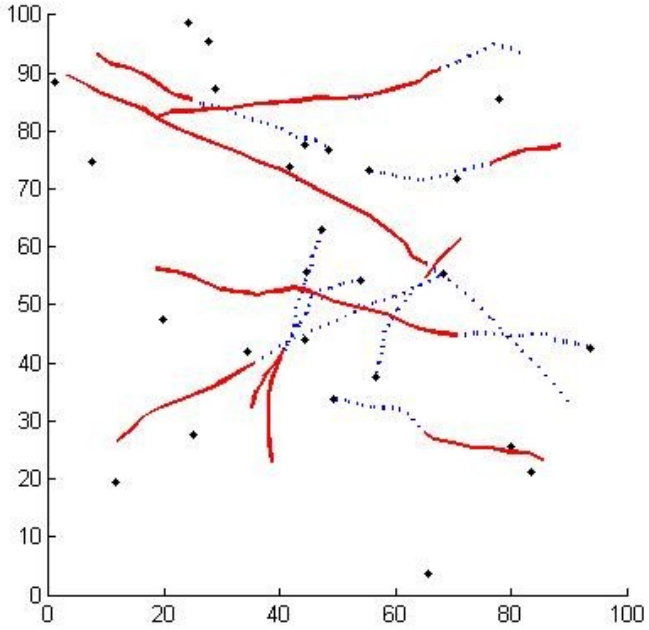


Figure 31. A snapshot throughout the tracking

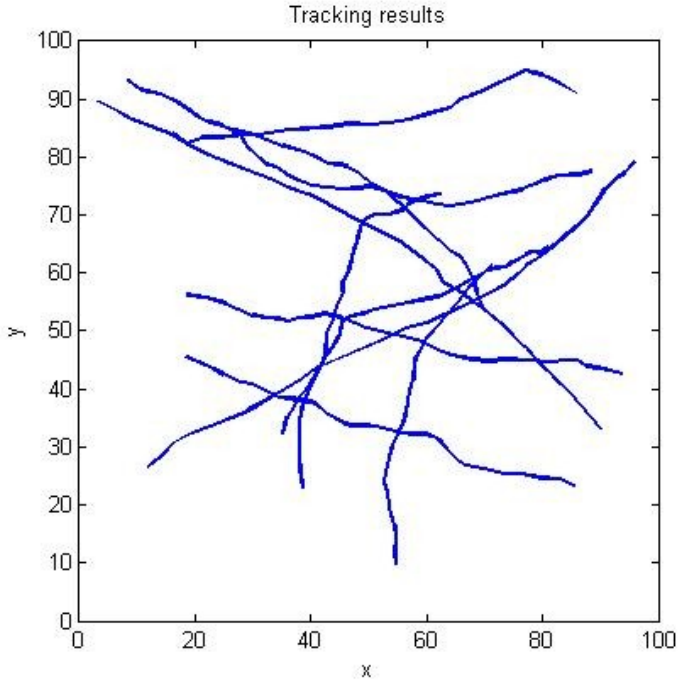


Figure 32. The tracking results

The accuracy of the proposed multi-target tracking algorithm can be visualized from the above four figures. Next, we'll use data to further demonstrate the efficiency of the proposed algorithm on running time.

7.3.2. Evaluation metrics

7.3.2.1. Running Time

Computational cost is another important factor that should be taken in performance evaluation of tracking algorithms. Though MHT provides accurate and optimal results, it suffers from the huge computational cost. Therefore, for every tracking algorithm, the total time needed in order to run the tracker is represented as the total execution time. This metric may be really important in practice where trackers are supposed to be applied to the real time problems.

The three lines in the Figure 33 represent CPU running time with the sample size as 2000, 3500, and 5000 (10 targets, 50 observing time steps). The corresponding data are shown in Table 18. For each line, the running time changes with the number of appeared targets at different observing time steps. Figure 33 shows that the execution time of a tracking algorithm is proportional to the sample size (or number of samples). According to statistics, the larger the sample size is, the more chances the selected samples have to be like the average value. So it seems impossible to find an optimal solution using both smaller number of samples and shorter running time. However, we can indirectly solve this problem through improving the quality of the selected samples. In other words, we can try to select only those samples that are more likely representative of the population instead of random selection. We will show the effects of the mentioned sampling strategy on target tracking as below.

Table 18. Running Time (Sec.) with Different Sample Size

Time Step	Sample Size		
	2000	3500	5000
0	0	0	0
5	0.0312	0.0312	0.0624
10	0.0624	0.0781	0.1248
15	0.1248	0.2028	0.2964
20	0.156	0.2808	0.39
25	0.1716	0.312	0.4524
30	0.2184	0.3432	0.4992
35	0.234	0.39	0.546
40	0.234	0.3744	0.546
45	0.2028	0.3588	0.4836
50	0.2028	0.3276	0.4524

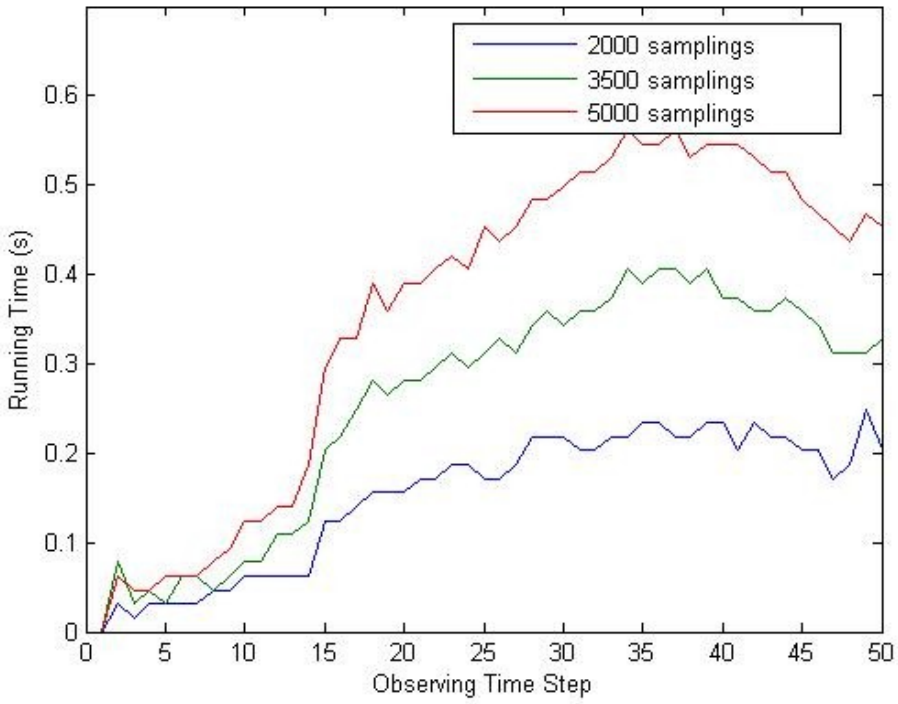


Figure 33. Time step vs. running time

7.3.2.2. Sequence Tracking Detection Accuracy–Distance (STDA-D)

We adopt the metric "Sequence Tracking Detection Accuracy–Distance" (STDA-D) proposed in [98] to evaluate track accuracy. STDA-D is a spatio-temporal based measure penalizing fragmentation in the temporal and the spatial domains. To compute the STDA-D score, one needs to compute one-to-one match between the tracked targets and the ground truth targets. Given M matched tracks including tracked targets $\tau_k(i)$ and the corresponding ground truth tracks $G_k(i)$, k =1, … , M, i = 1, … , T. The formula of STDA-D is

$$STDA - D = \sum_{k=1}^{M} \frac{\sum_{t=1}^{T}(1 - d_t^{'})}{N_{frame}\ (G_k \cup \tau_k\ \neq \emptyset)} \Big/ \left(\frac{N_G + N_T}{2}\right)$$

where the denominator for each track $N_{frame}\ (G_k \cup \tau_k\ \neq \emptyset)$ indicates the number of frames in which either a ground truth or a tracked target (or both) is present. The numerator for each track

measures the spatial accuracy by computing the overlap of the matched tracking results over the ground truth targets in the sequence. The normalization factor is the average of number of tracked targets N_T and the number of ground truth targets N_G. STDA-D produces a real number value between 0 and 1 (worst and best possible performance respectively).

The MCMCDA algorithm exhibits remarkable performance comparing to other association algorithms like MHT under extreme conditions, such as a large number of targets in a dense environment, low detection probabilities, and high false alarm rates [8]. In this experiment, instead of tracking 10 targets, we increase the number to 50. The number of frames or observing time steps is still 50. The data in Table 19 and the lines in Figure 34 show that the MCMCDA with new sampling strategies needs a smaller sample size to achieve the same STDA-D compared to the original MCMCDA.

Table 19. Sample Size vs. STDA-D

	Size of Sample (10^3)						
	2	4	6	8	10	12	14
I-MCMCDA	0.72	0.77	0.78	0.82	0.88	0.9	0.92
MCMCDA	0.64	0.7	0.75	0.79	0.8	0.81	0.83
MHT	0.18	0.2	0.22	0.3	0.39	0.41	0.42

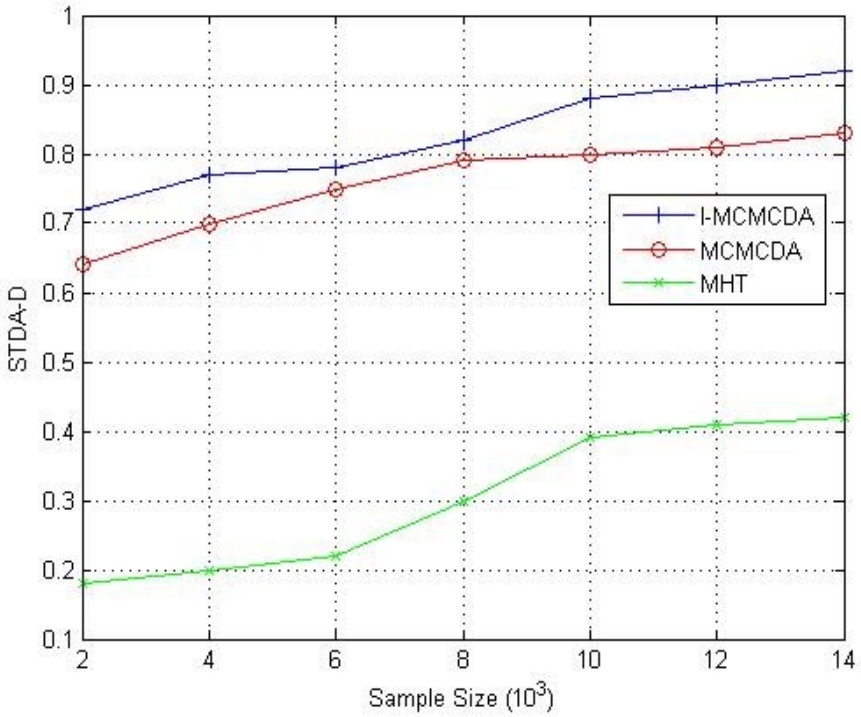


Figure 34. Sample size vs. STDA-D

7.3.2.3. STDA-D vs. NUMBER OF Targets

In this experiment, we vary the number of targets from 10 to 100. The other parameters are fixed: $\mathcal{R} = [0,\ 100] \times [0,\ 100]$, $T = 50$, $p_d = 0.85$, $\lambda_f V = 1.0$, $\lambda_b V = 1$, $p_z = 0.01$, $\overline{d} = 5, \overline{v} = 5$ unit lengths per unit time. Since all targets are observed, the number of observations increases as the number of targets increases. The results for the three data association algorithms are the average values over 10 repeated runs and 10,000 samples are used. The average STDA-D for three different algorithms is shown in Table 20 and Figure 35. MHT indicates deteriorated performance with increasing number of targets due to pruning. However, MCMCDA and the improved version maintain good performance.

Table 20. Number of Target vs. STDA-D

	Number of Targets									
	10	20	30	40	50	60	70	80	90	100
I-MCMCDA	.98	.96	.90	.83	.80	.74	.70	.64	.61	.58
MCMCDA	.97	.94	.87	.82	.79	.70	.68	.62	.58	.54
MHT	.90	.85	.56	.40	.26	.17	.15	.01	.08	.03

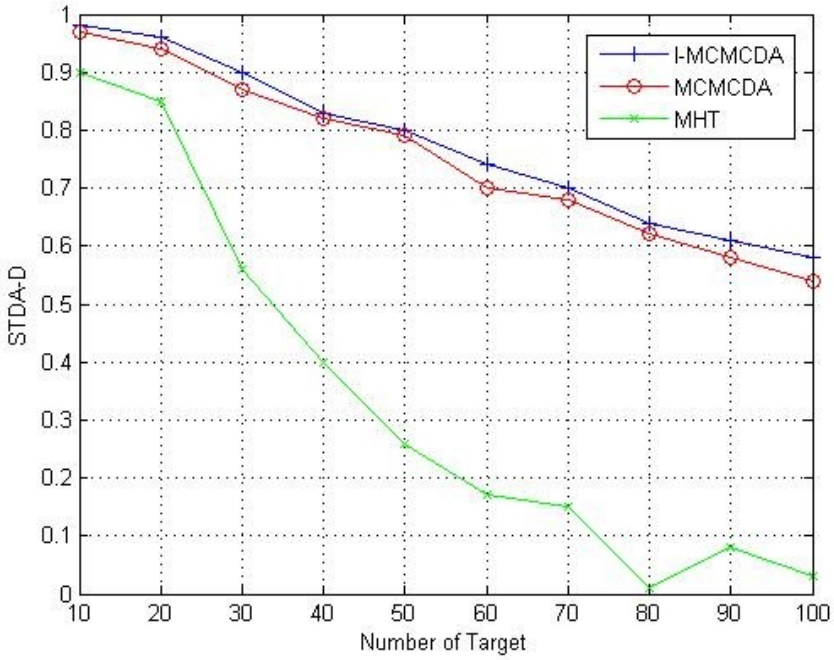


Figure 35. Number of targets vs. STDA-D

7.3.2.4. Running time vs. Number of Targets

To demonstrate the efficiency of the proposed multi-target tracing algorithm on searching of optimal tracks, we conducted five groups of simulations with different numbers of targets. There were three experiments in each group, and three different multi-target tracking algorithms I-MCMCDA, MCMCDA, and MHT were separately implemented. Table 21 demonstrates the

average running time of these algorithms. I-MCMCDA and MCMCDA have much better running speed than MHT. The average running time of MCMCDA and I-MCMCDA is similar because they run the same number of samples. In fact, I-MCMCDA needs less number of samples to find an optimal track than MCMCDA because we adopted the state space reduction mechanism and the Ejection Chain algorithm to accelerate the searching of optimal tracks. Figure 36 displays the expected results.

Table 21. Average Running Time (Sec.)

	Number of Targets				
	10	20	30	40	50
I-MCMCDA	35.88	67.021	102.23	266.45	502.47
MCMCDA	35.01	60.132	138.02	240.345	550.33
MHT	37.19	150.33	297.65	587.56	1243.32

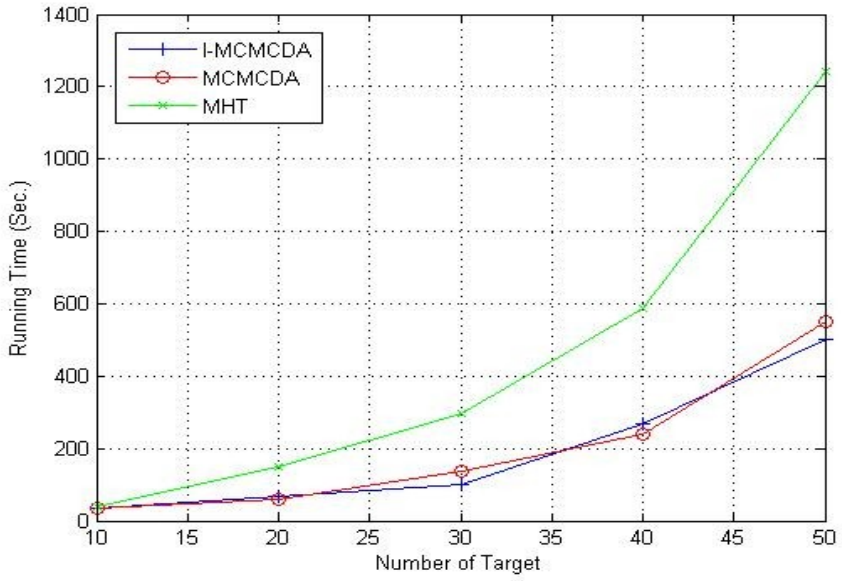


Figure 36. Average running time vs. number of targets

CHAPTER 8. CONCLUSION AND FUTURE WORKS

Wireless Sensor Networks have been applied to many different application domains because of the attractive characteristics of sensors: small in size, easy deployment, and cheap on price. At the same time, these attractive characteristics also present challenges on designing convenient and effective security mechanisms that fit for wireless sensor networks. With a single security technique, it is hard to secure sensors from variety types of attacks. This dissertation explored some different techniques on preventing attacks in WSNs and combined them to design a two-phase security mechanism that is cable to provide higher level of security than any single one of the techniques. It is one of the distinguishing features of our work to integrate multiple techniques into one effective protection mechanism. This protection mechanism is for preventing attacks from insiders and outsiders, especially flooding and packet dropping attacks in WSNs. The used techniques include a danger theory inspired Dendritic Cell algorithm, a Bayesian game-based monitoring scheme, a Markov Chain Monte Carlo based targets tracking technique, and a Tabu search based sampling method.

Each normal sensor node equipped with the DCA is responsible for detecting malicious sensor nodes and preventing their damage to a WSN by ceasing to respond to any requests from these nodes. Comparing with other immune-inspired intruder detection techniques, e.g., negative selection algorithm, the danger theory inspired Dendritic Cell algorithm has unique advantages. Since the DCA detects malicious nodes based on danger signals (divergent events) instead of only checking central database of blacklist, it is more capable of identifying new harmful intruders than negative selection algorithm. So DCA is more adaptable for sensing unpredictable environments. Also the DCA doesn't have complicated computation. It is a light resource consumption algorithm, which makes it possible to equip each sensor node with this algorithm.

Normally, a central control structure is difficult to apply to a wireless sensor network, especially a large one. Therefore, it is more applicable to run security algorithm on each sensor node itself. The designed DCA meets this requirement.

To prevent general gray hole attack that randomly or strategically drop received packets, we first adopt Bayesian game theory based monitoring strategy to detect the attackers, and then we implement the proposed short-and-safe routing protocol to mitigate the packet dropping attack. The fundamental purpose of using this technique is to make sensor nodes productively use their energy while detecting malicious nodes. The packet dropping attack is different from a flooding attack. Without a monitoring mechanism, the hidden attackers are difficult to detect. Of course, real-time monitoring is an ideal method. However, it is not practical to let energy-limited sensors do real-time monitoring. Some works have been proposed to let sensors do periodic monitoring. But, what monitoring cycle should be chosen to achieve an optimal detection rate? It is not easy to answer this question. Instead of using periodic monitoring, we adopt a Bayesian game based monitoring method. This monitoring method makes a normal node able to dynamically adjust its monitoring strategies based on the attacker's danger level, which can be obtained from the designed DCA. This Bayesian game based monitoring method lets a normal node make an optimal monitoring strategy without knowing when the attacker drops packets. The obvious advantage of the dynamic monitoring method is the ability to reduce extra energy consumption, which is always a precious resource of sensor nodes, without losing detection rate. The above mentioned techniques all have strong adaptability because they can dynamically adjust their control parameters with a change of the external environment. This is an important criterion for designing a security algorithm.

Besides preventing malicious insiders, the proposed two-phase security mechanism is also used to prevent malicious outsiders, which are the true culprits of the damage to the WSNs security. The intruders may indirectly attack a WSN through compromised nodes. If they are not be tracked down, sometimes it is hard to maintain the security of a WSN. The intruders can compromise more normal nodes, and use the compromised nodes to attack the WSN. Therefore, it is necessary to design a multi-target tracking algorithm to improve the security of a WSN. Multi-target tracking deals with the state estimation of an unknown number of moving targets. The main difficulty comes from the assignment of a given measurement to a target model. The data association problem is to work out which measurements were generated by which targets. It is the key step to successfully track down the mobile targets. The multiple hypothesis tracker (MHT) algorithm, a popular multi-scan tracking algorithm, can lead to an NP-hard problem because the number of possible associations increases exponentially with time. So we adopt another data association algorithm, Markov Chain Monte Carlo Data Association (MMCDA), to do the multi-target tracking. MCMCDA is a true approximation scheme for the optimal Bayesian filter; i.e., when run with unlimited resources, it converges to the Bayesian solution. As the name suggests, MCMCDA uses Markov Chain Monte Carlo (MCMC) sampling instead of summing over all possible associations. The purpose of the adopted target tracking technique is to track down harmful outsiders soon. So the convergence rate of MCMC sampling to reach an optimal solution is an important factor on evaluating the effectiveness of a multi-target tracking algorithm.

To accelerate the convergence rate, we make some improvements. Firstly, we classify the measurements under the assumption that the measurements from harmful outsiders and normal outsiders can be distinguished. Secondly, we roughly predict a target's moving direction at some

time by analyzing the change of signal intensity, which information comes from different normal nodes. These two improvements decrease the search space; thereby speeding up the convergence rate. The last improvement is to design an ejection chain algorithm and employ a Tabu search technique to do the MCMC sampling. Instead of randomly taking samples from reported observations (measurements), which may lead to high rejection rate, Tabu search starts from an initial solution and evolves that single solution into a iteratively improved solution. Comparing to other local search techniques, Tabu search can prevent premature convergence to local optima. We group the observations, apply the Tabu search method in each group, and then combine the sub-solutions. This approach partitions an optimization problem into relatively independent sub-optimization problems, and accelerates the optimization process. This is one of the contributions of our work.

Each of the proposed security techniques in this dissertation has a certain ability on defending attacks in WSNs, and can be independently applied to different network security environments. It is another distinguishing feature of our work to adopt modular design concept on designing the security mechanism.

The simulation results demonstrate that the proposed two-phase security mechanism can effectively improve the security of a WSN by promptly identifying internal and external trouble makers.

We designed a DCA for detecting malicious insiders and anomaly events in WSNs. The efficiency of this algorithm mainly depends on the ability of identifying danger signals. How to precisely identify danger signals is a topic that needs to be further researched. How to find a heuristic solution faster on sampling is another future work to do. We consider designing an improved Ejection Chain Algorithm, which should have variable-exchange (i.e., n-opt) ability.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” IEEE Communications Magazine 40(8): pp. 102-114, Aug. 2002.

[2] J. P. Walthers, Z. Liang, W. Shi, and V. Chaudhary, “Wireless sensor networks security: a survey,” Technical Report MIST-TR-2005-007, July 2005.

[3] J. Kim and P. J. Bentley, “Evaluating negative selection in an artificial immune system for network intrusion detection,” Genetic and Evolutionary Computation Conference (GECCO-2001), San Francisko, pp. 1330-1337, July 2001.

[4] S. Forrest, S. Hofmeyr, and A. Somayaji, “Computer immunology,” Communications of the ACM, 40(10): pp. 88-96, 1997.

[5] 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.

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

[7] 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, pp. 16–23, 2005.

[8] 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.

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

[10] D. B. Reid, “An algorithm for tracking multiple targets,” IEEE Transaction on Automatic Control, 24(6), pp. 843–854, December 1979.

[11] 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

[12] H. Yang, F. Ricciato, S. Lu, and L. Zhang, “Securing a wireless world,” Proceedings of the IEEE, Vol:94, Issue 2, pp. 442 –454, Feb. 2006.

[13] M. Ilyas and I. Mahgoub, “Handbook of sensor networks: compact wireless and wired sensing systems,” CRC Press LLC, 2005.

[14] E. Shi and A. Perrig, “Designing secure sensor networks,” Wireless Commun. Mag., vol. 11, no. 6, pp. 38–43, Dec. 2004.

[15] Y. Zhou, Y. Fang, and Y. Zhang, “Securing wireless sensor networks: a survey,” IEEE Communications Surveys & Tutorials, Vol:10, Issue 3, pp. 6 –28, Third Quarter 2008.

[16] A. Karnik and K. Passerini, “Wireless network security - a discussion from a business perspective, ” IEEE Wireless Telecommunications Symposium, pp. 261-267, 2005.

[17] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," IEEE Communications Surveys & Tutorials, Volume 8, No. 2, 2nd Quarter 2006.

[18] M. Saxena, "Security in wireless sensor networks - a layer based classification," Cerias Tech Report, April 2007.

[19] P. Matzinger, "Tolerance, danger and the extended family," Annual Review of Immunology, pp. 991-1045, 1994.

[20] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," In: Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, 1994.

[21] P. D'haeseleer, "An immunological approach to change detection: theoretical results," In: Proc. 9th IEEE Computer Security Foundations Workshop. 18–26 274 T. Stibor, J. Timmis, and C. Eckert, 1996.

[22] S. A. Hofmeyr, S. Forrest, and P. D'haeseleer, "An immunological approach to distributed network intrusion detection," In: First International Workshop on the Recent Advances in Intrusion Detection, 1998.

[23] Z. Ji and D. Dasgupta, "Real-valued negative selection algorithm with variable-sized detectors," In: Genetic and Evolutionary Computation – GECCO-2004, Part I. Volume 3102 of LNCS., Seattle, WA, USA, Springer-Verlag, pp. 287-298, 2004.

[24] J. Y. Le Boudec and S. Sara_janovic, "An articial immune system approach to misbehavior detection in mobile Ad-Hoc networks," TechReport IC/2003/59, EPFL-DI-ICA, Lausanne, Switzerland, September 2003.

[25] S. A. Hofmeyr, "An immunological model of distributed detection and it's application to computer security," PhD thesis, Department of Computer Sciences, University of New Mexico, April 1999.

[26] S. A. Hofmeyr and S. Forrest, "Architecture for an articial immune system," Evolutionary Computation 7(1): pp. 45-68, 2000.

[27] M. Just, E. Kranakis, and T. Wan, "Resisting malicious packet dropping in wireless ad hoc networks," Proc. Ad Hoc-Now, Oct. 2003.

[28] J. Kim and P. J. Bentley, "The articial immune system for network intrusion detection: an investigation of clonal selection with negative selection operator," The Congres on Evolutionary Computation (CEC- 2001), Seoul, Korea, pp. 1244-1252, May 2001.

[29] J. Kim and P. J. Bentley, "Towards an artifcial immune system for network intrusion detection: an investigation of dynamic clonal selection," The Congress on Evolutionary Computation (CEC-2002), Honolulu, pp. 1015 - 1020, May 2002.

[30] L. N. De Castro and F. J. Von Zuben, "Articial immune systems: part I basic theory and applications," Technical Report RT DCA, 1999.

[31] L. N. de Castro and J. Timmis, "Artificial immune systems: a new computational intelligence approach," Springer Verlag, Berlin, 2002.

[32] D. Dasgupta and F. Gonz´alez, "An immunity-based technique to characterize intrusions in computer networks," IEEE Trans. Evol. Comput., (6)9, pp. 1081-1088, June 2002.

[33] U. Aickelin and S. Cayzer, "The danger theory and its application to AIS," 1st International Conference on AIS, pp. 141-148, 2002.

[34] S. Gallucci and P. Matzinger, "Danger signals: SOS to the immune system," Current Opinions in Immunology 13, pp. 114-119, 2001.

[35] D. Holzman, "New danger theory of immunology challenges old assumptions," Journal Natl Cancer Inst, 87 (19): pp. 1436-1438, 1995.

[36] J. Kuby, "Immunology," Fifth Edition by Richard A. Goldsby et al., 2002.

[37] P. Matzinger, "Tolerance danger and the extended family," Annual reviews of Immunology 12, pp. 991-1045, 1994.

[38] P. Matzinger, "The danger model: a renewed sense of self," Science 296: pp. 301-305, 2002.

[39] J. J. Zhao and K. E. Nygard, "A Two-Phase Security Algorithm for Hierarchical Sensor Networks," International Conference on Future Computational Technologies and Applications, Sept. 25, 2011.

[40] R. Vance, "Cutting edge commentary: a copernican revolution? doubts about the danger theory," Immunology 165 (4): pp. 1725-1728, 2000.

[41] U. Aickelin and S. Cayzer, "Danger theory and its applications to AIS," In Proc. of the Second Internation Conference on Artificial Immune Systems, pp. 141–148, 2002.

[42] S. Sarafijanovic and J. Le Boudec, "An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal, and memory detectors," In Proc. of the Second Internation Conference on Artificial Immune Systems, pp. 342–356, 2004.

[43] S. Hofmeyr, "An immunological model of distributed detection and its application to computer security," PhD thesis, University Of New Mexico, 1999.

[44] J. Kim and P. J. Bentley, "Towards an artificial immune system for network intrusion detection: an investigation of clonal selection with a negative selection operator," In Proceeding of the Congress on Evolutionary Computation (CEC-2001), Seoul, Korea, pp. 1244–1252, 2001.

[45] E. Eskin, M. Miller, Z. Zhong, G. Yi, W. Lee, and S. Stolfo, "Adaptive model generation for intrusion detection," In Proceedings of the ACMCCS Workshop on Intrusion Detection and Prevention, Athens, Greece, 2000.

[46] P. Matzinger, "Tolerance, danger and the extended family," Annual Reviews in Immunology, 12: pp. 991-1045, 1994.

[47] P. Matzinger, "An innate sense of danger," Seminars in Immunology, 10: pp. 399-415, 1998.

[48] P. Matzinger, "The danger model: a renewed sense of self," Science, 296: pp. 301-304, 2002.

[49] T. R. Mosmann and A. M. Livingstone, "Dendritic cells: the immune information management experts," Nature Immunology, 5(6): pp. 564–566, 2004.

[50] M. B. Lutz and G. Schuler, "Immature, semi-mature and fully mature dendritic cells: which signals induce tolerance or immunity?" Trends in Immunology, 23(9): pp. 991–1045, 2002.

[51] E. P. Box and C. Tiao, "Bayesian inference in statistical analysis," Addison-Wesley, 1973.

[52] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, “Bayesian data analysis,” Chapman & Hall, 1995.

[53] J. M. Bernardo and A. F. M. Smith, “ Bayesian theory," Wiley, 2000.

[54] A. Tarantola, “Inverse problem theory and methods for model parameter estimation,” Siam, Philadelphia, 2005.

[55] J. P. Kaipio and E. Somersalo, “Computational and statistical methods for inverse problems,” Springer, 2004.

[56] G. L. Bowie, W.B. Mills, D. B. Porcella, C. L. Campbell, J. R. Pagenkopf, G. L. Rupp, K. M. Johnson, P. W. H. Chan, S. A. Gherini, and C. E. Chamberlin, “Rates, constants, and kinetic formulations in surface water modeling,” Technical Report EPA/600/3-85/040, U.S. Environmental Agency, ORD, Athens, GA, ERL, 1985.

[57] A. Tarantola, “Popper, bayes and the inverse problem,” Nature Physics, 2: pp. 492–494, August 2006.

[58] K. R. Popper, “Conjectures and refutations – the growth of scientific knowledge,” Routledge, fifth edition, 1989.

[59] D. Gamerman, “Markov Chain Monte Carlo – stochastic simulation for bayesian inference,” Chapman & Hall, 1997.

[60] R. E. Kalman, “A new approach to linear filtering and prediction problems,” Trans. of ASME, Journal of Basic Engineering, 82D(3): pp. 34-45, 1960.

[61] A. H. Jazwinski, “Stochastic processes and filtering theory,” NY: Academic Press, 1970.

[62] M. Chu, H. Haussecker, and F. Zhao, “Scalable information-driven sensor querying and routing for Ad Hoc heterogeneous sensor networks,” International Journal of High-Performance Computing Applications, 16(3): pp. 90-110, 2002.

[63] M. Isard and A. Blake, “Condensation - conditional density propagation for visual tracking,” In International Journal of Computer Vision, 1998.

[64] S. Deb, K. R. Pattipati, and Y. Bar-Shalom, “A new algorithm for the generalized multidimensional assignment problem,” In Proc. IEEE International Conference on Systems, Man and Cybernetics; Emergent Innovations in Information Trans- fer Processing and Decision Making, pp. 249-254, 1992.

[65] M. R. Garey and D. S. Johnson, “Computers and Intractability: a guide to the theory of NP-completeness,” W.H. Freeman and Company, 1979.

[66] D. B. Reid, “An algorithm for tracking multiple targets,” IEEE Trans. on Auto- matic Control, 24(6): pp. 843-854, 1979.

[67] Y. Bar-Shalom and T. E. Fortmann, “Tracking and data association,” Academic Press, San Diego, CA, 1988.

[68] Y. Bar-Shalom and X. R. Li, “Multitarget-multisensor tracking: principles and techniques,” YBS Publishing, Storrs, CT, 1995.

[69] I. J. Cox and S. L. Hingorani, “An effcient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking,” IEEE Trans. on PAMI, 18(2):138{150, February 1996.

[70] Y. Bar-Shalom, K. C. Chang, and H. A. Blom, "Tracking a maneuvering target using input estimation versus the interacting multiple model algorithm," AeroSys, 25(2): pp. 296-300, 1989.

[71] H. Pasula, "Identity uncertainty," Ph.D. thesis. University of California, Berkeley, 2003.

[72] D. Schulz, M. Burgard, D. Fox, and A. B. Cremers, "Tracking multiple moving objects with a mobile robot," In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2001.

[73] R. Neal, "Probabilistic inference using markov chain monte carlo methods," Tech- nical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

[74] http://it.wikipedia.org/wiki/File:Neuron-figure.svg.

[75] D. B. Reid, "An algorithm for tracking multiple targets," IEEE Transaction on Automatic Control, 24(6): pp. 843-854, December 1979.

[76] C. Y. Chong, S. Mori, and K.C. Chang, "Distributed multitarget multisensor tracking," In Y. Bar-Shalom, editor, Multitarget- Multisensor Tracking: Advanced Applications, pages 247–295. Artech House: Norwood, MA, 1990.

[77] D. Li, K. Wong, Yu Hen Hu, and A. Sayeed, "Detection, classification and tracking of targets," IEEE Signal Processing Magazine, 17-29, March 2002.

[78] J. J. Liu, J. Liu, M. Chu, J. E. Reich, and F. Zhao, "Distributed state representation for tracking problems in sensor networks," In Proc. of 3nd workshop on Information Processing in Sensor Networks (IPSN), April 2004.

[79] J. J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, "Distributed group management for track initiation and maintenance in target localization applications," In Proc. of 2nd workshop on Information Processing in Sensor Networks (IPSN), April 2003.

[80] M. Coates, "Distributed particle filters for sensor networks," In Proc. of 3nd workshop on Information Processing in Sensor Networks (IPSN), April 2004.

[81] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," IEEE Computer, Special Issue in Sensor Networks, Aug. 2004.

[82] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," Computer, 34(8): pp. 57-66, Aug. 2001.

[83] L. Schenato, S. Oh, and S. Sastry, "Swarm coordination for pursuit evasion games using sensor networks," In Proc. of the International Conference on Robotics and Automation, Barcelona, Spain, 2005.

[84] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. Sastry," Distributed control applications within sensor networks," Proceeding of the IEEE, 91(8): pp. 1235-1246, Aug. 2003.

[85] K. Whitehouse, F. Jiang, A. Woo, C. Karlof, and D. Culler, "Sensor field localization: a deployment and empirical analysis," Technical Report UCB//CSD-04-1349, Univ. of California, Berkeley, April 9 2004.

[86] L. Doherty, B. A. Warneke, B. Boser, and K. S. J. Pister, "Energy and performance considerations for smart dust," International Journal of Parallel and Distributed Sensor Networks, Dec 2001.

[87] A. B. Poore, "Multidimensional assignment and multitarget tracking," Partitioning Data Sets. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 19: pp. 169–196, 1995.

[88] S. Oh, I. Hwang, K. Roy, and S. Sastry, "A fully automated distributed multiple-target trackingand identity management algorithm," In Proc. of the AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, Aug. 2005.

[89] 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.

[90] 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.

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

[92] 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.

[93] F. Glover, "Future paths for integer programming and links to artificial intelligence," Comput. Open Res. 13: pp. 533-549, 1986.

[94] J. J. Zhao and K. E. Nygard, "A dendritic cell inspired security system in WSNs," FUTURE COMPUTING 2010, International Conference on Future Computational Technologies and Applications, Lisbon, Nov. 21, 2010.

[95] S. Kirkpatrick, Jr., C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science 220: pp. 671-680, 1983.

[96] F. Glover, " Heuristics for integer programming using surrogate constraints," Decision Sci. 8: pp. 156-166, 1977.

[97] F. Glover, "Tabu search—Part I," ORSA J. Comput. 1: pp. 190-206, 1989.

[98] P.S.R. Kasturi, D. Goldgof, and V. Manohar, "Performance Evaluation Protocol for Text and Face Detection and Tracking in Video Analysis and Content Extraction (VACE-II)," technical report, Univ. of South Florida, 2004.