UNIVERSITY OF OSLO
Department of informatics

Security in wireless sensor networks

Master thesis

60 credits

Skjalg Eggen

02. May 2008

ACKNOWLEDGMENTS

First of all, I would like to thank my thesis advisor Prof. Ph.D Chunming Rong for guiding this work and for his helpful input and discussions.

During my time at the University in Oslo department of informatics, I have come in contact with a variety of students and professors which have given me insight in and curiosity for the field of ad hoc communication and networks. The reason I had for taking this education was a keen interest in the field of information security, and because of these people, I came to realize the enormous potential for security research in this rapidly emerging technology. Needless to say, I immediately jumped at the chance of doing my thesis on security in wireless sensor networks.

In this thesis I propose a novel intrusion detection architecture for wireless sensor networks. However I have chosen to think somewhat more practical on the subject. I have chosen to consider small sensor networks which could be used in homes or office buildings, instead of large scale networks with 100's of thousands of nodes.

When you read this thesis, I hope you will be inspired, as I was, by the potential that lies within these types of networks.

-Thank you Vigdis, without you I would never have gotten this far…

ABSTRACT

Wireless sensor network (WSN) is an emerging important research area. The variety in and number of applications is growing in wireless sensor networks. These wireless sensor nodes are tiny devices with limited energy, memory, transmission range, and computational power. Because WSNs in general and in nature are unattended and physically reachable from the outside world, they could be vulnerable to physical attacks in the form of node capture or node destruction. These forms of attacks are hard to protect against and require intelligent prevention methods. It is necessary for WSNs to have security measures in place as to prevent an intruder from inserting compromised nodes in order to decimate or disturb the network performance. Intrusion detection in wireless sensor networks is a much needed security measure. In this thesis we present an intrusion detection framework for wireless sensor networks which does not require prior knowledge of network behavior or a learning period in order to establish this knowledge. We have taken a more practical approach and constructed this framework with small to middle-size networks in mind, like home or office networks. The proposed framework is also dynamic in nature as to cope with new and unknown attack types. This framework is intended to protect the network and ensure reliable and accurate aggregated sensor readings. Theoretical simulation results indicate that compromised nodes can be detected with high accuracy and low false alarm probability when as much as 25% compromised nodes is present in the network. Theoretical simulation results regarding data aggregation indicates that compromised nodes will be limited in their influence on the aggregated data even with as much as 40% compromised nodes in the network. We have only simulated the framework theoretically in a mathematics program and evaluated the theoretical properties of the algorithms. The results are promising and the framework should be simulated in a network simulator for further evaluation.

1. INTRODUCTION

Wireless sensor network (WSN) is an emerging important research area. The variety in and number of applications is growing in wireless sensor networks. They range from general engineering, environment science, health service, military, etc. Wireless sensor network range from sparse networks with 10's of nodes to populated networks with 1000's, possibly 10000's or 100000's of sensor collecting data from the environments.

These wireless sensor nodes are tiny devices with limited energy, memory, transmission range, and computational power. A base station is usually present in the network, which receives the sensor data from the sensors. Such a base station is usually a powerful computer with more computational power, energy and memory. Currently most research in wireless sensor networks have focused on routing protocols, data aggregation and clustering protocols. However, in most circumstances, wireless sensor networks require some amount of security in order to maintain high survivability and integrity of the network. Many emerging and future applications could require strong security in place, in order to function acceptably.

For military applications, WSNs could be placed behind enemy lines in order to detect and track enemy soldiers and vehicles. In indoor environment, sensor networks could be deployed in order to detect intruders and security violations via a wireless home security system. In office buildings, sensor networks could be deployed as a temperature monitoring/regulating or fire alarm system, etc.

Because WSNs in general and in nature are unattended and physically reachable from the outside world, they could be vulnerable to physical attacks in the form of node capture or node destruction. These forms of attacks are hard to protect against and require intelligent prevention methods. It is necessary for WSNs to have security measures in place as to prevent an intruder from inserting compromised nodes in order to decimate or disturb the network performance.

Classical intrusion prevention measures, such as encryption and authentication, can be used in wireless sensor networks to reduce intrusion. However, they cannot eliminate them. Encryption and authentication become useless in the event of a sensor node being compromised, because the nodes carry the private keys the attacker will then be in possession of the cryptographic keys.

Intrusion detection presents a second line of defense and it is a necessity in any high survivability network. Most existent works on intrusion detection systems in wireless sensor networks and mobile adhoc networks use some sort of definition of pattern for detection. i.e anomaly detection[Rajasegarar, et al. 2006] or rule based detection[Chen, et al. 2007]. These types of intrusion detection systems require prior knowledge of some sort in order to establish rules for behavior in the network, or definition of what normal behavior is. There are two drawbacks to these schemes.

1. The need prior knowledge of what to be expected in the behavior in the network
2. They cannot dynamically adapt to new unknown attacks.

In this thesis we present an intrusion detection framework for wireless sensor networks which does not require prior knowledge of network behavior or a learning period in order to establish this knowledge. The proposed system is also dynamic in nature as to cope with new and unknown attack types.

2. WHAT IS A WIRELESS SENSOR NETWORK (WSN)

A wireless sensor network is a network consisting of multiple wireless sensors, also called nodes, which cooperate in sensing some sort of physical or environmental conditions, such as temperature, sound, vibrations, light, movement etc. These networks can consist of everything from 10's of nodes for sparsely populated networks, up to 100's of thousands of nodes in densely populated networks. The individual sensor nodes are small and have limited energy, computational power and memory. This puts some restraints on the applications and protocols which are designed for use in such networks. Wireless sensor networks posses some unique characteristics which is listed below:

- Self-organizing
- Cooperating of sensor nodes
- Short range communication and multihop routing
- Limited energy, computational power and memory
- Dynamically changing topology

Self-organizing

The position of sensor nodes need not be engineered or pre-determined, which allows random deployment in inaccessible terrains or disaster relief operations. In a rescue scenario, thousands of nodes could drop from an airplane over a large area. In order for the network to function, the sensor nodes must have network protocols and algorithms with self-organizing capabilities.

Cooperating of sensor nodes

Because of the limited resources of the nodes, it is necessary for the nodes to cooperate with each other. Several nodes may be tasked with sensing the same phenomenon these nodes may cooperate in a "cluster" where one node is tasked with compressing the sensor result from all the other nodes in the cluster and produce a "collective view" of the cluster on the situation, this is called data aggregation.

Short range communication and multihop routing

Because of the nature of wireless sensor networks, all nodes in the network may not have a direct link to the base station. Hence, they use multihop communication in order to communicate [Draves, et al. 2004]. Multihop communication in wireless sensor networks is expected to consume less power than the traditional single hop communication, which is also desirable in order to keep the communication costs at a minimum.

Limited energy, computational power and memory

Since the nodes are desired to be as small an inexpensive as possible, wireless sensor networks are cursed with limited resources in the form of energy, computational power and memory.

Dynamically changing topology

In wireless sensor networks nodes will fail and drop out of the network, new nodes may be inserted into the network, hence, the topology will be dynamic and ever changing.

2.1 APPLICATIONS OF WIRELESS SENSOR NETWORKS

Figure 2.1 shows the complexity of wireless sensor networks, which generally consist of a data acquisition network and a data distribution network, monitored and controlled by a management center. The multitude of available technologies makes even the selection of components difficult, let alone the design of a consistent, reliable, robust overall system [Hu and Evans. 2003].

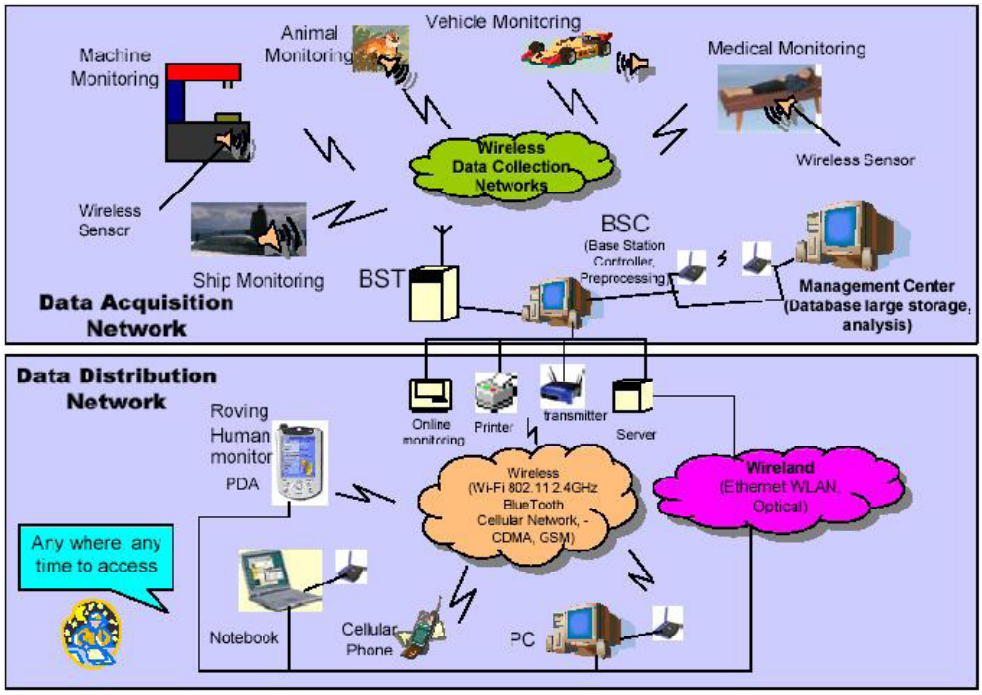


Figure 2.1 From [Hu and Evans. 2003]

The applications for wireless sensor networks are many. Wireless sensor networks could replace current wired counterparts and provide diverse operations where traditional networks would be impossible, due to the environment, scale of the operation or expenses in needed wiring. Some examples of areas of which WSNs could be used are listed below.

Home automation

Sensor nodes could be located in every room to measure the temperature, detect movement and report to an alarm system, detect smoke in case of a fire and report to a fire alarm system etc.

General engineering

It can be used for automotive driving, fingertip accelerometer virtual keyboards, sensing and maintenance in industrial plants, aircraft drag reduction, smart office space management, tracking of goods in retail stores, tracking of containers and boxes in shipping companies, social studies on human behavior, commercial and residential security [Ngai. 2005].

Agriculture monitoring

It can used in crop and livestock management and precision control with nodes placed on animals tracking their movement or temperature, managing the feeding of individual animals, automating the distribution of medicine for individual animals, etc.

Civil engineering

Nodes could be built into the walls or the concrete of buildings and detect changes in the structural integrity of buildings which can develop over the years or after earthquakes, extreme weather, fires etc.

Military applications

Large sensor networks could be deployed behind enemy lines in order to conduct surveillance or track enemy troops or vehicles. Wireless sensor networks could be used for asset monitoring and management, with sensor nodes embedded in the uniforms and weapons of the soldiers. Such networks could also be used for battlefield monitoring and asset coordination accordingly.

Health care

Small sensor nodes could be used for medical applications like surveillance of elderly people. Devices that are carried around or which are worn with the clothes like it is proposed in the field of *Wearable Computing* [Chen, et al. 1996] could monitor vital function and report them to the family doctor or directly to the ambulance in case of an emergency like a heart attack. In the future small sensor nodes could be implanted into the body in order to detect internal diseases like cancer at an early stage. This way it can be treated earlier, when the recovery probabilities are much higher [Ngai. 2005].

2.2 Requirements for applications in wireless sensor networks

Due to the characteristics and limitations of WSN, there are some requirements for building applications for use in such networks. These requirements are listed below:

- Power restrictions
- Limited Computational power
- Storage Restrictions

Power restriction

Because of the nature of wireless sensor networks, continuous power supply through wired connections is not an option. Sensor nodes are usually powered by small batteries or in some cases they could draw power from the environment, possibly by the use of small solar panels, motion generated power, etc. This limits the amount of power resources at hand. Wireless sensor networks could be placed in inhospitable environments, which make the replacements of batteries impossible. These limited power resources are used for sensor operations, sensor data processing and communication. The most power hungry operations of the nodes are communications between the nodes in the network, and this should be taken into consideration in the design of new applications.

Limited Computational power

The amount of available computational power is linked with the amount of available power for the entire node. Even though computations require less power than communications, there are still limitations to the amount computations the node can do. This is a critical limitation which requires consideration in the design of applications. Because of this limitation, robust security protocols with heavy encryption are not possible. Heavy asymmetric encryption require overhead in both communication and computation.

Storage restrictions

The amount of available memory is restricted by the small size of the nodes. This memory is shared for all operations of the node, and storing large numbers of encryption keys and routing information could easily fill up the available memory in large networks. This limitation needs to be considered in the development of applications

2.3 SECURITY IN WIRELESS SENSOR NETWORKS

Security in wireless sensor networks will and should become a hot topic in future WSN research. Security protocols should be developed and deployed in the same time as the routing and aggregation protocols, instead of adding them later on as patches to existing holes in order to create a robust platform to deploy security critical applications.

Sensors are often deployed in accessible areas, which add the risk of physical attack. This is why wireless sensor networks pose unique challenges. Security protocols and techniques used in traditional wired networks cannot be applied directly. Due to the scarce resources of wireless sensor networks, sensor nodes are limited in their energy, computation, and communication capabilities. New light weight, multi layered security schemes need to be developed in order to detect and prevent the new forms of attacks these networks are subject to, while using as little as possible of the scarce resources that exist in the networks.

2.3.1 THREAT MODELS

Attacks come in different sizes and shapes and can be conducted from the inside and the outside of the network.

External attackers are attackers that are not legally part of the network. They could be part of another network which is linked to the target network using the same infrastructure or same communication technology. These nodes can carry out attacks without being authorized on the target network. These attackers could also be an outside node, not part of the network, but with jamming or eavesdropping capabilities.

Internal attackers are compromised nodes which are authorized on the target network. These nodes are capable of more sophisticated attacks because they are seen as authorized by the network and fellow nodes. As an example they can produce false routing information to the network, and in this manner decimate the network or simply route attractive traffic through itself in order to collect data or maybe choose not to forward the packets consequently disrupting the connection. They could also be placed in strategic positions to report false sensor readings in order to “pollute” picture the sensor data presents.

The attacker can be either *active* or *passive*

Passive attackers do not disrupt service. Eavesdropping is a good example of a passive attack, in this attack the attacker only listens to traffic that it can intercept. The attacker does not do anything active in the sense of attracting traffic to itself through the network.

Active attackers alter data, obscure the operation or cut off nodes from neighbors. An active attacker must be able to inject packets to the network. A good example of an active attack is injecting false routing information to the network in order to decimate the network or route interesting traffic through itself. Active attackers can target the physical layer by jamming the transmissions of wireless signals or by destroying the hardware at certain nodes. Attackers can also target the network layer protocols such as routing by injecting false routing information, and they can target the application layer by injecting false information onto the network. In essence wireless sensor networks need protection on multiple layers in order to make attacks more difficult to carry out.

2.3.2 ATTACK TYPES

[Sarma et al. 2006] Gives a good, layered based, overview of the various types of attacks against wireless sensor networks:

Physical layer:

Jamming

Jamming is a popular Denial of Service (DoS) attack. In this attack the attacker attempts to jam the frequencies of the radio used for communication between the nodes in the network. In this attack, an adversary may use e few nodes in strategic positions to effectively jam most of the communications inside the network. In essence, an attacker needs only a few nodes in order to disseminate a large network.

Tampering

Because of the nature of wireless sensor networks, an adversary could easily get physical access to the sensor nodes. This may enable an attacker to compromise sensor nodes in order to get access to the cryptographic keying material, replicate the node and place several of his own nodes in the network or simply just destroy the sensor nodes in a DoS like manner.

Sybil attack

The Sybil attack is a particularly nasty attack. The Sybil attack is an attack that can span several layers in the protocol stack. The essence of the Sybil attack is that one single compromised node can impersonate several nodes. A node is inserted into the network and assumes identities of nodes from different parts of the network. The base of this attack is at the physical layer [Newsome, et al. 2004].

Data link layer:

Collision

This is a DoS attack, where a node induces a collision in some small part of a transmitted packet. The packet will then fail the checksum check, because of the changes brought on by the collision, and the receiver node will then ask for a retransmission of the packet

Exhaustion

This attack is a collision attack taken a bit further. A malicious node may conduct a collision attack repeatedly in order to exhaust the power supply of the communicating nodes.

Interrogation attack

When the physical layer uses Request To Send (RTS) / Clear To Send (CTS) messages in order to conduct medium access control, a malicious node can repeatedly send RTS messages to a target node and ignore the CTS messages. By doing so, the malicious node can flood the network link of the target node.

Sybil attack

The Sybil attack becomes more prominent in the data link layer. Two types of the Sybil attack on the data link layer are:

Data aggregation

In a data aggregation scheme, the sensor data is aggregated throughout parts of the network in order to present a collective view of the monitored phenomenon. If an attacker controls a few nodes, he can have some effect on the aggregation result. If those nodes are Sybil nodes with many identities, an attacker would have a more prominent effect on the aggregation result.

Voting

In voting schemes, Sybil nodes can be used to “rig the election”. The impact on the voting result depends on the amount of identities the Sybil nodes are in possession of.

Network layer:

Neglect and greed

In this attack, a malicious node that is part of a network route can drop the packets but still send ACK to the sender. An attacker can drop just certain packets, or simply just drop all packets coming down the routing path.

Missdirection

In this attack a malicious node, that is part of a route, can instead of dropping packets, quite simply send them on a different path where there does not exist a route to the destination. The malicious node can do this for certain packets, or all packets.

Internet smurf attack

This attack is a DoS attack, where a malicious node can spoof the address of a victim node and broadcast echoes and route the replies to the victim node. This way the malicious node can flood the victim's network link.

Black hole attack

This is a DoS attack, where a malicious node advertises a zero cost route through itself. If the routing protocol in the network is a "low cost route first" protocol, like distance vector, other nodes will chose this node as an intermediate node in routing paths. The neighbors of this node will also chose this node in routes, and compete for the bandwidth. This way the malicious node creates a black hole inside the network.

Sybil attack

In multipath routing protocols, a Sybil node can fool the protocol into thinking that multiple paths exist, when in reality all paths goes through the same Sybil node.

Spoofing and altering routing information

In this attack, a malicious node may be able to create routing loops, wormholes, black holes, partition the network, etc, by spoofing, altering or replaying routing information.

Worm hole attack

Just like the theoretical wormholes in space, this attack can send packets, routing information, ACK etc, through a link outside the network to another node somewhere else in the same network. This way an attacker can fool nodes into thinking they are neighbors, when they are actually in different parts of the network. This can also confuse routing mechanisms that rely on knowing distances between nodes. A wormhole attack can be used as a base for eaves dropping, not forwarding packets in a DoS like manner, alter information in packets before forwarding them etc.

Selective Forwarding attack

In this attack, malicious nodes can decide not to forward packets of certain types or to or from certain nodes.

Transport layer:

Flooding attack

In this attack, a malicious node may send continuous connection requests to a victim node effectively flooding the victim's network link.

2.3.3 SECURITY REQUIREMENTS

As we have discussed previously, wireless sensor networks are subject to passive eavesdropping as well as active interfering over several layers. This can lead to leaking of sensitive information which is observed by a passive eavesdropper, message contamination or node impersonation by an active attacker.

In this section we will examine the security requirements which are essential for a secure, high survivability wireless sensor network.

Confidentiality

Unlike wired networks, an attacker does not need to gain physical access to cables/switches for wiretapping or compromise routers and other nodes in order to conduct eavesdropping. Due to the fact that all signals are transmitted over the air, an attacker can eavesdrop on any node it chooses, as long as it is within radio coverage. In order to prevent information

traveling through the network from being compromised, confidentiality is a necessary requirement. Compromised nodes present a serious threat to confidentiality, because by compromising a node, an attacker can gain access to the cryptographic keys used to protect the communication.

Authentication

Authentication is necessary to distinguish legitimate sensor nodes from intruders. Authentication is also necessary in order to determine that the received data came from a authorized sensor node, and not injected by an attacker trying to falsify information.

Authentication is also crucial in cluster formations, where sensor nodes form clusters according to location, sensing data etc. Authentication is needed in order to ensure that nodes inside the cluster only accepts data from authorized nodes in the same cluster.

Integrity

Integrity is an important criterion, as information moving through the network could be altered. Without integrity we would not be able to trust the information received from the sensor network.

Freshness

Freshness is important, because attackers could replay packets to confuse the network. Freshness requirement ensures that only fresh unused data are accepted.

Secure management

Secure management is needed from the base station in order to securely manage distribution of cryptographic keying material, securely form clusters and adding or removing member of clusters.

Availability

Availability is a crucial requirement because it ensures that the services of the network are available at the time it is needed. DoS attacks could disrupt the availability of wireless sensor networks, so it is necessary to ensure that such attacks have as little impact as possible.

3. INTRUSION DETECTION

As we have discussed previously, wireless sensor networks are subject to passive eavesdropping as well as active interfering. This can lead to leaking of sensitive information which is observed by a passive eavesdropper, message contamination or node impersonation by an active attacker.

These attacks are similar to attacks on traditional wired networks. However, due to the characteristics of the wireless MAC layer, wireless sensor networks are subject to new forms of attack, and traditional protection mechanisms for wired networks are not sufficient to secure these networks. Unlike wired networks, an attacker does not need to gain physical access to cables/switches for wiretapping or compromise routers and other nodes in order to conduct eavesdropping. Due to all signals being transmitted over the air, an attacker can eavesdrop on any node it chooses, as long as it is within radio coverage, he can also easily inject packets and replay captured packets.

Another major threat is the compromising of nodes. Since the nodes in wireless sensor networks are unattended and could be accessible to the public, there is a risk of nodes being captured, compromised or hijacked. Because of this threat, every node in the network should be skeptical to all other nodes in the network. Integrity validations, which rely on redundant information from different nodes, such as secure routing information, also rely on the trustworthiness of other nodes.

Traditional defenses like encryption and authentication fail to protect the network when a node, which is in possession of the private cryptographic keys, can be compromised. Adhoc routing protocols also rely on the cooperation of other nodes in the network to establish routes and routing tables. If nodes are hijacked or compromised, attackers can decimate the entire network by submitting false routing information. On the other hand, they can make sure that all traffic passes through their node or nodes in order to eaves drop, conduct selective forwarding attacks, etc.

If we have the ability to detect an intrusion or attack once it comes into the network, it can be stopped before it can cause any real damage to the system. This is what intrusion detection systems (IDS) have to offer.

3.1 INTRUSION DETECTION FOR TRADITIONAL NETWORKS

All computer or network activity can be classified into three different classes:

1. Normal activity
2. Abnormal but not malicious
3. Malicious activity

The fundamental problem is how to classify different activities into one of these classes. Two main intrusion detection techniques are host-based and network-based intrusion detection.

Host based intrusion detection is located at a host in the network, and monitors processes executing on the host for misuse.

Network-based intrusion detection is located inside the network and monitors the activities inside the current network in order to detect misuse.

Hybrid solutions exist which employs both techniques. A hybrid solution on a host may observe network traffic to and from the host, as well as processes executing on the same host, in order to detect misuse.

The two most used traditional intrusion detection techniques used are the misuse detection and anomaly detection techniques.

Misuse detection employs pattern recognition in its efforts to detect intrusion. This technique relies on the prior knowledge of misuse patterns which it can compare current activities to in order to detect illegal activities. This technique has the drawback of not being able to detect new and unknown attack types.

Anomaly detection requires a learning period where it gathers information about the normal activities of the host or network. It then compares the current activities to the normal behavior it observed in the learning period in order to detect anomalies in the form of deviation from the norm. The drawback of this technique is that it may treat all anomalies as intrusions, hence, false detections are expected.

An intrusion detection model consists of six main components [Denning. 1987]:

- Subjects: Initiator of activity on a target system-normally users.
- Objects: Resources managed by the system-files, commands, devices, etc.
- Audit records: Generated by target systems in responses to actions performed or attempted by subjects on objects-user login, command execution, file access, etc.
- Profiles: Structures that characterize the behavior of subjects with respect to objects in terms of statistical metrics and models of observed activity. Profiles are automatically generated and initialized from templates.
- Anomaly records: Generated when abnormal behavior is detected.
- Activity rules: Actions taken when some condition is satisfied, which update profiles, detect abnormal behavior, relate anomalies to suspected intrusions, and produce reports.

The observations of such systems are in the form of statistical data with variables in the form of cumulative observations of one or more activities over a period of time. This data can be represented by a statistical model like a uni or multivariate distribution with mean and standard deviation, Markov process model, time series model, etc.

3.2 INTRUSION DETECTION FOR WIRELESS SENSOR NETWORKS

Intrusion detection systems for wireless sensor networks can be divided into three categories [Brutch and Ko. 2003].

1. Stand-alone
2. Distributed and Cooperative
3. Hierarchical

Stand-alone

In this architecture, an IDS is run on all nodes. All nodes run their IDS independently and without cooperation with each other. Decisions are based solely on the observations of the information collected at its own node. Nodes in the same network do not know anything about the situation of other nodes in the network. This could be suitable in networks where not all nodes are capable of running an IDS or does not have one installed. This architecture is not effective due to the fact that no alert information is passed between the nodes when an

intrusion is detected. Also there is no cooperation between the nodes, so if a node does not have conclusive evidence of an intrusion, it cannot cooperate with its neighbors in order to get more information about the suspected intrusion.

Distributed and cooperative

This architecture requires all nodes to have an IDS agent running on them. Every node participates in the intrusion detection and response. Every IDS agent is responsible for collecting data and detecting intrusions locally, but they can cooperate with neighbor nodes, when the evidence of an intrusion is inconclusive, in order to collect more information to verify or dismiss the attack. This architecture is suitable for flat network infrastructure.

Hierarchical

This architecture extends the distributed and cooperative IDS architecture and has been proposed for multi-layered network infrastructures where the network is divided into clusters. Cluster heads of each cluster generally have more functionality than other members in the cluster and, in a way act as the traditional control points in wired networks. Hierarchical IDS architecture makes use of this infrastructure model and assigns more of the responsibilities to the IDS agents running on the cluster heads. All nodes run an IDS agent and detect intrusions locally. The IDS agent running on the cluster head is responsible for detecting intrusions locally as well as globally for the cluster. Thus the cluster head IDS agent monitors network packet and initiates a global response when it detects an intrusion.

3.2.1 INTRUSION DETECTION TECHNIQUES

The intrusion detection systems for wireless sensor networks can base their detection techniques on the same approaches as the traditional systems, namely *anomaly* detection and *misuse* detection, or they could also use *specification-based* detection techniques.

In ***misuse*** detection technique the system compares the actions in the network with known attack patterns. As described in the traditional IDS systems, these systems have the drawback of not being able to detect new and unknown attacks. They also require memory space to store the attack signatures. Hence, they can be difficult to implement in resource constraint sensor networks.

Anomaly detection techniques compare the activities of the network with what is expected as normal behavior and detect intrusions as activities that differ from the normal expected behavior by a statistical significant amount. These systems usually establish knowledge of what is "normal" behavior of the network during a learning period where they gather information which is considered as normal behavior. These systems also have some drawbacks. First, they are vulnerable to intruders in the learning period. Second, due to the unreliable communication of wireless sensor networks, the "normal" behavior of past observations may not be concurrent with the present behavior. Thus, false detections should be expected.

Specification-based detection techniques monitor network behavior and compare, in the same manner as anomaly detection techniques, current behavior with what is expected to be "normal" behavior. The IDS flags nodes that differ from the norm by a significant statistical amount as intruders. The difference from the anomaly detection scheme is how the "normal" behavior knowledge is established. In the specification-based detection technique, the normal behavior is a manually specification of what is normal behavior. Like the other techniques, this one also has a drawback. Because the "normal" behavior is manually specified, it is unable to detect new or unknown attacks. Since it is an anomaly detection technique, it also suffers from the weakness of possibly producing false detections.

3.2.2 EXAMPLE IDS ARCHITECTURES

Local Intrusion Detection System (LIDS)

The Local Intrusion Detection System (LIDS) is an example of a distributed and cooperative intrusion detection system [Alberts and Kuhn. 2002]. All nodes run a Local Intrusion Detection System (LIDS) for local detection, however, this can be extended globally by cooperating with LIDS on other nodes in the network. The LIDS exchange two types of data:

- Security data to obtain complementary information from collaborating nodes
- Intrusion alerts to inform others of locally detected intrusions

The LIDS analyze data obtained locally and data obtained from other nodes in order to detect intrusions. In LIDS SNMP data is proposed as an audit data source. This eliminates this

problem and reduces power usage if an SNMP agent is already running on each node. Mobile agents are used to transport SNMP requests to other nodes, in order to distribute the detection tasks. The way this is done differs from traditional SNMP in that this approach brings the code to the data and not the data to the code. Thus, reducing the overhead of the network by telling the mobile agents, on the node in question, to audit the data on that node instead of having to transport all the data to the querying node. The result from the mobile agents is then transported back to their LIDS or to another node for further investigation. The LIDS architecture is shown in Figure 3.1

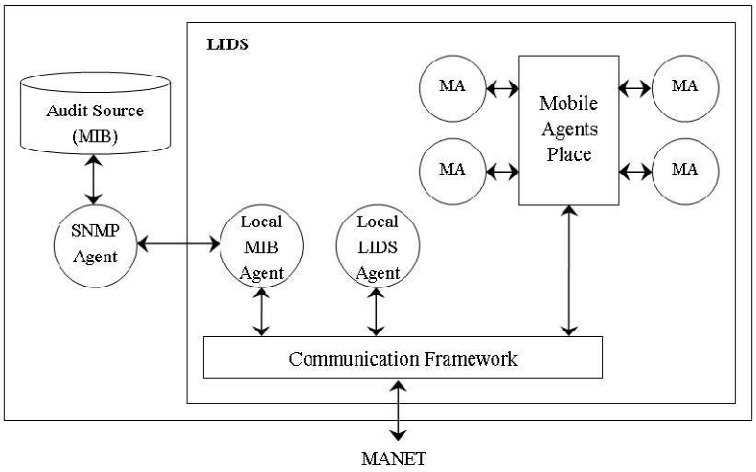


Figure 3.1: The LIDS Architecture From [Anatvalee and Wu. 2007]

LIDS can use misuse, anomaly and specification-based detection techniques.

Dynamic Hierarchical IDS architecture

This IDS is based on cooperation of IDS agents on nodes in the network [Anatvalee and Wu. 2007]. The network infrastructure consists of clusters and all nodes must be part of a cluster. Cluster heads can again form clusters with each other and so on to form a multi layered infrastructure.

The cluster nodes are responsible for monitoring, logging, analyzing, responding to intrusions detected and reporting to the cluster heads. The cluster heads are responsible for correlating reports from nodes in its cluster and, if needed, request additional information from nodes in order to correlate reports correctly. They are also required to analyze collected data before passing it to upper levels. The uppermost levels of the hierarchy have the authority and responsibility for managing the detection and response capabilities of the clusters and cluster

heads below them. They can send signature updates or directives and policies to alter the configurations for intrusion detection and response. Because of the responsibilities and authority of the upper layer nodes, there must be some criteria on selecting these nodes. The most important one being resistance to compromise of the node, which is to say the probability of the node being compromised. Figure 3.2 shows the architecture of this IDS.

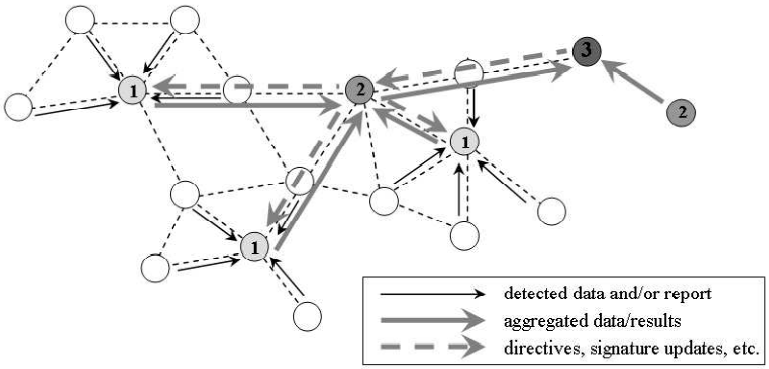


Figure 3.2: Dynamic intrusion detection hierarchy From [Anatvalee and Wu. 2007]

3.2.3 DECISION MAKING

An important part in cooperative intrusion detection is who makes the final decision about whether or not a node is malicious. This could be done in two ways. Either the nodes decide independently, or the nodes cooperate in a majority voting scheme.

In an independent decision scheme a few nodes are tasked with collecting evidence of intrusion from other nodes and make the decision. The other nodes do not participate in this decision. In a cluster topology network, the cluster head would typically be the deciding node for the nodes in its cluster. This scheme could be vulnerable if an attacker compromises the deciding node. Then he would be able to control which nodes are decided to be intruders or not.

In a cooperative decision scheme nodes cooperate in deciding the guiltiness of a suspected node. This could be done by a majority vote where nodes share their suspicion of other nodes and if the majority > 50% of the nodes think a node is malicious, then that node is decided to be an intruder. This scheme is less vulnerable in that no single node is in charge of the

decision making. Hence, he would have to control 50% of the nodes or more, in order to be in control of the decisions.

4. DATA AGGREGATION

4.1 WHAT IS DATA AGGREGATION

Data aggregation is a process in which information is gathered and expressed in a summary form. Data aggregation is mostly used when we want to obtain information about particular groups based on some kind of variable. As an example we might be interested in knowing how our income compares to other people in the same job situation and in the same location as our self. In this case the information is income and the variables are location and job situation. If we have a database with information about individual income and personal information, like work place and living location, we could aggregate the data and produce a mean value, or perhaps a max value, of the income of people in the same situation as our self.

4.2 DATA AGGREGATION IN WIRELESS SENSOR NETWORKS

Data aggregation is a fundamental part of energy saving in wireless sensor networks. As a result of energy resources being scarce and communication between nodes being the most power hungry operations, reducing the amount of communication is essential to prolong the network lifetime. Data aggregation helps to reduce the communication in cluster based network topologies by allowing only the cluster heads or appointed aggregator nodes to forward data to the receiver (i.e the base station or a sink). The data aggregator node receives sensor data results from sensor nodes and does some computations on the data to produce a collective view of the observed physical phenomenon. This computation could be simply finding a mean value of the observations in that nodes area. This aggregation result is then forwarded towards the destination as a single observation instead of having every single sensor node convey their result to the destination, and by doing so using considerably more energy resources.

Because of the energy saving potential in using in network data aggregation in wireless sensor networks, this has been a hot research topic and several aggregation schemes have been proposed [Considine et al. 2004] [Manjhi, et al. 2005] [Cormode, et al. 2005] [Madden, et al. 2002] [Shrivastava, et al. 2004] [Skraba, et al. 2006] are just a few examples.

4.3 SECURITY IN DATA AGGREGATION

As we have previously discussed, in-network data aggregation in wireless sensor networks is an essential part in energy efficiency and helping prolong network lifetime. However, this is a potential security risk as it introduces a single point of failure. If the sensor node in charge of data aggregation is compromised, an attacker would be in complete control of sensor reading for all nodes which have that node as its aggregation point.

By having a compromised aggregation node in the network, an attacker could have serious influence over the final aggregation result, which is received by the destination, depending on the size, topology and which node he has compromised. In the worst case scenario of a hierarchical aggregation topology, an attacker compromises the aggregation node closest to the destination and by doing so, he could induce a DoS attack by refusing to forward any data, or he could simply manipulate the aggregation result to his own choosing. In a flat aggregation topology with data aggregation only being performed at the cluster head, an attacker who compromises the cluster head would be in complete control of the sensor readings of that cluster.

An attacker does not necessarily have to compromise the aggregation node to have an impact on aggregation result. Compromised sensor nodes could also have an impact on the aggregation result by submitting false readings to the aggregating node. If an attacker controls several nodes, he could have a severe impact on the aggregation result, depending on the size of the network. It is important to implement some sort of robust aggregation in order to minimize the impact of compromised nodes.

It is important to protect the network from these types of vulnerabilities by designing security mechanisms to counter the threat of aggregation node compromise and compromised nodes interfering with aggregation results. As discussed previously in the intrusion detection chapter, intrusion detection systems could be used to detect compromised sensor nodes. Hence, intrusion detection schemes could be designed to defeat threats against the aggregation

process also. In the next section we present our proposal for an intrusion detection framework to defeat threats against data aggregation in a flat aggregation topology as well as protect the network against several of the other threats posed by compromised nodes.

5. PROPOSED INTRUSION DETECTION ARCHITECTURE

In this section we present our proposal for an algorithm that does anomaly based intrusion detection, and at the same time handles threats against data aggregation in a flat aggregation topology.

5.1 BACKGROUND MATHEMATICS

Before we go into the details of the algorithm we need to introduce some of the mathematics we use in the computations. In this section we shall become acquainted with *Mahalanobis distances* theory, the *chi-squared distribution* and the *Orthogonalized Gnanadesikan-Kettenring* (OGK) robust estimators) [Maronna, et al 2006]. These mathematical operations and principals are at the heart of our algorithm and therefore it is important to understand the properties of them.

5.1.1 MAHALANOBIS DISTANCES

Our algorithm depends on multivariate data to calculate the normal behavior of the network. These multivariate data are represented by a collection of attribute vectors for each node in the neighborhood being watched. The attribute vectors represent the behavior of the nodes, with each attribute representing a part of that behavior. The attributes could be observed packet loss rate, forward delay time, packet sending rate, sensor readings, etc. All these attribute vectors are collected in a data set in the form of a matrix (Table 5.1), where the columns represent the individual attributes and the rows representing attribute vectors for each node. This data set is a representation of a multivariate data set.

	Packet sending rate	Forward delay time	Sensor readings
Node 1			
Node 2			
Node 3			
Node 4			
Node 5			
Node 6			
Node 7			
Node 8			
Node 9			

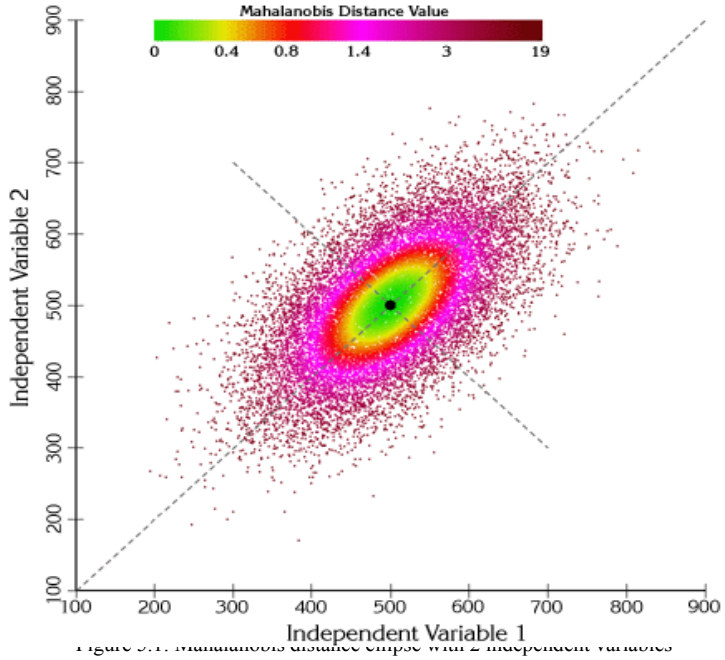
Tabel 5.1: Visual represenstation of data set vith attribute vectors

The shape and size of this multivariate data set is quantified by the covariance matrix [Filzmoser]. The Mahalanobis distance is a well known distance measure that takes into account the covariance matrix, and the Mahalanobis squared distance (MD^2) is an approach to multivariate outlier detection based on the means of the variables (attributes) and the covariance matrix for a sample data set with one or more variables (attributes) [Maronna, et al 2006].

The MD^2 is a measure of how far a random vector, of variables or attributes, is from the middle (mean) of its distribution. The MD^2 provides a reasonable summary distance of each item from the mean. For a q-dimensional multivariate sample $x_i = \{y_1, y_2, \dots, y_q\}$ $(1 \leq i \leq n)$ the MD^2 is defined as

$$MD_i^2 = \left((x_i - \mu)^T \Sigma^{-1} (x_i - \mu)\right) \; for \; 1 \leq i \leq n \qquad (1)$$

Where $\mu = \{\mu_1, \mu_2, \dots, \mu_q\}$, is the estimated multivariate location or means and Σ is the estimated covariance matrix.



(http://www.jennessent.com/arcview/mahalanobis_description.htm)

Figure 5.1 depicts a visual representation of a Mahalanobis distances distribution for a two-variable sample. We can observe how the samples are scattered in a cluster with the mean as the single two-dimensional point in the middle. In a three-variable distribution the samples will be scattered in a three-dimensional sphere, with the mean as the three-dimensional point in the middle of the distribution. What is important to notice is that the Mahalanobis distances distribution is not necessarily completely circular in shape, it may be more ellipsoid. This is influenced by the distribution of the individual variables and the covariance between them.

5.1.2 CHI-SQUARED DISTRIBUTION

The chi-squared distribution is a much used probability distribution in inferential statistics, where one wants to draw a conclusion about something on the basis of what one already knows. The chi-squared distribution has one variable: q – a positive integer that specifies the degree of freedom (number of variables).

If $f(x_i)$ is distributed as $N_q(\mu, \Sigma)$, i.e., q-dimensional vector $f(x_i)$ follows a multivariate normal distribution with mean vector μ and variance-covariance matrix Σ, the Mahalanobis squared distance = $\left((x_i - \mu)^T \Sigma^{-1} (x_i - \mu)\right)$ is distributed as χ_q^2, where the q is the number of

variables in the multivariate data, in our case is the number of variables (attributes) in the observation data set [Liu et al. 2007] [Maronna, et al 2006].

We use the chi-squared cumulative distribution function (CDF) in our algorithm depicted in figure 5.2 below. This function can be used to express the probability (α) of a multivariate sample (attribute vector) being less than or equal to a value $X_q^2(\alpha)$, where $X_q^2(\alpha)$ is the value of the chi-squared distribution where it is expected that α percent of the distribution is situated. We use this value $X_q^2(\alpha)$ in the calculation of the threshold value Θ for the MD^2, over which a node is suspected to be an outlier.

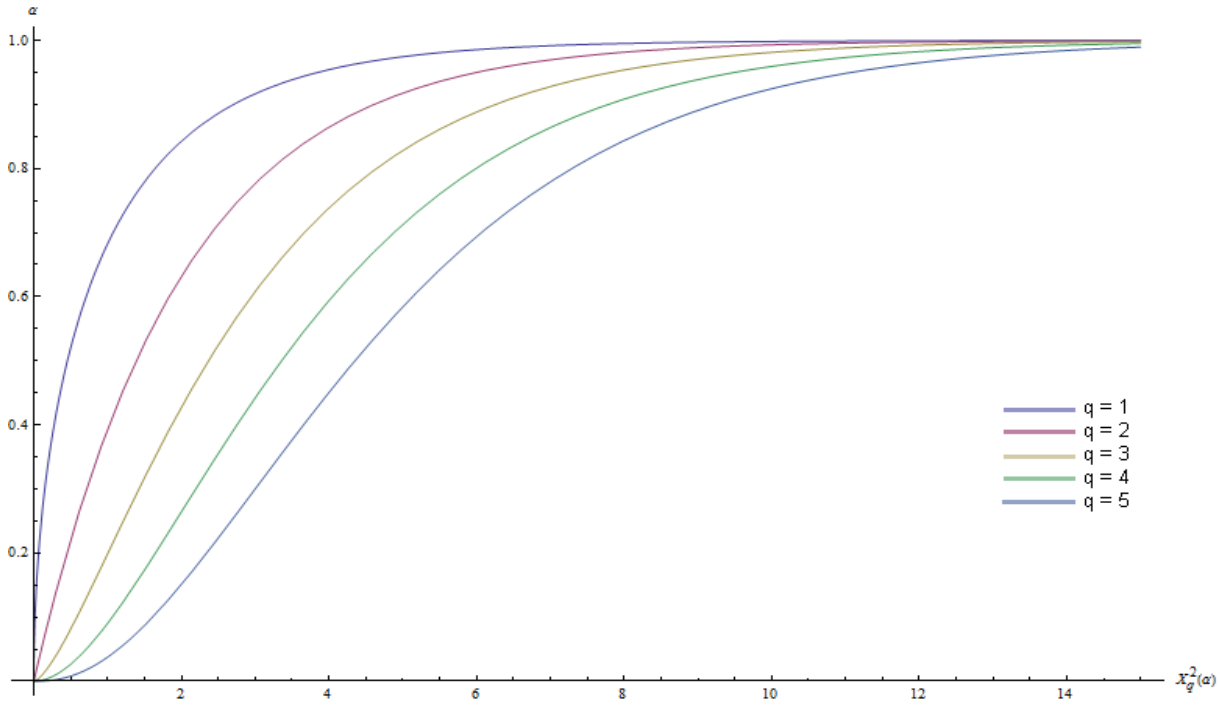


Figure 5.2: Chi-squared cumulative distribution function (CDF)

Using this approach, multivariate outliers can be defined as x_i's with significantly large Mahalanobis squared distances. However, this approach has some shortcomings. If the Mahalanobis distances are not estimated by some robust procedure, single outlying x_i's or groups of outlying x_i's can have a severe influence on the distance measure because the multivariate location μ and the covariance matrix Σ are usually estimated in a non-robust manner[Filzmoser]. As a result, real outliers may not be found. Hence, robust estimates for the multivariate location μ and the covariance matrix Σ, are needed.

5.1.3 ORTHOGONALIZED GNANADESIKAN-KETTENRING (OGK)ROBUST ESTIMATES

"The insider attacker detection algorithm" by [Liu et al. 2007] propose the use of the orthogonalized Gnanadesikan–Kettenring estimate (OGK) [Maronna, et al 2006] in calculating the robust estimates of the sample means μ and the robust estimate of the samples covariance matrix Σ. Acording to [Maronna, et al 2006] and [Liu et al. 2007] the computation of the robust estimates of the sample means μ and covariance matrix Σ, is done in the following manner:

First we have to define how to calculate the single-variate mean $\hat{\mu}$ and variance $\hat{\sigma}^2$.

Let $Y = \{y_1, y_2, \ldots, y_n\}$ be a single-variate sample set coming from a distribution with mean μ and variance σ^2. Let μ_0 be the median and and σ_0 be the MAD of Y., where MAD(Y) = median(|Y − median(Y)|).

Define a weight function

$$W(x) = \left(1 - \left(\frac{x}{c_1}\right)^2\right)^2 I(|x| \leq c_1) \quad (1)$$

and a ρ-function

$$\rho(x) = min(x^2, c_2^2), \quad (2)$$

where c_1 = 4.5 and c_2 = 3. Then $\hat{\mu}$, $\hat{\sigma}^2$ can be estimated by:

$$\hat{\mu} = \frac{\sum_{i=1}^{n} y_i W(v_i)}{\sum_{i=1}^{n} W(v_i)} \; for \; v_i = \frac{y_i - \mu_0}{\sigma_0} \quad (3)$$

$$\hat{\sigma}^2 = \frac{\sigma_0}{n} \sum_{i=1}^{n} \rho\left(\frac{y_i - \hat{\mu}}{\sigma_0}\right), \quad (4)$$

Now we describe the OGK estimates $\hat{\mu}$ and $\hat{\Sigma}$ based on the multivariate data set $F(x) = \{f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T \,|x_i \in N(x)\}$. Let $\hat{\mu}\,(\cdot)$ and $\hat{\sigma}^2(\cdot)$ denote the univariate statistics, as described in Eq. (3) and (4). The OGK estimates can be computed in the following manner:

1) Compute $G(x) = \{g(x_i)|x_i \in N(x)\}$ from $F(x)$, where $g(x_i) = P^{-1} f(x_i)$ for $P = diag(\hat{\sigma}(\widetilde{F_1}(x)), \hat{\sigma}(\widetilde{F_2}(x)), \ldots, \hat{\sigma}(\widetilde{F_2}(x)))$. Here $F_j(x)$ is the j-th component set of $F(x)$, with $F_j(x) = \{f_j(x_i)|x_i \in \widetilde{N}(x)\}, 1 \leq j \leq q$.

2) Calculate a q × q matrix R, with $R_{j,k}$, the element at the jth-row and k-th column defined as $R_{j,k} = \begin{cases} [\hat{\sigma}^2(G_j + G_k) - \hat{\sigma}^2(G_j - G_k)] & if\ j \neq k \\ 1 & if\ j = k \end{cases}$

3) Apply the spectral decomposition to obtain R = QΛQT , where Q is the q×q matrix whose columns are the eigenvectors of R, and Λ is the diagonal matrix composed of R's eigenvalues.

4) Compute $H(x) = \{h(x_i)|x_i \in N(x)\}$ from $G(x)$, where $h(x_i) = Q^T g(x_i)$. Then calculate $\Delta = (\hat{\mu}(H_1(x)), \hat{\mu}(H_2(x)), \ldots, \hat{\mu}(H_q(x)))^T$, and $\Gamma = diag(\hat{\sigma}^2(H_1(x)), \hat{\sigma}^2(H_2(x)), \ldots, \hat{\sigma}^2(H_q(x)))$. Here $H_j(x)$ denotes the j-th component set of $H(x)$.

5) Let $V = P\ Q$. The robust estimates of multivariate location and dispersion are $\hat{\mu} = V\ \Delta$ and $\hat{\Sigma} = V\ \Gamma V\ T$, respectively.

The use of OGK in calculating the robust estimations of μ and Σ will make them less influenced by outliers and more close to the true values of μ and Σ, hence real outliers will have a higher probability of being spotted. The OGK computes the multivariate dispersion estimates based on pair wise robust correlation or covariance estimation, which reduces the computational complexity in the data dimension q from exponential (2^q) to quadratic (q^2) [Alqallaf, et al. 2002].

5.1.4 COMPUTING THE THRESHOLD Θ_0

Previously we discussed that $MD_i^2 = \left((x_i - \mu)^T \Sigma^{-1} (x_i - \mu)\right)$ is distributed as the chi-squared distribution with q- degrees of freedom [Liu et al. 2007] [Filzmoser]. Hence, a simple solution for calculating the threshold Θ_0 could simply be to use $X_q^2(\alpha)$ as the threshold, where α is in the upper 90'th percentile of the chi-squared distribution. We could choose $\alpha = 0{,}975$ i.e 97.5% of the Mahalanobis squared distances is expected to be equal to or lesser then the chi-squared distributions value at 97.5% This is a fixed value and can therefore produce false positives when the multivariate dataset has not so normal distributed variables (attributes), which could be the circumstances in real life sensor networks. Therefore we propose the use of a mor dynamic threshold based on the weight function in the reweighting step of [Maronna

& Zamar 2002]. Where they define the threshold for the Mahalanobis distances in the reweighting step in the following manner:

$$d_0 = \frac{X_q^2(\alpha) * median(MD_1^2, MD_2^2, \dots, MD_n^2)}{X_q^2(0.5)},$$

Where n is the number of nodes in the multivariate sample data set. In our detection algorithm we use d_0 as the threshold Θ_0 for detection. If a node has a Mahalanobis squared distance larger than d_0, it is considered to be an insider attacker. We expect that this threshold will produce fewer false positives than the fixed threshold, as a result of its dependency of the calculated Mahalanobis squared distances distribution.

5.2 MODIFICATIONS

Our algorithm is based on the proposal of F.Liu, X.Cheng and D.Chen [Liu et al. 2007], but with some small modifications. While [Liu et al. 2007] operates with two neighborhoods, our algorithm operates with three. The third neighborhood is a result of the data aggregation scheme, as it introduces the cluster as a separate neighborhood N_1^*. This is needed because of the assumption we make that the subject of the sensor measurements is locked to the region of the cluster, and may differ from the subject of the sensor measurements of the other nodes in the two other neighborhoods N_1 and N_2. Hence the sensor measurements will only be evaluated by the intrusion detection algorithm inside the cluster.

The next modification we introduce is a different calculation of the threshold for decision making regarding whether a node is to be regarded as an outlier node or not. We propose a dynamically computed threshold which depends on the actual distribution of the Mahalanobis squared distances (5.1.4). We introduce this modification because we assume that it will reduce the number of falsely accused nodes in the network, when no outliers are present.

The third modification we propose is the use of two different values for α in the calculation of the thresholds with the use of $X_q^2(\alpha)$. We propose to increase the value of α when the threshold for the cluster neighborhood is to be calculated. We find this necessary as a result of the assumption that the clusters will be very sparse networks with few nodes and, as we will

observe in the simulation results, the false detection probability tends to increase as the neighborhood becomes increasingly sparse. This is a result of the algorithm having less information to calculate the robust statistics of the Mahalanobis squared distances.

The fourth and final modification I introduce, is the use of MAD as the sample standard deviation in the "False information filtering" protocol (5.3.3). The reason I introduce this modification is that the MAD calculations is less effected by the presence of outliers, as it relies on the median instead of the mean. Hence, it should be more effective in detecting real outliers.

5.3 MY PROPOSED SCHEME FOR INTRUSION DETECTION

In this section we introduce the proposal for an intrusion detection algorithm.

5.3.1 INTRODUCTION

This algorithm uses the anomaly detection technique in detection of intrusion. The proposed intrusion detection algorithm collects attribute vectors for each node in its neighborhood and comprises a data set consisting of all attribute vectors.

5.3.2 ASSUMPTIONS

We consider a cluster based sensor network with N sensors uniformly distributed in clusters within the network area. We assume that the data aggregation is only done by the clusterhead of each cluster, and there is only one level of clusters (i.e flat aggregation topology).

All sensors have the same capabilities, and communicate through bidirectional links. We assume sensors in the clusters are burdened with similar workloads and sensor readings. We assume sensors in the close proximity of each other, also in different clusters, behave similarly under normal conditions.

An *insider attacker* is a sensor under the control of an adversary. It has the same network resource as a normal sensor, but its behavior is different compared to others. For example, an insider attacker may drop or broadcast excessive packets, report false readings that deviate significantly from other readings of neighboring sensors, etc.

We assume each sensor works in promiscuous mode intermittently and listens for activities of direct neighbors. Which means, sensor x can overhear the message to and from the immediate neighbor x_i no matter whether or not x is involved in the communication. The monitoring is conducted intermittently, and x_i's networking behavior is modeled by a q-component attribute vector $f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T$ with each component describing x_i's activity in one aspect. For each fixed j $(1 \leq j \leq q)$, the component $f_j(x_i)$ represents the actual monitoring result, such as the number of packets being dropped or broadcasted during one monitoring period. Therefore, $f_j(x_i)$ can be continuous or discrete.

We assume that the base station has a complete overview of the entire network and controls the initial clustering and cluster head elections. We also assume that the base station controls the adding of new sensor nodes to the network.

We assume that there exists a clustering protocol that can handle the reelection of cluster heads based on remaining energy levels, and that this protocol can handle cluster head removal in case of faulty or compromised cluster heads detected by the intrusion detection system.

We assume that in any local area of the sensor field, all $f(x_i)$, where x_i's are normal sensors, follow the same multivariate normal distribution. After an internal adversary is detected, the cluster head of that node should remove it from the cluster and send a report to the base station. In addition, we assume there exists a MAC layer protocol to coordinate neighboring broadcastings such that no collision occurs.

5.3.3 OPERATIONS

In this section we describe the operations of the proposed intrusion detection system

Clustering method

Due to the assumption that the base station has a complete overview over the network, prior to deployment, we propose that the base station coordinates the clustering operation, and elects the cluster head of each cluster after the sensor nodes are deployed in the working environment. For the remaining network lifetime the election of new cluster heads are performed within the individual cluster based on remaining reported energy level of nodes. New cluster heads are elected at a timely manner, when the current cluster head fails or if the

current cluster head is found to be an insider attacker by the IDS. New nodes are not allowed into the clusters unless the base station explicitly organizes the introduction of the new node.

Intrusion detection

Our algorithm consists of four main phases with some sub-phases to each main phase. The phases are:

- Information collection phase
  - Information collection for N_1 and N_2
  - Information collection phase for cluster (sensor data)
- Information filtering phase
  - Information filtering phase for N_2
- Outlier detection phase
  - Outlier detection phase for N_1 and N_2
  - Outlier detection phase for cluster
  - Data aggregation for sensor data in cluster
- Voting phase
  - Voting phase for N_1 and N_2
  - Voting phase for cluster

Information collection phase

Information collection in N_1 and N_2

Let $N_1(x)$ denote a bounded closed set of R^2 that can be directly monitored by sensor x. Specifically, $N_1(x)$ is x's one-hop neighborhood.

Let $N(x)(\supseteq N_1(x)$ denote another closed set of R^2 that contains the sensor x and additional $n - 1$ nearest sensors. The set $N_2(x)$ represents another neighborhood of x, whose selection is determined by the node density in the network. For a dense network, we can simply choose $N_2(x) = N_1(x)$, while for a sparse network, $N_2(x)$ may include x's two-hop neighbors, sensor x monitor the activities of sensors in $N_1(x)$ and express the results using q-component attribute vectors. Then, the observed results are broadcasted within the neighborhood $N_2(x)$, so that sensor x obtains a set $F_1(x)$of attribute vectors, where $F_1(x) = \{f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T | x_i \in N_2(x)\}$.

During this phase sensor x will have acquired a dataset which should represent the true activities of the neighborhood $N_2(x)$.

Information collection within the cluster

Let $N_1^*(x)$ denote a bounded closed set of R^2 that can be directly monitored by sensor x. Specifically, $N_1^*(x)$ is x's cluster.

Sensor x monitors the activities of sensors in $N_1^*(x)$ and express the results using $q-1$ - component attribute vectors. Then, the observed results are broadcasted within the neighborhood $N_1^*(x)$ *together with x's sensor reading*, so that sensor x obtains a set $F_2(x)$ of attribute vectors, where $F_2(x) = \{f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_{q-1}(x_i), f_q(x_i))^T \mid x_i \in N_1^*(x)\}$, where $f_q(x_i)$ is the sensor readings of x_i

During this phase sensor x will have acquired a dataset which should represent the true activities of the neighborhood $N_1^*(x)$.

Information filtering phase

After the information collection phase for $N_1(x)$ and $N_2(x)$, sensor x will have acquired a dataset $F_1(x)$ which should represent the true activities of the neighborhood $N_2(x)$. However there may exist insider attackers within the neighborhood $N_1(x)$ which could have modified and forwarded a monitoring result of one or more nodes in the neighborhood $N_2(x) - N_1(x)$. Hence, node x should filter the results as much as possible in order to produce accurate detections of outliers according to the modified *"trust-based information filtering protocol"* proposed by[Liu et al. 2007].

Based on the direct neighborhood monitoring, sensor x assigns a trust value to each 1-hop neighbor $x_i \in N_1(x)$. The trust value $T(x_i)$ is in the range $\{0,1\}$, where values closer to 1 indicates a higher trust in that the neighbor x_i is a normal sensor.

The consideration is that the sensors in close proximity should behave similarly. This indicates that $T(x_i)$ can be computed according to the degree of x_i's deviation from the neighborhood activities.

The modified trust-based information filtering protocol is described in detail next.

Let $F_1(x)$ denote the attribute vectors of $N_1(x)$, i.e.
$F_1(x) = \{f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T | x_i \in N_2(x)\}$

Let $\hat{\mu}_j$, $\hat{\sigma}_j$ denote the sample mean and sample standard deviation of $F_1(x)$'s j-th component set $F_{1,j}(x) = \{f_j(x_i) | x_i \in N_1(x)\}$, respectively, i.e.,

$$\hat{\mu}_j = \frac{1}{n_1} \sum_{i=1}^{n_1} f_j(x_i),$$

$$\hat{\sigma}_j = MAD(f_j(x_i))$$

where n_1 is the number of nodes in $N_1(x)$, and MAD(Y) = median(|Y − median(Y)|) is our proposal for the calculation of the sample standard deviation. In [Liu et al. 2007] the sample standard deviation is calculated in the following manner:

$$\hat{\sigma}_j = \sqrt{\frac{1}{n_1 - 1} \sum_{i=1}^{n_1} \left(f_j(x_i) - \hat{\mu}_j\right)^2},$$

Sensor x first standardizes each data set $F_{1,j}(x)$ $(1 \leq j \leq q)$ and computes the absolute values to obtain

$F_{1,j}(x) = \{f_j(x_i) | x_i \in N_1(x)\}$,

Where

$$f'_j(x_i) = \left| \frac{f_j(x_i) - \hat{\mu}_j}{\hat{\sigma}_j} \right|.$$

For each $x_i \in N_1(x)$, sensor x computes the maximum attribute component $f'_M(x_i) = max\{f'_j(x_i) | 1 \leq j \leq q\}$, which indicates the "extremeness" of x_i's deviation from the neighborhood activities. Then, the trust value is computed as

$$T(x_i) = \frac{f_M^m}{f'_M(x_i)}$$

Where

$$f_M^m = \min\{f'_M(x_i) | x_i \in N_1(x)\}.$$

A node $x_{jT}(1 \leq T \leq t)$ is said to be the reliable relay node for x_j if

$$T(x_{jT}) = \max\{T(x_{js}) | 1 \leq s \leq t\}$$

$$T(x_{jT}) \geq T_{min}$$

Where

$$T_{min} = f_M^m/7$$

T_{min} is the minimum acceptable trust value.

The decision of defining $T_{min} = f_M^m/7$, is based on conducted simulations and should be regarded as such. However, as the simulation results later in the thesis will show, $T_{min} = f_M^m/7$ appears to be a good threshold for minimum acceptable trust value.

Sensor x will dismiss the information about $x_j \in N_2(x) - N_1(x)$ if no reliable relay node for x_j can be found in $N_1(x)$. Thus after filtering $F_1(x)$, sensor x will only consider information carried by sensors from the set $\widetilde{N}_2(x)$, where $\widetilde{N}_2(x) \in N_2(x)$. $\widetilde{N}_2(x)$ contains x's direct neighbors in $N_1(x)$ and x's neighbors in $N_2(x) - N_1(x)$ that have a trustworthy relay node in $N_1(x)$. The new data set to be assessed by sensor x is $\tilde{F}_1(x) = \{f(x_i) = (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T | x_i \in \widetilde{N}_2(x)\}$

Insider attacker detection phase for $\widetilde{N}_2(x)$

Sensor x detects if any insider attackers exist by studying the data set $\tilde{F}_1(x)$. The detection is conducted by computing the Mahalanobis squared distance of each nodes attribute vector and comparing it to the threshold Θ_0 calculated with the chi-squared value $X_q^2(\alpha)$ (5.1.4). Sensor x_i is determined to be an insider attacker if the distance is larger than the threshold θ_0. The detection phase is conducted in the following manner:

1. Compute the Orthogonalized Gnanadesikan-Kettenring (OGK) robust estimates of the sample set $\tilde{F}_1(x)$ means μ and covariance matrix Σ as described in 5.1.3
2. Compute the Mahalanobis squared distances of each $f(x_i)|x_i \in \widetilde{N}_2(x)$
3. Compute the threshold θ_0 as described in 5.1.4.
4. Compare the calculated Mahalanobis squared distances to the threshold θ_0.
5. Mark the nodes with Mahalanobis squared distances larger than the threshold θ_0 as insider attackers. Mark the nodes with Mahalanobis squared distances lesser than the threshold θ_0 as normal nodes.
6. Broadcast the result from previous the step within the neighborhood $N'(x)$, where $\widetilde{N}_2(x) \subseteq N'(x)$. Choosing a larger neighborhood $N'(x)$ ensures that more nodes participate in the voting. At the same time node x will receive the results from others and registers the votes about its neighbors in $N_1^*(x)$.
7. After x receives the broadcasted results, it count the number of positive detections for each node in $N_1^*(x)$.If the proportion of positive detections for a node $x_i \in N_1^*(x)$, Then that node is decided to be an insider attacker.
    a. If x Is the cluster head, it removes the confirmed insider attacker from the cluster and notifies the base station.
    b. If x is a regular cluster node, it waits to see if the cluster head removes the confirmed attacker from the cluster.
    c. If the cluster head does nothing, x broadcasts an alarm message within the cluster specifying the confirmed insider attacker. If x receives alarm messages from other nodes in the cluster, it counts the number of received alarm messages.
        i. If the majority of the cluster has confirmed the specified node as an insider attacker, the cluster head is deemed as an insider attacker and removed by the cluster nodes. A new node is elected as cluster head and removes the confirmed insider attacker.
        ii. If the majority of the cluster has not confirmed the specified node as an insider attacker, x regards the detection as an error.
    d. If the cluster head tries to remove a node which is not detected by the other nodes in the cluster as an insider attacker, the other nodes will collude in removing the cluster head in the same manner as c.

Insider attacker detection phase for the cluster $N_1^*(x)$

Sensor x detects if any outliers exist by studying the data set $F_1^*(x)$. The detection is conducted by computing the Mahalanobis squared distance of each nodes attribute vector and comparing it to the threshold θ_0 calculated with the chi-squared value $X_q^2(\alpha)$. Sensor x_i is determined to be an insider attacker if the distance is larger than the threshold θ_0. The detection phase is conducted in the following manner:

1. Compute the Orthogonalized Gnanadesikan-Kettenring (OGK) robust estimates of the sample set $F_1^*(x)$ means μ and covariance matrix Σ as described in 5.1.3
2. Compute the Mahalanobis squared distances of each $f(x_i)|x_i \in N_1^*(x)$.
3. Compute the threshold θ_0 as described in 5.1.4.
4. Compare the calculated Mahalanobis squared distances to the threshold θ_0.
5. Mark the nodes with Mahalanobis squared distances larger than the threshold θ_0 as insider attackers. Mark the nodes with Mahalanobis squared distances lesser than the threshold θ_0 as normal nodes.
6. Broadcast the result from previous the step within the neighborhood $N_1^*(x)$. At the same time node x will receive the results from the other nodes in $N_1^*(x)$.
7. After x receives the broadcasted results, it count the number of positive detections for each node in $N_1^*(x)$.If the proportion of positive detections for a node $x_i \in N_1^*(x)$, Then that node is decided to be an insider attacker.
    a. If x Is the cluster head, it removes the confirmed insider attacker from the cluster and notifies the base station.
    b. If x is a regular cluster node, it waits to see if the cluster head removes the confirmed attacker from the cluster.
    c. If the cluster head does nothing, x broadcasts an alarm message within the cluster specifying the cluster head as an insider attacker.
        i. If the majority of the nodes in the cluster broadcast an alarm, the cluster head is deemed as an insider attacker and removed by the cluster nodes. A new node is elected as cluster head and removes the confirmed insider attacker.
        ii. If the majority of the nodes in the cluster broadcast an alarm, x regards the detection as an error.

d. If the cluster head tries to remove a node which is not detected by the other nodes in the cluster as an insider attacker, the other nodes will collude in removing the cluster head in the same manner as c.

Data aggregation phase for the cluster $N_1^*(x)$

During the insider detection phase for the cluster $N_1^*(x)$, each node computes the Orthogonalized Gnanadesikan-Kettenring (OGK) robust estimates of the sample set $F_1^*(x)$ means μ and covariance matrix Σ as described in 5.1.3. We propose the use of the robustly calculated mean of the sensor readings as the aggregation result. However, if an insider attacker has been identified, the sensor readings of that node should be dismissed and the robust estimate should be calculated from the resulting sample set $\tilde{F}_1^*(x)$. The data aggregation phase is conducted in the following manner:

1. If no insider attacker has been identified, the cluster head uses the robustly calculated mean value of the sensor readings.
2. If an insider attacker has been identified, remove this nodes attribute vector from the sample data set $F_1^*(x)$ and compute the new robust estimate for the sensor readings from the resulting sample set $\tilde{F}_1^*(x)$. The resulting robustly calculated mean is used as the aggregation result.

However, if the cluster head has been compromised, the other nodes in the cluster should monitor the aggregation result of the cluster head for discrepancies.

As a result of the intrusion detection phase, all nodes in the cluster has the exact same sample set $F_1^*(x)$. Hence, they have calculated the exact same robust mean values for the sensor readings. Thus, if the aggregation result from the cluster head differs from the robust means calculated by each of the other nodes, it will be discovered, and the other nodes in the cluster regards the cluster head as an insider attacker and removes conducts a mutiny. This way the network is protected from compromised cluster heads trying to alter the aggregation results.

6. SIMULATION STUDY

Simulations are preformed in Wolfram Mathematica version 6.0.0. Needed package loaded: "MultivariateStatistics`".

System: Windows Vista Ultimate SP1

Graphs and tables are constructed in Microsoft Excel, Microsoft office 2007.

SCENARIOS

During this section we denote insider attackers as outliers or outlying nodes. In order to observe how our modifications perform, we compare the results with the results of the original algorithm proposed by [Liu et al. 2007]. In our simulation results we denote our proposal for the dynamic threshold by θ', and the original proposal for the fixed chi-squared threshold by θ.

We will simulate various scenarios in which insider attackers are present, and evaluated the theoretical performance of the algorithm.

We have limited the simulations to evaluate the performance of the algorithm on one node $x_i \in N(x)$, with a neighborhood $N_1^*(x)$, consisting of the 1-hop neighbors in the cluster, a neighborhood $N_1(x)$ consisting of all 1-hop neighbors (including the neighbors in the cluster), and a neighborhood $N_2(x)$ consisting of all 1 and 2-hop neighbors, where $N_1^*(x) \in N_1(x) \in N_2(x) \in N(x)$.

We will evaluate the algorithm in sparse networks with a cluster neighborhood $N_1^*(x)$ consisting of 5 nodes, and the 1-hop neighborhood $N_1(x)$, consisting of 10 nodes, while the neighborhood $N_2(x)$ consists of 20 nodes.

The behavior of each node is modeled by a vector containing q=3 attributes. As stated in section 5.3, the IDS only examine the sensor readings as an attribute within the cluster. Hence we only evaluate the sensor readings in the simulation scenario for $N_1^*(x)$.

We evaluate the IDS in six scenarios.

- In the first scenario we evaluate how our algorithm performs in the neighborhood $N_2(x)$, containing 20 nodes, while the percentage of outliers varies from 0 to 50%. We also evaluate how our proposal for the dynamic threshold Θ' performs as opposed to the fixed chi-squared threshold Θ proposed by [Liu et al. 2007], by comparing false alarm probability as well as the detection accuracy for both thresholds Θ' and Θ in $N_2(x)$

- In the second scenario we evaluate how our algorithm performs in the neighborhood $N_1(x)$ containing 10 nodes, while the percentage of outliers varies from 0 to 50%. We also evaluate how our proposal for the dynamic threshold Θ' performs as opposed to the fixed chi-squared threshold Θ proposed by [Liu et al. 2007], by comparing false alarm probability as well as the detection accuracy for both thresholds Θ' and Θ in $N_1(x)$.

- In the third scenario we evaluate how our algorithm performs in the neighborhood $N_1^*(x)$ consisting of only 5 nodes, while the number of outliers varies from 0 to 40% (0-2 nodes). We also observe how our proposal for the dynamic threshold Θ' performs as opposed to the fixed chi-squared threshold Θ proposed by [Liu et al. 2007], in $N_1^*(x)$.

- In the fourth scenario We evaluate the accuracy of the robust estimator μ'for the sensed data, which will be used as the data aggregation result, by comparing it to the true mean value of the sensed data from normal nodes only. We will also evaluate the degree of which an outlier can contaminate the aggregation result before the probability of getting caught is over 50%. This is the worst case scenario.

- In the fifth scenario we evaluate the detection accuracy and false alarm probability in $N_1^*(x)$ when the accuracy of the sensors increases and the standard deviation of the sensor data distribution decreases to 1, 0.1 and 0.01 percent of the true mean value. We will evaluate the degree of which an outlier can be deviating from the true mean value of the attributes, before it is deemed an outlier by the detection algorithm.

- In the sixth scenario we will evaluate the modified *false information filtering protocol* and evaluate how our proposal of using the MAD as the sample standard deviation σ compares with the original proposal of [Liu et al. 2007].

In all scenarios the outliers can be outlying in one single attribute, or all three attributes, we also assume that the data set $F_1(x_i)$ of attribute-vectors have already been filtered according to the modified "thrust-based information filtering protocol" described in 5.3.3.

All results are averaged over 10 runs consisting of 1000 iterations with randomly generated multivariate attribute values for every iteration.

The values of the attributes of the nodes are generated with the built in RandomReal [MultinormalDistribution [$\{\mu_1, \mu_2, \mu_3\}$,Σ,"number of vectors"]] function in Mathematica (Apendix A) where the values are drawn from $N_3(\mu_i, \Sigma)$, where $\mu_i = (\mu_1, \mu_2, \mu_3)$, and the variance-covariance matrix Σ is defined using the standard deviations $\sigma_i = (\sigma_1, \sigma_2, \sigma_3)$ and the correlation coefficient matrix $\rho = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$, which indicate no correlation between the attributes.

The variance-covariance matrix $\Sigma = (\Sigma_{ij})$can be determined by $\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$.

We will specify the values for μ_i and $\sigma_i$in each scenario.

DETERMINING THE CHI-SQUARED PERCENTILE

The thresholds Θ and Θ' are calculated with $X_3^2(0{,}975)$ for $N_1(x)$ and $N_2(x)$ and $X_3^2(0{,}9999)$ for $N_1^*(x)$. The reason for increasing the detection threshold in $N_1^*(x)$ is the tendency of the outlier detection algorithm to increase in false alarm probability in sparse networks and no outlying nodes are present. This is a result of the data set $F_2(x_i)$becoming small, and less information is used in calculating the robust statistics. Figures 6.1 and 6.2 below illustrate this tendency.

However we can observe that the detection accuracy does not suffers so much from this increase in the threshold, because the robust statistics employed in the calculation of the Mahalanobis distances will still punish outliers by assigning them with significantly larger distances in comparison to the normal nodes.

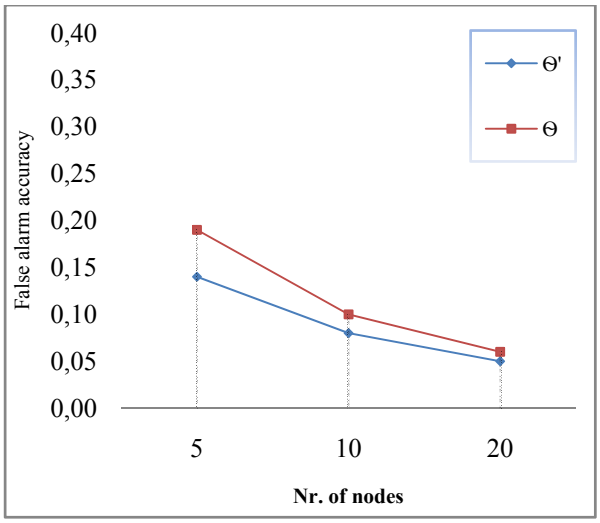


Figure 6.1: False alarm accuracy tendency $X_3^2(0{,}975)$

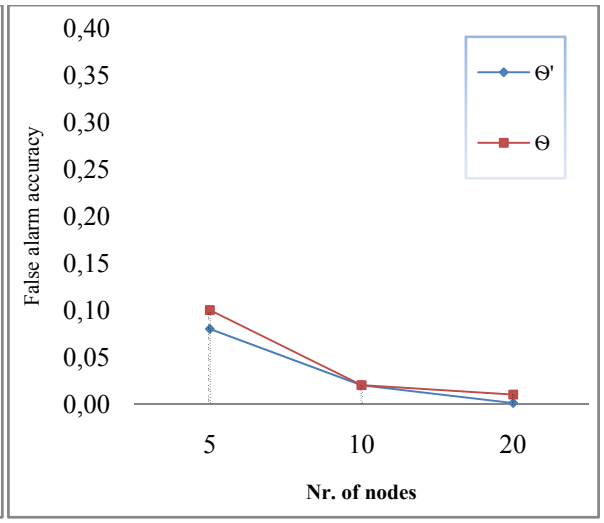


Figure 6.2: False alarm accuracy tendency $X_3^2(0{,}9999)$

Θ' denotes our proposal for the dynamic threshold, while Θ denotes the original proposal for the static threshold [Liu et al. 2007]

Figure 6.1 illustrates the false alarm probability when the number of nodes varies from 5 to 20 with 0 % outliers, and the threshold is calculated with $X_3^2(0{,}975)$.

Figure 6.2 illustrates the false alarm probability when the number of nodes varies from 5 to 20 with 0 % outliers, and the threshold is calculated with $X_3^2(0{,}999)$.

Observe that the false alarm accuracy is brought back to an acceptable level with a larger chi-squared percentile, and that the proposed dynamic threshold Θ' has a lower false alarm probability than the fixed threshold Θ.

FIRST SCENARIOTO

In this scenario there are 20 nodes in the neighborhood $N_2(x)$ of which the percentage of outliers varies from 0 to 50%. We evaluate how our proposal for the dynamic threshold Θ' performs as opposed to the fixed chi-squared threshold Θ by comparing false alarm probability as well as the detection accuracy for both thresholds.

- For normal nodes, the attributes are generated with $\mu_i = (10{,}50{,}100)$ and $\sigma_i = (1{,}5{,}10)$.
- For outliers in one attribute, the attribute values are generated with $\mu_i = (10{,}50{,}200)$ and $\sigma_i = (1{,}5{,}20)$,

- For outliers with three outlying attributes, the attribute values are generated with $\mu_i = (20,100,200)$ and $\sigma_i = (1,5,20)$.

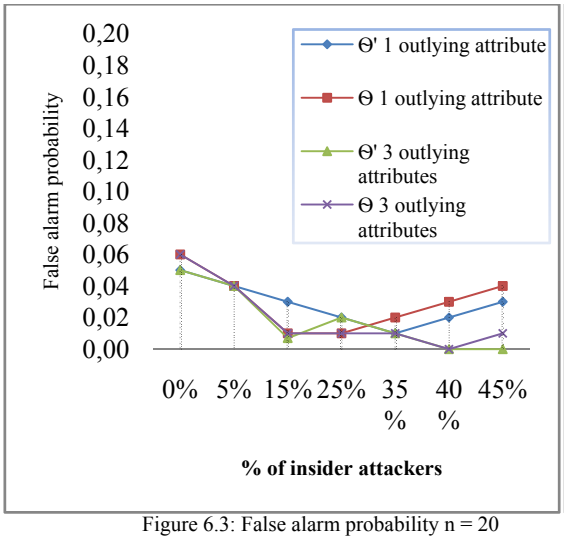


Figure 6.3: False alarm probability n = 20

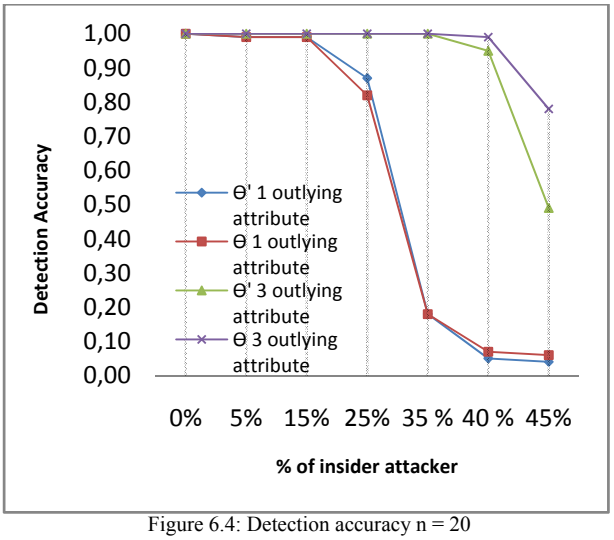


Figure 6.4: Detection accuracy n = 20

Θ' denotes our proposal for the dynamic threshold, while Θ denotes the original proposal for the static threshold [Liu et al. 2007]

Figure 6.3 illustrates the false alarm probability when the percentage of outliers varies from 0 to 40% with 20 nodes.

Figure 6.4 illustrates the detection accuracy when the percentage of outliers varies from 0 to 40% with 20 nodes.

SECOND SCENARIO

In this scenario there are 10 nodes in the neighborhood $N_1(x)$ of which the percentage of outliers varies from 0 to 40%. As in the first scenario we observe how our proposal for the dynamic threshold Θ' performs as opposed to the fixed chi-squared threshold Θ by comparing the false alarm probability as well as the detection accuracy for both thresholds.

- For normal nodes, the attributes are generated with $\mu_i = (10,50,100)$ and $\sigma_i = (1,5,10)$.
- For outliers in one attribute, the attribute values are generated with $\mu_i = (10,50,200)$ and $\sigma_i = (1,5,20)$,
- For outliers with three outlying attributes, the attribute values are generated with $\mu_i = (20,100,200)$ and $\sigma_i = (1,5,20)$.

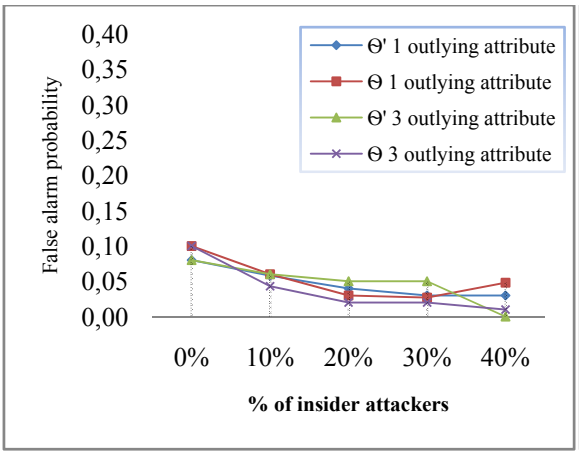


Figure 6.5: False alarm probability n = 10

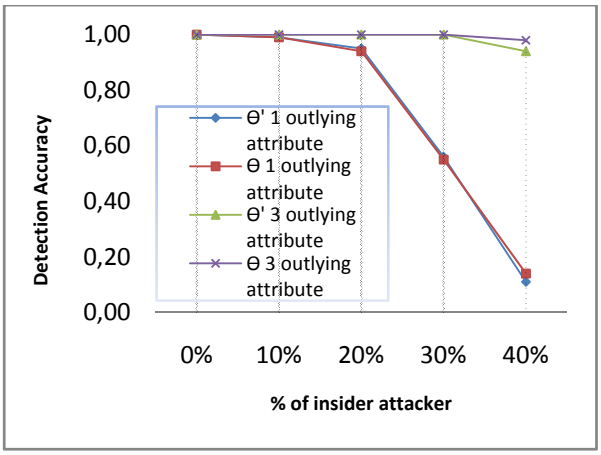


Figure 6.6: Detection accuracy n = 10

Θ' denotes our proposal for the dynamic threshold, while Θ denotes the original proposal for the static threshold [Liu et al. 2007]

Figure 6.5 illustrates the false alarm probability when the percentage of outliers varies from 0 to 40% with 10 nodes.

Figure 6.6 illustrates the detection accuracy when the percentage of outliers varies from 0 to 40% with 10 nodes.

THIRD SCENARIO

In this scenario there are 5 nodes in the neighborhood $N_1^*(x)$, and the percentage of outliers vary from 0 to 40% (0 to 2 nodes). As in the two previous scenarios, we also here evaluate how our proposal for the dynamic threshold Θ' performs as opposed to the fixed chi-squared threshold Θ by comparing the number of false positives with the use of both values as thresholds.

However we have chosen to increase the chi-squared threshold to the 0,9999 percentile as a result of the increase of false alarm probability we observed earlier, when the number of nodes in the neighborhood decreases.

- For normal nodes, the attributes are generated with $\mu_i = (10,50,100)$ and $\sigma_i = (1,5,10)$.
- For outliers in one attribute, the attribute values are generated with $\mu_i = (10,50,200)$ and $\sigma_i = (1,5,10)$,
- For outliers with three outlying attributes, the attribute values are generated with $\mu_i = (20,100,200)$ and $\sigma_i = (1,5,10)$.

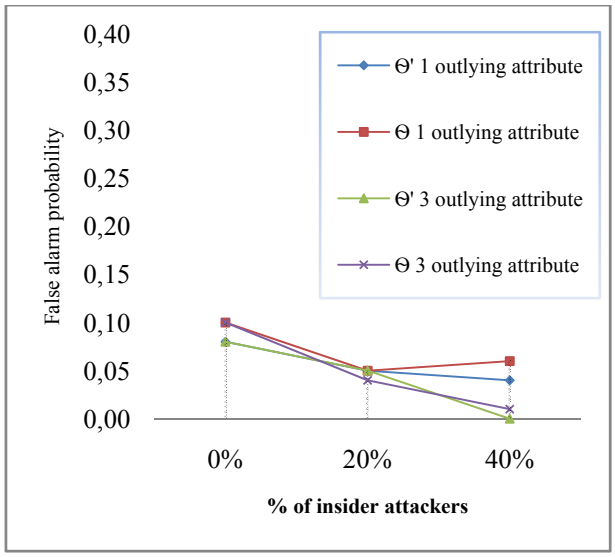


Figure 6.7: False alarm probability n = 5

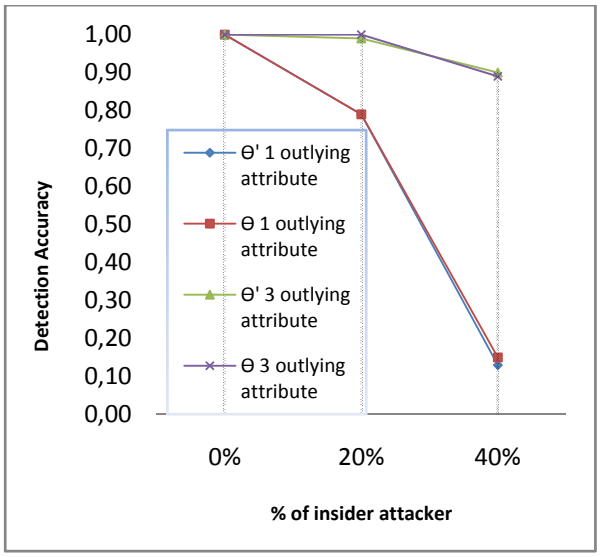


Figure 6.8: Detection accuracy n = 5

Θ' denotes our proposal for the dynamic threshold, while Θ denotes the original proposal for the static threshold [Liu et al. 2007]

Figure 6.7 illustrates the false alarm probability when the percentage of outliers varies from 0 to 40% with 5 nodes.

Figure 6.8 illustrates the detection accuracy when the percentage of outliers varies from 0 to 40% with 5 nodes.

FOURTH SCENARIO

In this scenario there are 5 nodes in the neighborhood $N_1^*(x)$, and 0 outliers. We evaluate the accuracy of the robust mean estimation of the sensed data distribution, which will be used as the data aggregation result in our proposed IDS, by comparing it with the true mean of the sensed data set. We evaluate the accuracy of the roust mean astimation when the standard deviation σ of the normal sensor data distribution varies between 0.01%, 0,1% and 1% of the true mean.

- For all nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (100,10,1)$.

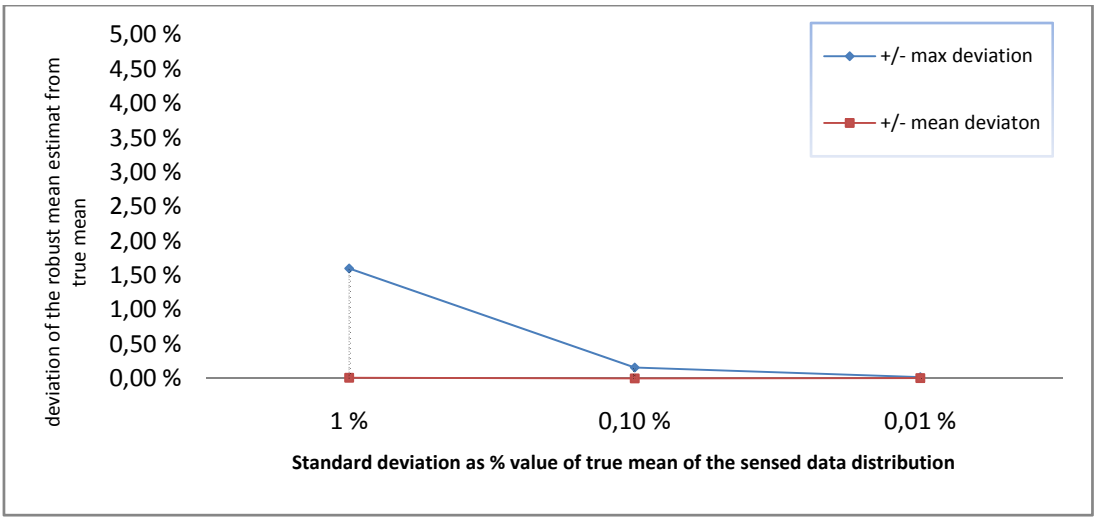


Figure 6.9: max deviation of the robust mean estimate μ' from the true mean μ

(The Max +/- gives the maximal observed deviations, while the Mean +/- gives the mean of the deviations)

Figure 6.9 illustrates how much the robust mean estimate deviates from the true mean of the sensed data with 0% outliers.

FIFTH SCENARIO

In the fifth scenario we evaluate the detection accuracy and false alarm probability in $N_1^*(x)$ when the accuracy of the sensor readings increases and the standard deviation of the sensor data distribution decreases to 1, 0.1 and 0.01 percent of the true mean.

We will evaluate the degree of which an outlier can be deviating from the true mean of the attributes, before he is deemed an outlier by the detection algorithm.

Part 1

20% outliers

σ =1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (100,100,100)$.
- For outliers in one attribute, the attribute values are generated with
  - $\mu_i = (10000,10000,12000)$ and $\sigma_i = (100,100,0.0001)$, for 20% outlying value.

- $\mu_i = (10000,10000,11500)$ and $\sigma_i = (100,100,0.0001)$, for 15% outlying value
- $\mu_i = (10000,10000,11000)$ and $\sigma_i = (100,100,0.0001)$, for 10% outlying value
- $\mu_i = (10000,10000,10500)$ and $\sigma_i = (100,100,0.0001)$, for 5% outlying value

σ =0.1% of the mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (10,10,10)$.
- For outliers in one attribute, the attribute values are generated with
  - $\mu_i = (1000,1000,1020)$ and $\sigma_i = (10,10,0.0001)$, for 2% outlying value.
  - $\mu_i = (1000,1000,1010)$ and $\sigma_i = (10,10,0.0001)$, for 1% outlying value
  - $\mu_i = (1000,1000,1005)$ and $\sigma_i = (10,10,0.0001)$, for 0.5% outlying value
  - $\mu_i = (1000,1000,1003)$ and $\sigma_i = (10,10,0.0001)$, for 0.3% outlying value

σ =0.01% of the mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (1,1,1)$.
- For outliers in one attribute, the attribute values are generated with
  - $\mu_i = (10000,10000,10020)$ and $\sigma_i = (1,1,0.0001)$, for 0,2% outlying value.
  - $\mu_i = (10000,10000,10010)$ and $\sigma_i = (1,1,0.0001)$, for 0,1% outlying value
  - $\mu_i = (10000,10000,10005)$ and $\sigma_i = (1,1,0.0001)$, for 0.05% outlying value
  - $\mu_i = (10000,10000,10003)$ and $\sigma_i = (1,1,0.0001)$, for 0.03% outlying value

20% outliers

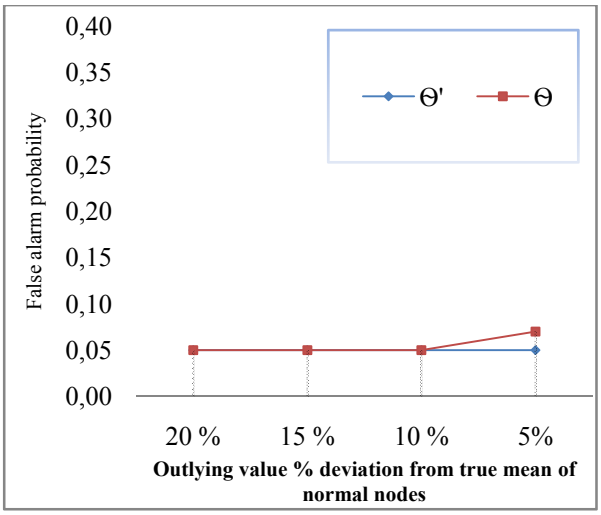


Figure 6.10: False alarm probability when σ = 1%

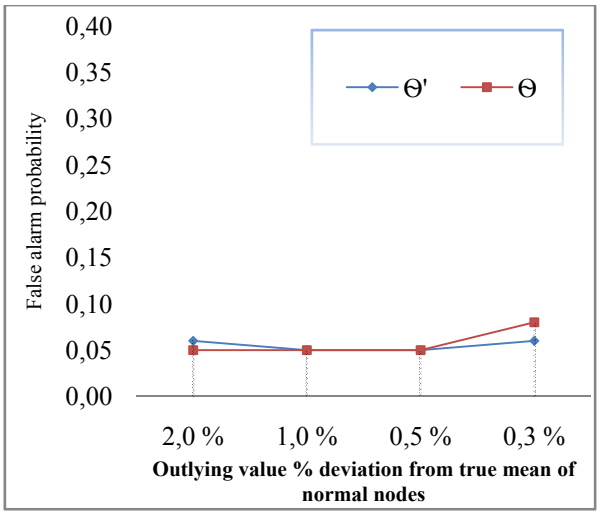


Figure 6.11: False alarm probability when σ = 0.1%

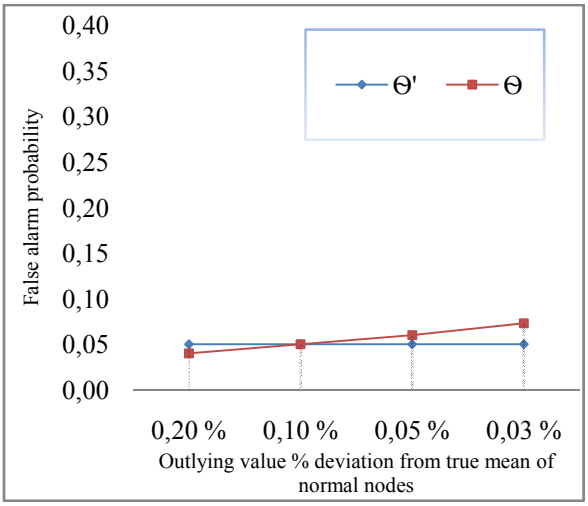


Figure 6.12: False alarm probability when σ = 0.01%

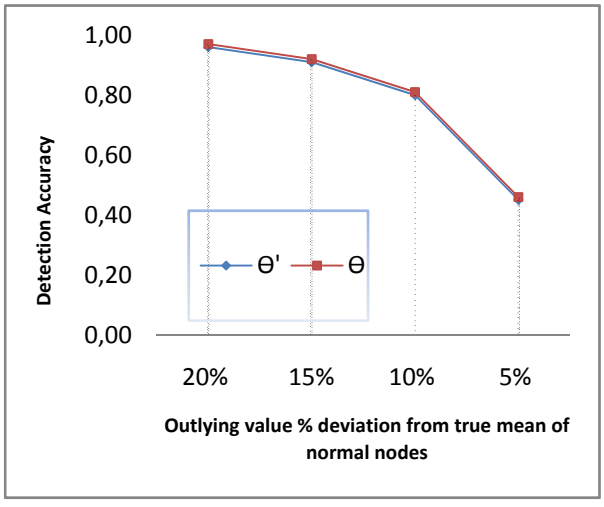


Figure 6.13: Detection accuracy when σ = 1%

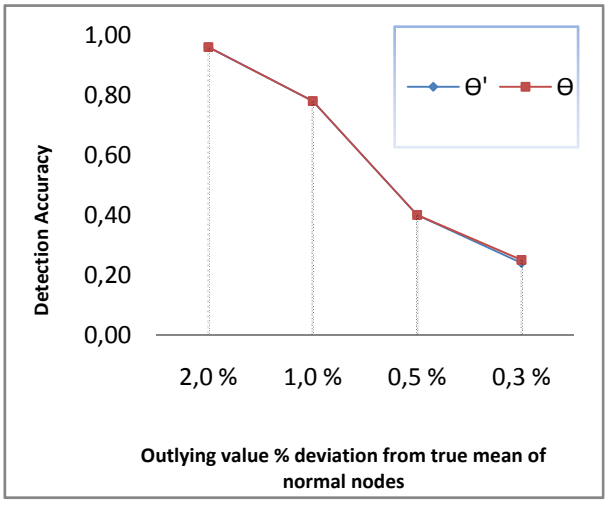


Figure 6.14: Detection accuracy when σ = 0.1%

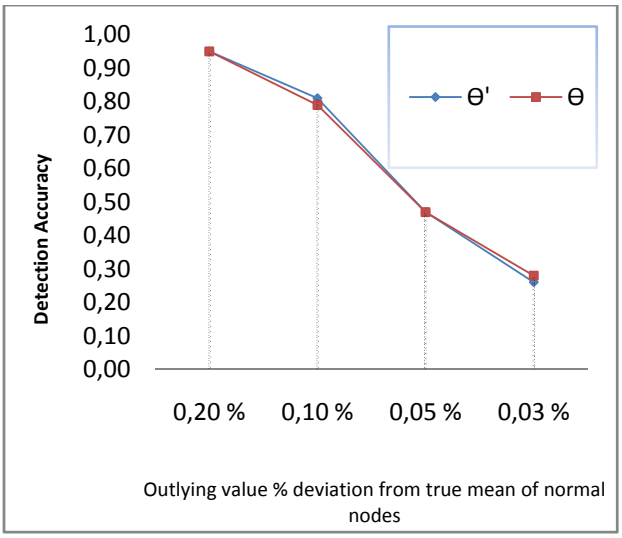


Figure 6.15: Detection accuracy when σ = 0.01%

Θ' denotes our proposal for the dynamic threshold, while Θ denotes the original proposal for the static threshold [Liu et al. 2007]

Figs. 6.10 and 6.13 illustrate how the false alarm probability and detection accuracy changes when 20% outliers (1 node) are "decreasingly outlying" and the sensed data have a 1% standard deviation σ from the true mean.

Figs. 6.11 and 6.14 illustrate how the false alarm probability and detection accuracy changes when 20% outliers (1 node) are "decreasingly outlying" and the sensed data have a 0.1% standard deviation σ from the true mean.

Figs. 6.12 and 6.15 illustrate how the false alarm probability and detection accuracy changes when 20% outliers (1 node) are "decreasingly outlying" and the sensed data have a 0.01% standard deviation σ from the true mean.

40% outliers

σ =1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (100,100,100)$.
- For outliers in one attribute, the attribute values are generated with
  - $\mu_i = (10000,10000,16000)$ and $\sigma_i = (100,100,0.0001)$, for 60% outlying value.

- $\mu_i = (10000,10000,15000)$ and $\sigma_i = (100,100,0.0001)$, for 50% outlying value
- $\mu_i = (10000,10000,14000)$ and $\sigma_i = (100,100,0.0001)$, for 40% outlying value
- $\mu_i = (10000,10000,13000)$ and $\sigma_i = (100,100,0.0001)$, for 30% outlying value

σ =0.1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (10,10,10)$.
- For outliers in one attribute, the attribute values are generated with
  - $\mu_i = (10000,10000,10600)$ and $\sigma_i = (10,10,0.0001)$, for 6% outlying value.
  - $\mu_i = (10000,10000,10500)$ and $\sigma_i = (10,10,0.0001)$, for 5% outlying value
  - $\mu_i = (10000,10000,10400)$ and $\sigma_i = (10,10,0.0001)$, for 4% outlying value
  - $\mu_i = (10000,10000,10300)$ and $\sigma_i = (10,10,0.0001)$, for 3% outlying value

σ =0.01% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (1,1,1)$.
- For outliers in one attribute, the attribute values are generated with
  - $\mu_i = (10000,10000,10060)$ and $\sigma_i = (1,1,0.0001)$, for 0,6% outlying value.
  - $\mu_i = (10000,10000,10050)$ and $\sigma_i = (1,1,0.0001)$, for 0,5% outlying value
  - $\mu_i = (10000,10000,10040)$ and $\sigma_i = (1,1,0.0001)$, for 0.4% outlying value
  - $\mu_i = (10000,10000,10030)$ and $\sigma_i = (1,1,0.0001)$, for 0.3% outlying value

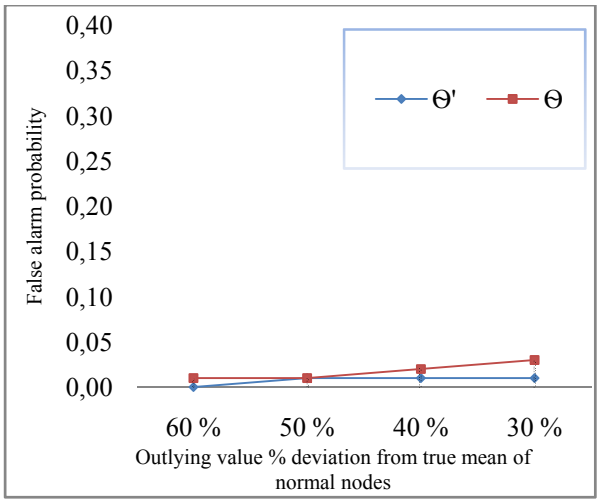


Figure 6.16: False alarm probability when σ = 1%

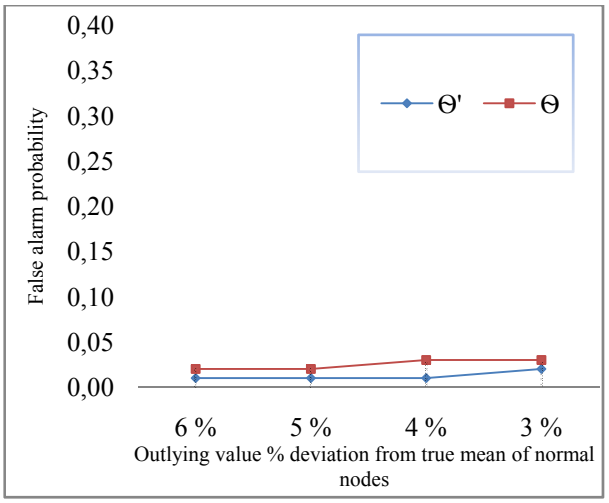


Figure 6.17: False alarm probability when σ = 0.1%

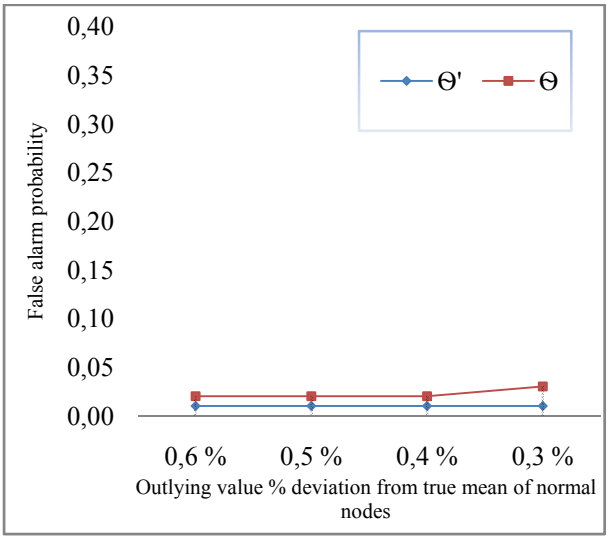


Figure 6.18: False alarm probability when σ = 0.01%

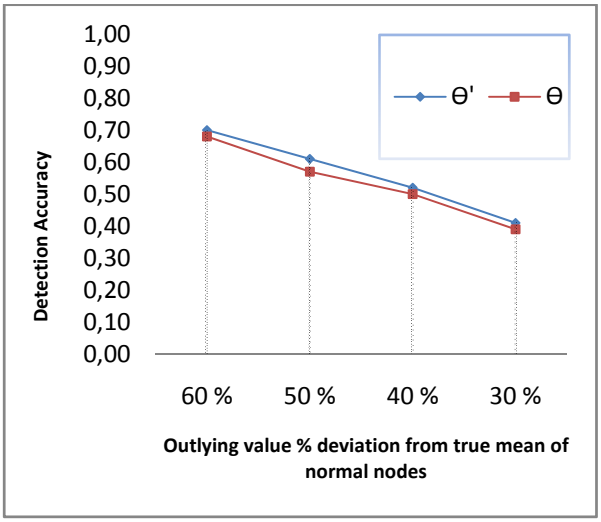


Figure 6.19: Detection accuracy when σ = 1%

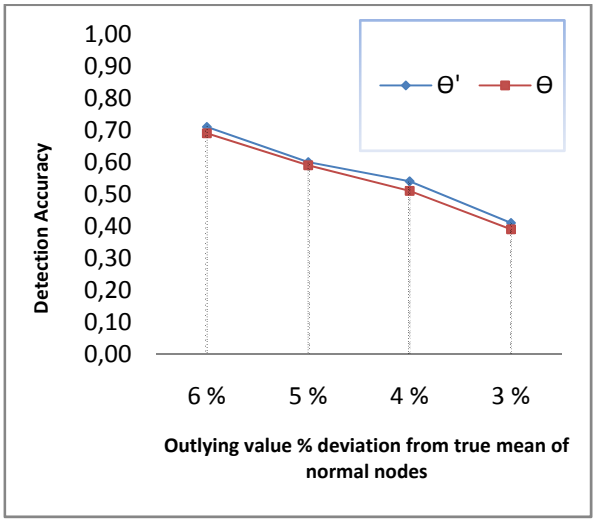


Figure 6.20: Detection accuracy when σ = 0.1%

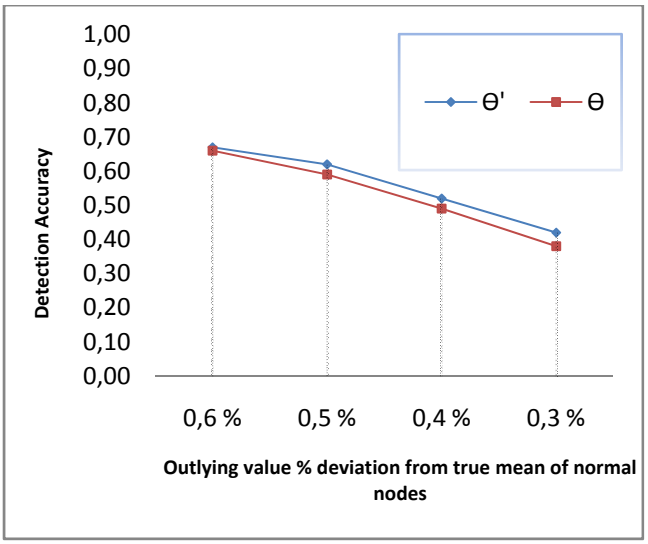


Figure 6.21: Detection accuracy when σ = 0.01%

Θ' denotes our proposal for the dynamic threshold, while Θ denotes the original proposal for the static threshold [Liu et al. 2007]

Figs. 6.16 and 6.19 illustrate how the false alarm probability and detection accuracy changes when 40% outliers are "decreasingly outlying" and the sensed data have a 1% standard deviation σ from the true mean.

Figs. 6.17 and 6.20 illustrate how the false alarm probability and detection accuracy changes when 40% outliers are "decreasingly outlying" and the sensed data have a 0.1% standard deviation σ from the true mean.

Figs. 6.18 and 6.21 illustrate how the false alarm probability and detection accuracy changes when 40% outliers are "decreasingly outlying" and the sensed data have a 0.01% standard deviation σ from the true mean.

Part 2

Now that we have observed the degree of which an outlier can deviate from the normal attribute distribution before getting caught, it would be interesting to examine the detection accuracy and false alarm probability of the algorithm when outliers deviates beyond these values. In the following observations, the outliers have had an outlying value of 200% the true mean of the normal attribute distribution.

σ =1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,15000,20000)$ and $\sigma_i = (100,100,200)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,15000,60000)$ and $\sigma_i = (100,100,300)$, for 200% outlying value.

σ =0.1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,15000,20000)$ and $\sigma_i = (10,10,20)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,15000,60000)$ and $\sigma_i = (10,10,30)$, for 200% outlying value.

σ =0.01% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,15000,20000)$ and $\sigma_i = (1,1,2)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,15000,60000)$ and $\sigma_i = (1,1,3)$, for 200% outlying value.

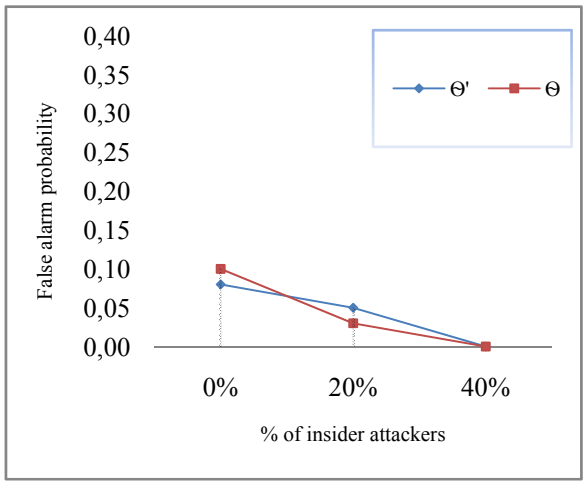


Figure 6.22: False alarm probability when σ = 0.01%

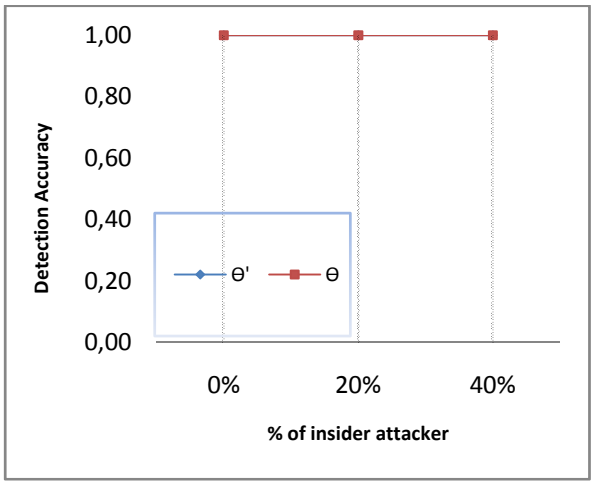


Figure 6.23: Detection accuracy when σ = 0.01%

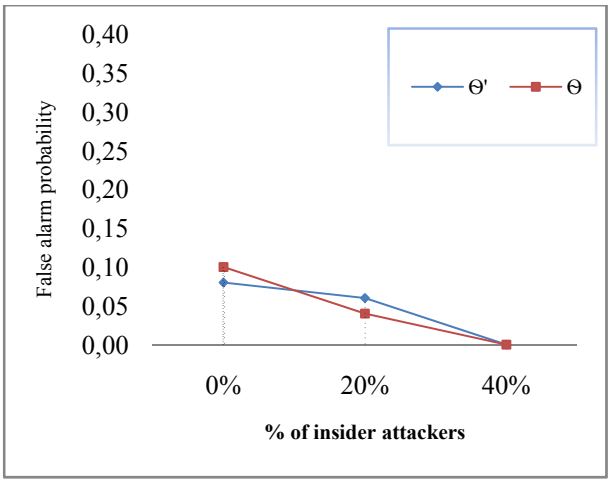


Figure 6.24: False alarm probability when σ = 0.1%

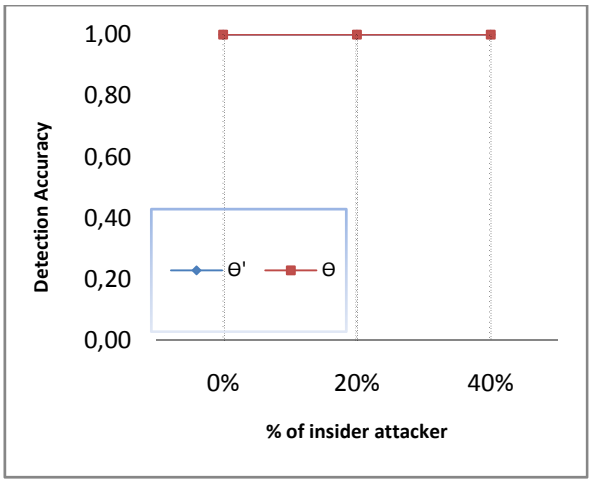


Figure 6.25: Detection accuracy when σ = 0.1%

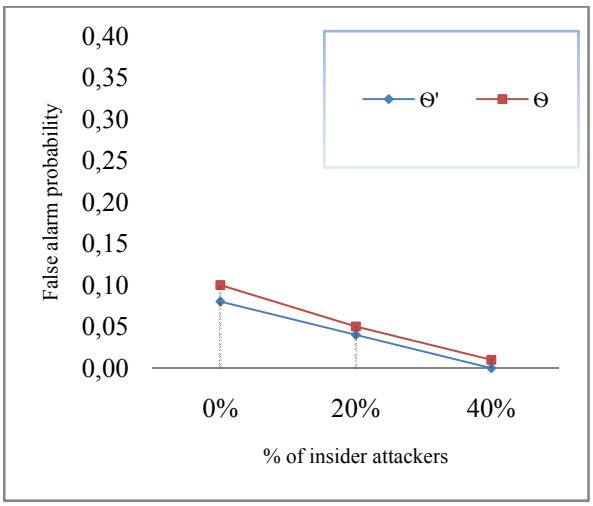


Figure 6.26: False alarm probability when σ = 1%

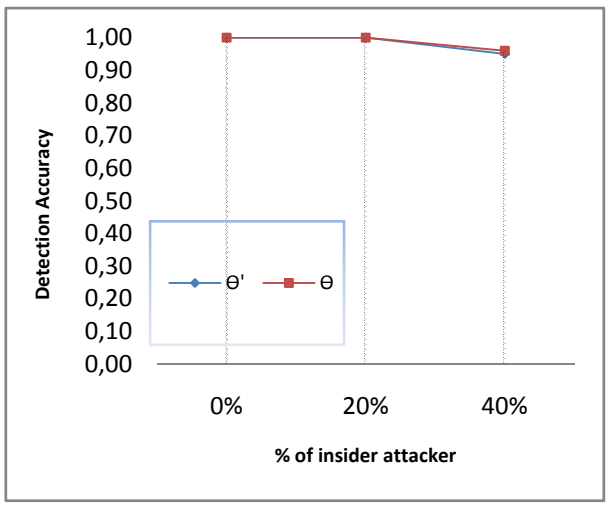


Figure 6.27: Detection accuracy when σ = 1%

Θ' denotes our proposal for the dynamic threshold, while Θ denotes the original proposal for the static threshold [Liu et al. 2007]

Figure 6.22 illustrates the false alarm probability when the outlying value is 200% of true mean and the standard deviation of the normal distribution for all three attributes is 0.01% of true mean, while the percentage of outliers varies from 0 to 40%.

Figure 6.24 illustrates the false alarm probability when the outlying value is 200% of true mean and the standard deviation of the normal distribution for all three attributes is 0.1% of true mean, while the percentage of outliers varies from 0 to 40%.

Figure 6.26 illustrates the false alarm probability when the outlying value is 200% of true mean and the standard deviation of the normal distribution for all three attributes is 1% of true mean, while the percentage of outliers varies from 0 to 40%.

Figure 6.23 illustrates the detection accuracy when the outlying value is 200% of true mean and the standard deviation of the normal distribution for all three attributes is 0.01% of true mean, while the percentage of outliers varies from 0 to 40%.

Figure 6.25 illustrates the detection accuracy when the outlying value is 200% of true mean and the standard deviation of the normal distribution for all three attributes is 0.1% of true mean, while the percentage of outliers varies from 0 to 40%.

Figure 6.27 illustrates the detection accuracy when the outlying value is 200% of true mean and the standard deviation of the normal distribution for all three attributes is 1% of true mean, while the percentage of outliers varies from 0 to 40%.

ROBUST MEAN ACCURACY

We now evaluate the accuracy of the robust mean, which is to be used as the aggregation result. First we observe the accuracy when no outliers are present in the cluster. Then we evaluate the worst case scenario with 20 and 40% outliers and examine just how much they can disturb the aggregation result while having less than 50% chance of being caught.

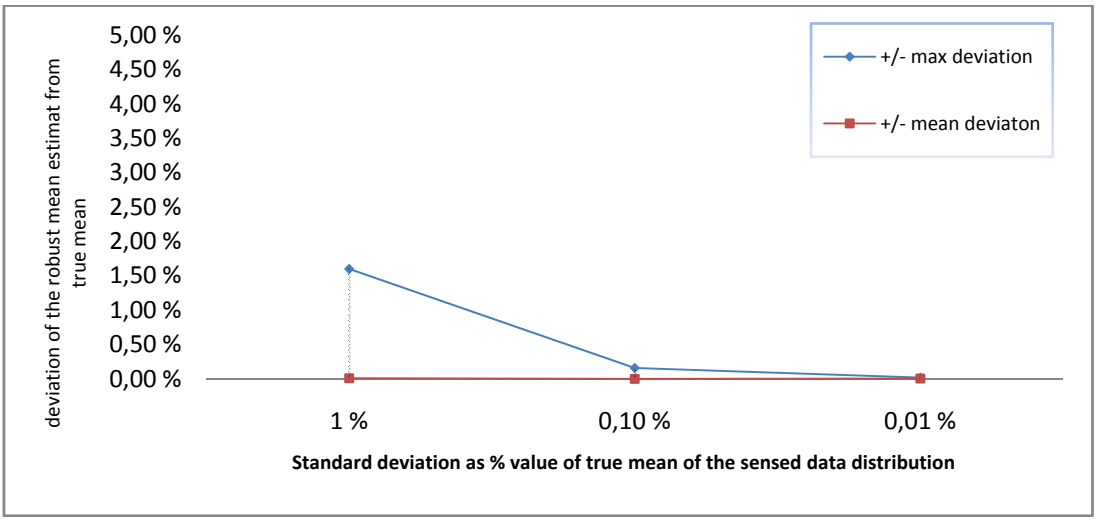


Figure 6.28: Deviation of the robust mean estimate μ' from the calculated mean μ

As we observed in the fourth scenario, the accuracy of the robust mean increase as the accuracy of the sensors increase, and as we observed above (part 1), the maximal deviation an outlier can have from the sensed data, while having a <50% chance of being caught, is limited by the sensor accuracy.

The results from the simulations in part 1, with the standard deviation σ = 1, 0.1 and 0.01% of the true mean, give us the max outlying deviation, with the outlier having less than 50% probability of being cought, of 5, 0.5% and 0.05% with 20% outlying probability, and 40%, 4% and 0,4% with 40% outlier probability.

We present the result of the evaluation in table 6.1 below.

20% OUTLIERS

σ =1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (100,100,100)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,10000,10500)$ and $\sigma_i = (100,100,0.0001)$, for 5% outlying value.

σ =0.1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (10,10,10)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,10000,10050)$ and $\sigma_i = (10,10,0.0001)$, for 0.5% outlying value.

σ =0.01% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (1,1,1)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,10000,10005)$ and $\sigma_i = (1,1,0.0001)$, for 0.05% outlying value.

40% OUTLIERS

σ =1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (100,100,100)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,10000,14000)$ and $\sigma_i = (100,100,0.0001)$, for 40% outlying value.

σ =0.1% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (10,10,10)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,10000,10400)$ and $\sigma_i = (10,10,0.0001)$, for 4% outlying value.

σ =0.01% of the true mean

- For normal nodes, the attributes are generated with $\mu_i = (10000,10000,10000)$ and $\sigma_i = (1,1,1)$.
- For outliers, the attribute values are generated with
  - $\mu_i = (10000,10000,10040)$ and $\sigma_i = (1,1,0.0001)$, for 0.4% outlying value.

Deviation 0 % outliers

Deviation	1 %	0,10 %	0,01 %
Max +/-	1,60 %	0,16 %	0,02 %
Mean +/-	0,01 %	0,00 %	0,01 %

Deviation, 50% detection accuracy, 20% outliers

	1%	0,1%	0,01 %
Max +/-	4,10 %	0,34 %	0,04 %
Mean +/-	0,50 %	0,05 %	0,01 %

Deviation, 50% detection accuracy, 40% outliers

	1%	0,1%	0,01 %
Max +/-	29,70 %	3,89 %	0,33 %
Mean +/-	4,70 %	0,47 %	0,05 %

Table 6.1: deviation of the robust mean when outliers try to influence the estimation

(The Max +/- row gives the maximal observed deviation values, while the Mean +/- row gives us the mean of the deviations over the 1000 iterations the simulation has run)

Table 6.1 contains the results of the max and mean deviation of the robust mean from the true mean when the presence of outliers varies from 0 to 40%, and the probability of the outliers being detected is < 50%

SIXTH SCENARIO

In this scenario we evaluate the modified false information filtering protocol and evaluate how our proposal of using the MAD as the sample standard deviation σ compares with the original proposal of [Liu et al. 2007] in detecting outliers and generating false alarms. We consider the percentage of outlying nodes that is flagged as untrustworthy relay nodes.

The number of nodes in the 1-hop neighborhood $N_1(x)$ is set to 10, and the percentage of outliers varies from 0 to 40%.

- For normal nodes, the attributes are generated with $\mu_i = (10,15,20)$ and $\sigma_i = (1,1,1)$.
- For outliers in one attribute, the attribute values are generated with $\mu_i = (10,15,40)$ and $\sigma_i = (1,1,1)$,
- For outliers with three outlying attributes, the attribute values are generated with $\mu_i = (30,35,40)$ and $\sigma_i = (1,1,1)$.

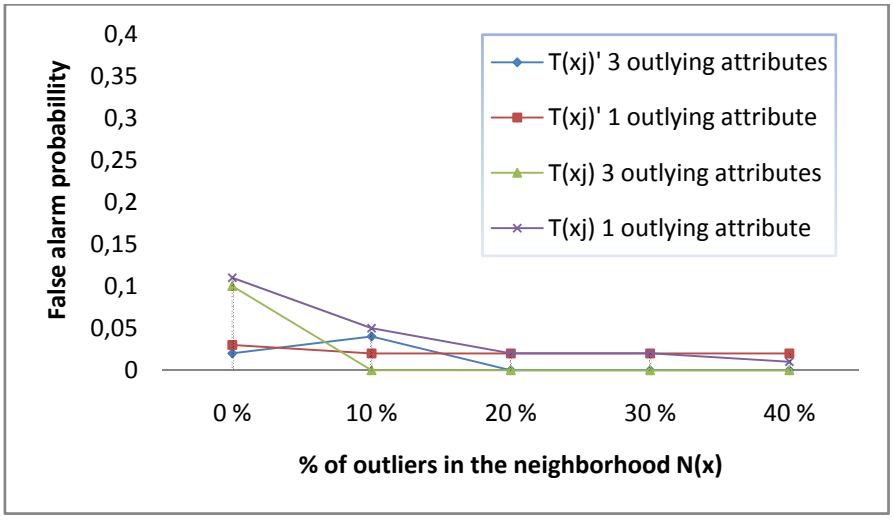


Figure 6.29: false alarm probability in thrust based filtering

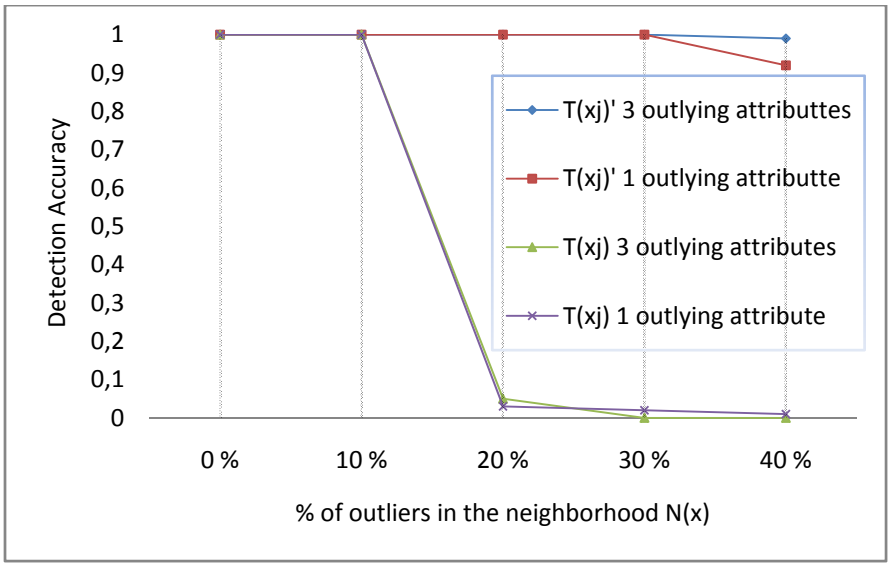


Figure 6.30: detection accuracy of outliers in thrust based filtering

Figure 6.29 Illustrates the false alarm probability of the trust based information filtering protocol with T(xj)' denoting our modified version and T(xj) denoting the original proposal by [Liu et al. 2007].

Figure 6.29 Illustrates the detection accuracy of the trust based information filtering protocol with T(xj)' denoting our modified version and T(xj) denoting the original proposal by [Liu et al. 2007].

6.1 Discussion of simulation results

In the first scenario with 20 nodes.

As illustrated in figures 6.1 and 6.4, clearly the algorithm can detect outliers with a high detection accuracy and a low false alarm probability. This is consistent with the findings in the simulation study of [Liu et al. 2007].

We can also note that the dynamic threshold Θ' is slightly more accurate than the static chi-squared threshold Θ in false alarm probability, and we notice that the increase of the percentage of outlying nodes does not lead to an increase in the false alarm probability, this property is the result of the robust statistics employed in the calculations of the Mahalanobis squared distances [Liu et al. 2007].

However, the static chi-squared threshold Θ is slightly more accurate in detecting outliers when the outlying probability becomes large, since it is not influenced by the values of the Mahalanobis squared distances, which will be more and more influenced by outlying values as the number of outliers increase.

We also observe that for outliers in one attribute, both thresholds has similar accuracy, however for outliers in 3 attributes, the detection accuracy of the proposed dynamic chi-squared threshold Θ' decreases significantly with over 40% outliers, while the static chi-squared threshold Θ remains fairly accurate, even with 45% outliers in the neighborhood.

All in all, we notice that our proposal for the dynamic threshold Θ' holds some attractive properties, like low false detection probability with 0 outliers present in the neighborhood, which will normally be the case, and fairly high detection accuracy, even with 25% outliers in the neighborhood.

In the second scenario with 10 nodes.

As illustrated in figures 6.5 and 6.6, the algorithm can detect outliers with a high detection accuracy and a low false alarm probability even with a sparse neighborhood consisting of only 10 nodes. However we observe an increase in the false alarm probability from the previous scenario with 20 nodes, when 0 outliers exist in the neighborhood. This is expected

because the dataset $F_1(x_i)$ is smaller and less information is used to estimate the sample means and sample covariance matrix.

We notice that the dynamic threshold Θ' has a slightly lesser false alarm probability than the static chi-squared threshold Θ, and we observe that the increase of the percentage of outlying nodes does not lead to an increase in the false alarm probability. This is consistent with the findings in the first scenario.

We observe in this scenario, with 10 nodes, that the accuracy of detection algorithm to detect outliers decrease dramatically when the number of outliers exceeds 20% and the outliers are outliers in only 1 attribute. This is consistent with the findings in the first scenario and the simulation study of [Liu et al. 2007].

In the third scenario with 5 nodes.

As illustrated in figures 6.7 and 6.8, the algorithm can detect outliers with a fairly high detection accuracy and a low false alarm probability even with a sparse neighborhood consisting of only 5 nodes.

We notice that our proposed dynamic threshold Θ', in this scenario as well, has a lower false alarm probability than the static chi-squared threshold Θ.

However, we observe a decrease in the detection accuracy from the previous scenario with 10 nodes, as the percentage of outliers in only 1 attribute increases. However, this is to be expected as a result of the increase of the chi-squared percentile to $X_3^2(0{,}9999)$ in the determination of the thresholds Θ' and Θ.

Since the same nodes already have been evaluated by the detection algorithm in the previous scenario regarding $N_1(x)$, which doesn't suffer from this increase of the threshold, this decrease in detection accuracy should have little impact on the total performance of the algorithm.

In the fourth scenario with 5 nodes.

An important part of our proposed intrusion detection scheme, is the generation and protection of the aggregated data of the cluster head. First we evaluate how well the robust estimates perform in a cluster of 5 nodes and no outliers present.

Figure 6.9 illustrates how much the robust mean estimate deviates from the true mean of the sensed data. This is important to observe because it illustrates how accurate the aggregated result will be as a result of the accuracy of the sensors. More accurate sensors will have less variance in the sensed data, when the sensors are sensing the same physical phenomenon, and as figure 6.9 illustrates, the more accurate the sensors are, the more accurate the aggregation result will become. This observation tells us that we need to be aware of how much the sensor readings might fluctuate between the nodes in a cluster, as well as the demanded accuracy of the sensor data from each cluster.

In the fifth scenario with 5 nodes.

From our observations in part 1, it would appear that an outlier can have an outlying attribute value of approximately $5 * \sigma$ with 20% outliers and approximately $40 * \sigma$ with 40% outliers, while having less than 50% chance of being caught. This indicates that an outlying node is limited in how much it can deviate in its behavior before the algorithm deems it an outlier.

We also notice that the proposed dynamic threshold Θ' has a slightly less detection accuracy compared the static threshold Θ, but not drastically so.

In part 2 we are interested in examining the detection accuracy and false alarm probability of the algorithm when outliers deviate beyond the values in part 1. In the observations the outliers have an outlying value of 200% the true mean of the normal attribute distribution.

We observe in Figs. 6.22, 6.24 and 6.26, that the false alarm probability is consistent with the previous results from the third scenario, which implies that, the variance of the normal attribute distribution has no effect on the false alarm probability.

In Figs. 6.23, 6.25 and 6.26 however, we observe that the detection accuracy is greatly influenced by the variance of the normal attribute distribution. This is of course a result of the outliers distance to the center of the multivariate distribution becoming larger when the multivariate standard deviation decreases, and the outlying deviation remains the same, in this case 200% of the true mean.

As a result of this, an attacker should become increasingly limited in his influence on the aggregated data result, as the accuracy of the sensors increase. But just how much can a smart attacker influence the aggregation result. From table 6.1 we can observe the max deviation of

the robust mean estimate an outlier can influence. We observe that the more accurate the sensors are ($\sigma -> 0\%$) the lesser the outliers can influence the estimate. This nice property of the algorithm can be very useful when you are able to estimate that the worst case scenario can cause only so much deviation on the aggregated result.

Sixth scenario

In this scenario we evaluate the modified false information filtering protocol and evaluate how our proposal of using the MAD as the sample standard deviation σ compares with the original proposal of [Liu et al. 2007] in detecting outliers and generating false alarms.

We observe in Figs. 6.29 and 6.30 that our proposed trust value function T(xj)', using MAD as the sample standard deviation σ, performs quite nicely in detecting outliers as well as having acceptable low false alarm probability. This is a result of using MAD as the sample standard deviation σ, due to outliers having little effect on this method until they number ≥ 50% of the nodes in the neighborhood. We also observe that the original proposal of the trust value function T(xj) [Liu et al. 2007] has acceptable low false alarm probability, but has a breakdown point of 20% in the detection accuracy.

All in all, our proposed algorithm has a quite acceptable theoretical performance. We have observed that it has good detection accuracy with as much as 25% outliers present, while having low false alarm probability. We have observed that it limits how much an adversary can manipulate the aggregation result, depending on the variance of the normal sensor readings. We have observed that the detection accuracy increases with the number of outlying attributes for each outlier and we have observed that the proposal of using the robust mean of the sensor data distribution could be used as the aggregation result of the cluster, depending on the demands for accuracy in the sensor readings and aggregation result of course.

7. CONCLUSION

Wireless sensor network (WSN) is an emerging important research area, and the variety in and number of applications is growing in wireless sensor networks. These wireless sensor nodes are tiny devices with limited energy, memory, transmission range, and computational power. Currently most research in wireless sensor networks have focused on routing protocols, data aggregation and clustering protocols.

Because WSNs in general and in nature are unattended and physically reachable from the outside world, they could be vulnerable to physical attacks in the form of node capture or node destruction (DoS). These forms of attacks are hard to protect against and require intelligent prevention methods. It is necessary for WSNs to have security measures in place as to prevent an intruder from inserting compromised nodes in order to decimate or disturb the network performance.

Data aggregation is a fundamental part of energy saving in wireless sensor networks. However, this is a potential security risk as it introduces a single point of failure. It is important to protect the network from these types of vulnerabilities by designing security mechanisms to counter the threat of aggregation node compromise and compromised nodes interfering with aggregation results.

In this thesis we have presented a novel intrusion detection framework for wireless sensor networks to act as second line of defense in conjunction with the preventive measures, such as encryption and authentication. We have taken a more practical approach in our assumptions and tried to develop a framework for small home or office sensor networks with few nodes.

Our IDS framework does not require prior knowledge of network behavior or a learning period in order to establish this knowledge. The proposed framework is also dynamic in nature as to cope with new and unknown attack types. By exploiting the similarity in behavior of nodes in proximity of each other, our IDS framework can achieve a high detection accuracy and low false alarm probability as indicated by the theoretical simulation study in section 6, and by using the OGK robust estimate of the mean for the sensor readings distribution within the clusters, our framework can achieve high accuracy of aggregation results, even with smart attackers trying to influence the aggregation result by seeding the aggregator node with slightly outlying sensor readings.

A nice property of the framework is that it can be “tailored” to fit various scenarios as a result of the possibility of monitoring different aspects of networking behavior simultaneously. Another nice property of our IDS framework is that it has low memory usage, as it doesn’t require a specification file for normal behavior or a pattern recognition file for known attacks.

Due to time constraints, we have not been able to properly simulate our intrusion detection framework, or implement it in real life situations. This should be a target for future work regarding this framework. Another possible future research topic could be “tailoring” the framework to detect certain attacks by choosing which behavior attributes to monitor. Further research is needed in determining the minimum trust value in the modified trust-based information filtering protocol, as it has been determined purely by theoretical simulation in this thesis.

REFERENCES:

[Liu et al. 2007]"Insider Attacker Detection in Wirelss Sensor Networks" IEEE INFOCOM 2007 proceedings. F.Liu, X.Cheng and D.Chen

[Filzmoser] "A MULTIVARIATE OUTLIER DETECTION METHOD" P. Filzmoser

[Maronna, et al 2006] "*Robust Statistics: Theory and Methods"*, Wiley Publisher, 2006. R. A. Maronna, R. D. Martin and V. J. Yohai chapter 6.9.1 pp.205-208.

*[Hussain, et al. 2008] "PERFORMANCE EVALUATION BASED ON THE ROBUST MAHALANOBIS DISTANCE AND MULTILEVEL MODELLING USING TWO NEW STRATEGIES" S. Hussain, M. A. Mohamed, R. Holder, A. Almasri and G. Shukur February 2008

[Alqallaf, et al. 2002] "Scalable robust covariance and correlation estimates for data mining", ACM SIGKDD 2002, pp.14-23, Edmonton, Alberta, Canada. F. A. Alqallaf, K. P. Konis, R. Douglas Martin and Ruben H. Zamar,

*[Lancaster 1969] "The chi-squared distribution" , Wiley (1969) H.O. Lancaster.

[Ngai. 2005]"Intrusion Detection for Wireless Sensor Networks". Ph.D. -- Term 2 Paper, Edith C.H. Ngai 2005

[Sarma et al. 2006] "Security Threats in Wireless Sensor Networks" Hiren Kumar Deva Sarma, Sikkim Manipal, Avijit Kar

[Newsome, et al. 2004] "The Sybil Attack in Sensor Networks: Analysis & Defenses" James Newsome, Elaine Shi, Dawn Song and Adrian Perrig

[Chen, et al. 1996] "GRIDS – A Graph Based Intrusion Detection System for Large Networks," S. Staniford-Chen, S. Cheng, R. Crawford, and M. Dilger, The 19th National Information Systems Security Conference, 1996.

[Brutch and Ko. 2003] "Challenges in intrusion detection for wireless sensor networks" in Applications and the Internet Workshops. P. Brutch and C. Ko, 2003. Proceedings. 2003 Symposium on, pp. 368373, 2003

[Alberts and Kuhn. 2002] “Security in ad hoc networks: A general intrusion detection architecture enhancing trust based approaches”. P. Alberts and M. Kuhn. In First International Workshop on wireless Information Systems, 4th International Conference on Enterprise Information Systems, 2002

[Considine et al. 2004]” Approximate aggregation techniques for sensor databases”. In ICDE '04: Proceedings of the 20th International Conference on Data. Engineering, pages 449.460, 2004. J. Considine, F. Li, G. Kollios, and J. Byers.

[Manjhi, et al. 2005] “Tributaries and deltas: Ef_cient and robust aggregation in sensor network streams”. In SIGMOD '05: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pages 287.298, 2005. A. Manjhi, S. Nath, and P. B. Gibbons.

[Cormode, et al. 2005] “Holistic aggregates in a networked world: Distributed tracking of approximate quantiles”. In SIGMOD '05: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pages 25.36, 2005. G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi.

[Madden, et al. 2002] ”TAG: A Tiny Aggregation service for ad-hoc sensor networks”. SIGOPS Oper. Syst. Rev., 36(SI):131.146, 2002. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong.

[Shrivastava, et al. 2004] “Medians and beyond: New aggregation techniques for sensor networks”. In SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pages 239.249, 2004. N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri.

[Skraba, et al. 2006] ”Sweeps over wireless sensor networks”. In IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks, pages 143.151, 2006. P. Skraba, Q. Fang, A. Nguyen, and L. Guibas.

[Hu and Evans. 2003] "Secure aggregation for wireless networks," 2003. L. Hu and D. Evans,

[Maronna & Zamar 2002] “Robust estimates of location and dispersion for high-dimensional datasets” Technometrics Nov , 2002. Ricardo A. MARONNA Ruben H. ZAMAR

[Chen, et al. 2007]"Key Feature and Rule-based Intrusion Detection for Wireless Sensor Networks" 2007 IFIP International Conference on Network and Parallel Computing - Workshops. Haiguang Chen, Huafeng Wu, Xi Zhou and Chuanshan Gao.

[Rajasegarar, et al. 2006]"DISTRIBUTED ANOMALY DETECTION IN WIRELESS SENSOR NETWORKS" Sutharshan Rajasegarar, Christopher Leckie, Marimuthu Palaniswami and James C. Bezdek

[Draves, et al. 2004]"Comparison of Routing Metrics for Static Multi-Hop Wireless Networks" SIGCOMM’04, Aug. 30–Sept. 3, 2004. Richard Draves, Jitendra Padhye and Brian Zill

[Denning. 1987] “An Intrusion-Detection Model,” D. E. Denning, IEEE Transactions on Software Engineering, vol.SE-13, no. 2, February 1987, pp. 222-232.

[Anatvalee and Wu. 2007] “A Survey on Intrusion Detection in Mobile Ad Hoc Networks” Signals and Commmunication Technologies, ISSN 1860-4862 ,Wireless Network Security, Springer US 2007, pp. 159-180. Tiranuch Anatvalee and Jie Wu

APENDIX A

Mathematica code for calculation simulations of outlier detection.

```
Needs["MultivariateStatistics`"];
fpc={};
fpv={};
ndo1={};
ndo2={};
vnrm1={};
vnrm2={};
vnrm3={};
vonm1={};
vonm2={};
vonm3={};
def={};
kk=0;
Clear[X,c1,c2,R,G,P,u,u0,m,p,n,q,W,v,v2,k,Q,Qt,V,H,Δ,Λ,aa,bb,cc,dd,ff,c2,bb
b,aa,bb,cc,dd,Xt,qq,A,n1,n2,ch1,x1,x2,x3,x4,x5,def,rr,oo,pp,α];
n1=3;               "number of normal nodes";
n2=2;               "number of insider attackers";
n=n1+n2;     "Number of nodes in the neighborhood";
q=3;                 "Number of variables measured";
c1=4.5;
c2=3;
α=0.9999;

aa={100,150,200};"standard deviation normal nodes";
bb={{1,0,0},{0,1,0},{0,0,1}};"corelation matrix p normal nodes";
cc=Table[Extract[aa,i]*Extract[aa,j]*Extract[bb,{i,j}],{i,q},{j,q}];"covari
ance matrix Σ for normal nodes";
dd={10000,15000,20000};"μ mean values for normal nodes";

oo={100,150,300};"values of σ (standard deviation) insider attackers";
pp={{1,0,0},{0,1,0},{0,0,1}};"corelation matrix insider attackers";
rr=Table[Extract[oo,i]*Extract[oo,j]*Extract[pp,{i,j}],{i,q},{j,q}];."creat
e the covariance matrix Σ insider attackers";
ss={10000,15000,60000};"values of μ (mean) for the insider attackers";

"Defining the functions used in the calculations by the algoritm";
"σ0"; MAD[x_]:=Median[Abs[x-Median[x]]];

"μ0"; u0[x_]:=Median[x];

W[x_,i_]:=If[Abs[x]≤c1,(1-(x/c1)^2)^2,0]; "Weighting function";

p[x_]:=Min[x^2,(c2)^2];
"p function";

v[x_,i_]:=(Extract[x,i]-u0[x])/MAD[x];"this is what the weight function
takes as input";

u[x_]:=∑_{i=1}^{n} (Extract[x, i] * W[v[x, i], i]) ∑_{i=1}^{n} W[v[x, i], i];

m[x_]:=If[MAD[x]≠0,MAD[x]^2/n*∑_{i=1}^{n} p[v2[x, i]]
                                ,1];

v2[x_,i_]:=(Extract[x,i]-u[x])/MAD[x];"This is what the p function takes as
input";


"this is where the calculations begin"
```

```
kkk=Input["enter number of iterations"];
While[kkk≠0,
 If[!IntegerQ[kkk]||kkk≤0,Break[]];

Clear[X,V,X1,X2,Xa,Xb,Xc,H,Δ,Γ,A,i,Xt,R,G,P,k,Q,Qt,V,H,Δ,Λ,ff,bbb,ccc,Xt,qq
,A,ch1,x1,x2,x3,x4,x5,x6,x7,sp,sq];

X1=RandomReal[MultinormalDistribution[dd,{cc[[1]],cc[[2]],cc[[3]]}],n1];"He
re we generate the values for the normal nodes";

X2=RandomReal[MultinormalDistribution[ss,{rr[[1]],rr[[2]],rr[[3]]}],n2];."G
enerating the data for the insider attackers";
 X={};"putting the normal nodes and the insider attackers in the same
matrix of observation data for neighborhood";
 For[i=1,i≤n1,i++,AppendTo[X,X1[[i]]]];
 For[i=1,i≤n2,i++,AppendTo[X,X2[[i]]]];
 Xt=Transpose[X];

 "Step 1";
 bbb={};
 ccc={};
 For[i=1,i≤q,i++,AppendTo[bbb,(m[Xt[[i]]]^(1/2))]];
 P=DiagonalMatrix[bbb];
 Pin=Inverse[P];
 G=X.Pin;
 Gt=Transpose[G];

 "Step 2";
 qq={};

For[i=1,i≤q,i++,For[j=1,j≤q,j++,AppendTo[qq,If[i==j,1,1/4*(m[(Gt[[i]]+Gt[[j
]])]-m[(Gt[[i]]-Gt[[j]])])]]]];
 R=Table[Take[qq,{q*i-(q-1),q*i}] ,{i,q}];

 "Step 3";
 {k,Qt}=Eigensystem[R];
 Λ=DiagonalMatrix[k];
 Q=Transpose[Qt];

 "Step 4";
 H={};
 For[i=1,i≤n,i++,AppendTo[H,Qt.G[[i]]]];
 Δ={};
 For[i=1,i≤q,i++,AppendTo[Δ,u[H[[All,i]]]]];
 A={};
 For[i=1,i≤q,i++,AppendTo[A,m[H[[All,i]]]]];
 Γ=DiagonalMatrix[A];

 "Step 5";
 V=P.Q;
 μ̂=V.Δ;
 Σ̂=V.Γ.Transpose[V];
 Xa={};
 Xb={};
 Xc={};
 sp={};
 sq={};
 For[i=1,i≤q,i++,AppendTo[Xa,Mean[X1[[All,i]]]]];
 For[i=1,i≤q,i++,AppendTo[Xc,Mean[X[[All,i]]]]];

 For[i=1,i≤q,i++,AppendTo[sp,(((Extract[μ̂,i]/Extract[Xa,i])*100)-100)]];
 For[i=1,i≤q,i++,AppendTo[sq,(((Extract[Xc,i])/Extract[Xa,i])*100)-100]];
 For[i=1,i≤q,i++,AppendTo[Xb,Variance[X[[All,i]]]]];
```

```
 "Calculating the squared Mahalanobis distances";
 ff={};
 MD3[x_]:=(x-μ̂).PseudoInverse[Σ̂].(x-μ̂);
 For[i=1,i≤n,i++,AppendTo[ff,MD3[X[[i]]]]];

 "Determinig the thresholds";
 x1={};
 x2={};
 x3={};
 x4={};
 x5={};
 "Regular chi-squared threshold";
 ch=Quantile[ChiSquareDistribution[q],α];
 "Variating chi-squared threshold";
 ch1=(ch*Median[ff])/Quantile[ChiSquareDistribution[q],0.5];

 For[i=1,i≤n1,i++,If[Extract[ff,i]≥ch,AppendTo[x1,1]]];
 For[i=1,i≤n1,i++,If[Extract[ff,i]≥ch1,AppendTo[x2,1]]];
 x6={};
 x7={};
 For[i=(n1+1),i≤n,i++,If[Extract[ff,i]≤ch,AppendTo[x6,1]]];
 For[i=(n1+1),i≤n,i++,If[Extract[ff,i]≤ch1,AppendTo[x7,1]]];

 def={};
 For[i=1,i≤q,i++,AppendTo[x4,StandardDeviation[X1[[All,i]]]]];
 For[i=1,i≤q,i++,AppendTo[def,Abs[(100-
(((Extract[bbb,i])/Extract[x4,i])*100))]]];
 For[i=1,i≤q,i++,AppendTo[x3,StandardDeviation[X[[All,i]]]]];
 For[i=1,i≤q,i++,AppendTo[x5,(Extract[Σ̂,{i,i}])^(1/2)]];
 AppendTo[fpc,Length[x1]];
 AppendTo[fpv,Length[x2]];
 AppendTo[ndo1,Length[x6]];
 AppendTo[ndo2,Length[x7]];
 AppendTo[vnrm1,Extract[sp,1]];
 AppendTo[vnrm2,Extract[sp,2]];
 AppendTo[vnrm3,Extract[sp,3]];
 kk++;
 kkk--;
 ]
Print[kk,"iterations"]


  Length[fpc]
     ∑      (Extract[fpc, i])
(    i=1                    )."Number of false positives calculated chi-squared"
""
  Length[fpv]
     ∑      (Extract[fpv, i])
(    i=1                    )."Number of false positives variable chi-squared"
""
""

  Length[ndo1]
     ∑      (Extract[ndo1, i])
(    i=1                     )."Number of not detected outliers calculated chi-
squared"
""
  Length[ndo2]
     ∑      (Extract[ndo2, i])
(    i=1                     )."Number of not detected outliers variable chi-
squared"
""
""

Min[vnrm1]."Min variation normal vs. robust μ1"
Min[vnrm2]."Min variation normal vs. robust μ2"
```

```
Min[vnrm3]."Min variation normal vs. robust μ3"
""
Max[vnrm1]."Max variation normal vs. robust μ1"
Max[vnrm2]."Max variation normal vs. robust μ2"
Max[vnrm3]."Max variation normal vs. robust μ3"
""
Mean[vnrm1]."Mean variation normal vs. robust μ1"
Mean[vnrm2]."Mean variation normal vs. robust μ2"
Mean[vnrm3]."Mean variation normal vs. robust μ3"
```

APENDIX B

Mathematica code for the trust-based information filtering protocol

```
aac={};
aad={};
aaa={};
aab={};
kk=0;
```

Trust - based information filtering

```
Needs["MultivariateStatistics`"];
Clear[F,a,•,•,n1,a,b,c,d,n,n1,n2,q,aa,bb,cc,dd,oo,pp,rr,ss,F11,F22];
n1=16;                "Number of normal nodes in the neighborhood";
n2=4;                "Number of insider attacker nodes in the neighborhood";
n=n1+n2;     "Number of nodes in the neighborhood";
q=3;                 "Number of variables measured";
b={};

c={};
aa={1,1,1};"standard deviation normal nodes";
bb={{1,0,0},{0,1,0},{0,0,1}};"corelation matrix p normal nodes";
cc=Table[Extract[aa,i]*Extract[aa,j]*Extract[bb,{i,j}],{i,q},{j,q}];"covari
ance matrix • for normal nodes";
dd={10,15,20};"• mean values for normal nodes";
F11=RandomReal[MultinormalDistribution[dd,{cc[[1]],cc[[2]],cc[[3]]}],n1];"H
ere we generate the values for the normal nodes";

oo={1,1,1};"values of • (standard deviation) insider attackers";
pp={{1,0,0},{0,1,0},{0,0,1}};"corelation matrix insider attackers";
rr=Table[Extract[oo,i]*Extract[oo,j]*Extract[pp,{i,j}],{i,q},{j,q}];."creat
e the covariance matrix • insider attackers";
ss={30,35,40};"values of • (mean) for the insider attackers";
F22=RandomReal[MultinormalDistribution[ss,{rr[[1]],rr[[2]],rr[[3]]}],n2];."
Generating the data for the insider attackers";

F={};"putting the normal nodes and the insider attackers in the same matrix
of observation data for neighborhood";
For[i=1,i•n1,i++,AppendTo[F,F11[[i]]]];
For[i=1,i•n2,i++,AppendTo[F,F22[[i]]]];
F//MatrixForm
•[x_]:=Median[x];
•[x_,j_]:= \sqrt{\frac{1}{n1-1} * \sum_{i=1}^{n} (Extract[x, i] - Extract[b, j])^2} ;

MAD[x_]:=If[Median[Abs[x-Median[x]]]•0,Median[Abs[x-Median[x]]],Median[x]];
k={};
For[i=1,i•q,i++,AppendTo[b,•[F[[All,i]]]]];
For[i=1,i•q,i++,AppendTo[c,MAD[F[[All,i]]]]];
For[i=1,i•q,i++,AppendTo[k,•[F[[All,i]],i]]];
b//MatrixForm
c//MatrixForm
k//MatrixForm
a={};
For[j=1,j•n,j++,For[i=1,i•q,i++,AppendTo[a,If[Abs[(Extract[F[[j]],i]-
Extract[b,i])/Extract[c,i]]==0,0.1,Abs[(Extract[F[[j]],i]-
Extract[b,i])/Extract[c,i]]]]]];
l={};

For[j=1,j•n,j++,For[i=1,i•q,i++,AppendTo[l,If[Abs[(Extract[F[[j]],i]-
Extract[b,i])/Extract[k,i]]==0,0.1,Abs[(Extract[F[[j]],i]-
Extract[b,i])/Extract[k,i]]]]]];
d=Table[Take[a,{q*i-2,q*i}] ,{i,n}];
d//MatrixForm
o=Table[Take[l,{q*i-2,q*i}] ,{i,n}];
o//MatrixForm
```

```
g={};
For[i=1,i•n,i++,AppendTo[g,Max[d[[i]]]]];
g//MatrixForm
m={};
For[i=1,i•n,i++,AppendTo[m,Max[o[[i]]]]];
Clear[h,Mm,p];

h={};
t={};
u={};
Mm=Min[g];
For[i=1,i•n,i++,AppendTo[h,If[Extract[g,i]•0,Mm/(Extract[g,i]),(Mm/0.1)]]];
For[i=1,i•n1,i++,AppendTo[t,If[Extract[g,i]•0,Mm/(Extract[g,i]),(Mm/0.1)]]
;
For[i=(n1+1),i•n,i++,AppendTo[u,If[Extract[g,i]•0,Mm/(Extract[g,i]),(Mm/0.1
)]]];
h//MatrixForm
p={};
r={};
s={};
Mm2=Min[m];
For[i=1,i•n,i++,AppendTo[p,Mm2/(Extract[m,i])]];
For[i=1,i•n1,i++,AppendTo[r,Mm2/(Extract[m,i])]];
For[i=(n1+1),i•n,i++,AppendTo[s,Mm2/(Extract[m,i])]];

Tmin=Mm/7
Tmin2=Mm2/2
For[i=1,i•n,i++,If[Extract[h,i]•Tmin,Print[i+"Node is NOT trustworthy with
trustvalue of".(Extract[h,i])],Print[(i)+"Node is trustworthy with trust
value of".(Extract[h,i])]]]

For[i=1,i•n1,i++,If[Extract[h,i]•Tmin,AppendTo[aaa,1]]];
For[i=(n1+1),i•n,i++,If[Extract[h,i]>Tmin,AppendTo[aab,1]]];
""
""
""

For[i=1,i•n,i++,If[Extract[p,i]•Tmin2,Print[i+"Node is NOT trustworthy with
trustvalue of".(Extract[p,i])],Print[(i)+"Node is trustworthy with trust
value of".(Extract[p,i])]]]
For[i=1,i•n1,i++,If[Extract[p,i]•Tmin2,AppendTo[aac,1]]];
For[i=(n1+1),i•n,i++,If[Extract[p,i]>Tmin2,AppendTo[aad,1]]];
 Length[aaa]
     ∑      (Extract[aaa, i])
(    i=1                      )."Number of false positives MAD"
""
 Length[aab]
     ∑      (Extract[aab, i])
(    i=1                      )."Number of not detected outliers MAD"
""
""
""
 Length[aac]
     ∑      (Extract[aac, i])
(    i=1                       )."Number of false positives original"
""
 Length[aad]
     ∑     (Extract[aad, i])
(    i=1                   )."Number of not detected outliers original"
kk++;
kk."iterations"
```