An Efficient Weighted Trust-Based Malicious Node Detection Scheme for Wireless Sensor Networks

Firas Zawaideh

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
May 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Assoc. Prof. Dr. Mohammed Salamah
Supervisor

Examining Committee

1. Prof. Dr. Doğu Arifler
2. Prof. Dr. Işık Aybay
3. Prof. Dr. Semih Bilgen
4. Prof. Dr. Emrullah Turhan Tunalı
5. Assoc. Prof. Dr. Mohammed Salamah

ABSTRACT

The aim of wireless sensor networks (WSNs) is to gather sensor data from a monitored environment. However, the collected or reported information might be falsified by faults or malicious nodes. Hence, identifying malicious nodes in an effective and timely manner is essential for the network to function properly and reliably. Maliciously behaving nodes are usually detected and isolated by reputation and trust-based schemes before they can damage the network. In this thesis, we propose an efficient weighted trust-based malicious node detection (WT-MND) scheme that can detect malicious nodes in a clustered WSN. The node behaviors are realistically treated by accounting for false-positive and false-negative instances. The simulation results confirm the timely identification and isolation of maliciously behaving nodes by the WT-MND scheme. The effectiveness of the proposed scheme is afforded by the adaptive trust-update process, which implicitly performs trust recovery of temporarily malfunctioning nodes and computes a different trust-update factor for each node depending on its behavior. The proposed scheme is more effective and scalable than the related schemes in the literature, as evidenced by its higher detection ratio (DR) and lower misdetection ratio (MDR), which only slightly vary with the network's size. Moreover, the scheme sustains its efficient characteristics without significant power consumption overheads.

Keywords: wireless sensor networks, network security, malicious node, weighted trust, intrusion detection system, LEACH protocol.

ÖZ

Kablosuz sensör ağlarının (WSN'ler) amacı, izlenen bir ortamdan sensör verilerini toplamaktır. Ancak, toplanan veya bildirilen bilgiler, hatalar veya kötü niyetli düğümler tarafından tahrif edilebilir. Bu nedenle, kötü niyetli düğümleri etkili ve zamanında tanımlamak, ağın düzgün ve güvenilir bir şekilde çalışması için çok önemlidir. Kötü niyetli davranan düğümler genellikle ağa zarar vermeden önce itibar ve güven temelli programlar tarafından algılanır ve izole edilir. Bu tezde, kümelenmiş bir WSN'deki kötü amaçlı düğümleri tespit edebilen etkili ağırlıklı güvene dayalı bir kötü amaçlı düğüm algılama (WT-MND) düzeni öneriyoruz. Düğüm davranışları, gerçekçi olarak yanlış pozitif ve yanlış negatif örnekleri hesaba katarak ele alınmıştır. Simülasyon sonuçları, WT-MND düzeni tarafından kötü niyetli davranan düğümlerin zamanında tanımlanmasını ve izole edilmesini onaylar. Bu tezde önerilen düzenin etkinliği, geçici olarak hatalı çalışan düğümlerin güven kurtarma işlemlerini dolaylı olarak yapan ve davranışına bağlı olarak her düğüm için farklı bir güven güncelleme faktörü hesaplayan uyarlanabilir güven güncelleme süreci ile sağlanır. Önerilen düzen, literatürdeki ilgili düzenlerden daha etkili ve ölçeklenebilirdir; bu da ağın boyutuna göre yalnızca biraz farklı olan yüksek algılama oranı (DR) ve düşük yanlış algılama oranı (MDR) ile kanıtlanmaktadır. Ayrıca, önerilen düzen önemli bir güç tüketimi ek masrafı olmadan etkin özelliklerini sürdürmektedir.

Anahtar Kelimeler: kablosuz sensör ağları, ağ güvenliği, kötü niyetli düğüm, ağırlıklı güven, izinsiz giriş tespit sistemi, LEACH protokolü.

DEDICATION

I dedicate this humble work

To my Father and Mother for their endless love, support and encouragement.

To my beloved wife who stood by me during my hard times.

To my Brothers and Sisters for their prayers to me with success

ACKNOWLEDGMENT

Firstly, I would like to extend my deepest gratitude and appreciation to my thesis supervisor, Dr. Muhammed Salamah, who always guided; encouraged and supported me from the very initial stage until the final one. I could not have imagined completing this task without him.

I must express my profound gratitude to my parents, without their inseparable support and prayers, I wouldn't succeed. Firstly My Father, Eng. Hanna Zawaideh, the person whom I had planted fundamentals of my knowledge, who taught me the joy of intellectual pursuit. Secondly, I want to thank my beloved Mother, Lina Haddad, without her love and prayers this journey wouldn't have been possible. From them I learned to withstand, no matter how difficult.

Behind every great man, a woman. A very special thanks for my affectionate wife Lina Ammari. She was always beside me and supporting me. At every moment I was feeling hopeless, she was always planted hope in my myself and directed me to my target.

Finally, I would like to appreciate the encourgement and support of each and everybody who had always been for help in my thesis. I am greatly indebted for their love and guidance.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AH	Ampere-Hour
ARR	Automatic Repeat Request
BS	Base Station
BW	Bandwidth
CH	Cluster Head
CM	Cluster Member
CDMA	Code-Division Multiple Access
DoS	Denial Of Service
DR	Detection Ratio
DT	Direct Trust
DMAC	Dynamic Medium Access Control
FTMNDI	Fair-Trust-Based Malicious Node Detection and Isolation
FEC	Forward Error Correction
FN	Forwarding Node
IT	Indirect Trust
IEEE	Institute of Electrical and Electronics Engineers
IDS	Intrusion Detection System
J	Joule
LEACH	Low Energy Adaptive Clustering Hierarchy
MAC	Medium Access Control
MAT	Minimum Acceptable Trust
MDR	Misdetection Ratio

MANET	Mobile Ad Hoc Network
NCA	Node Clone Attack
PAN	Personal Area Network
QoS	Quality of Service
RT	Response Time
SNs	Sensor Nodes
SD	Standard Deviations
TDMA	Time-Division Multiple Access
TMS	Trust Monitoring System
TUV	Trust Update Vector
TV	Trust Vector
UPS	United Parcel Services
W	Watt
WTA	Weighted Trust Algorithm
WTE	Weighted Trust Evaluation
WTE(R)	Weighted Trust Evaluation (Weight-Recovery)
WT-MND	Weighted Trust-Based Malicious Node Detection
WSNs	Wireless Sensor Networks

Chapter 1

INTRODUCTION

1.1 Introduction

Wireless Sensor Networks (WSNs) are made up of tiny devices called sensor nodes (SNs). SNs are limited power and house integrated sensors which consist of, sensing unit, processing unit, transceiver unit, storage and a power unit, normally distributed randomly in the field. SNs oversee data collection, data aggregation and transmitting the aggregated data to the base station (BS) that performs data fusion. In order to avoid deception through falsified information and corrupted data injected by the attacking node through compromised nodes, these nodes are to be detected and isolated from the WSNs. SNs serve multiple applications in various fields including emergency-response networks, management of energy, medical issues, object tracking, and military management.

Unlike normal wireless networks, matters related to security and performance call for a careful consideration of sensor networks. That is to say, because sensor networks are unguarded, they are vulnerable to various types of attacks which often remain undetected. As a result, SNs should be resistant against attacks, and in case of a successful attack, its impact should be minimized. That is to say, if a single SN or a few SNs are contaminated, this should not lead to an all-out network crash.

Multiple communication models comprising unicast, multicast, and broadcast are supported by SN in WSNs, thus making energy efficiency a major concern. It is worth noting that SNs should be energy efficient as battery lifetime is limited. Therefore, it is of outmost importance that the number of transmited messages and the amount of expensive computation be kept at minimum.

SNs often operate in aggressive and unprotected environments. As a result, they are prone to malicious nodes entering through different kinds of attacks including Denial of Service attacks (DoS), Hello flooding attacks, wormhole attacks, sinkhole attacks, black hole attacks, and Sybil attacks [1-4]. In contrast to traditional wireless networks, WSNs possess limited computation, communication, power, and memory resources. This makes it necessary for WSNs to be protected from malicious attacks by means of lightweight, energy-efficient security schemes with minimum bandwidth (BW) overhead [5].

Reputation and trust-based schemes can be employed to eliminate the aforementioned problems. By description, reputation is the observations accumulated by a node about another node's behavior via direct and indirect inspection of its prior performance, while trust is the subjective anticipation of a node about other nodes future performance regarding a particular behavior [6-9]. Accordingly, trust can be determined through direct or indirect observation of other nodes' behaviors. Direct Trust (DT) and Indirect Trust (IT) are determined in two ways: by direct observations and from the reputations of other nodes.

The main aim of reputation and trust-based mechanisms is to make a distinction between normal and abnormal (malicious) nodes in the network and isolate the

malicious ones before they can damage the network. A trustable node is a node whose trust value equals or exceeds the preset minimum acceptable trust (MAT); if not, it is called a malicious node and will be isolated. Trust-based mechanisms are set to solely accept the trusted nodes as sources of information and data forwarding [10-11].

In this thesis, we propose an efficient weighted-trust-based malicious node detection (WT-MND) scheme that can detect malicious nodes in WSNs. In accordance with our proposed scheme, the SNs transmit their readings to a cluster head (CH) to aggregate the received data. The node behaviors are realistically treated by accounting for both false-positive (FP) and false-negative (FN) instances. The proposed scheme is rendered effective by the adaptive trust-update process that implicitly performs trust recovery and computes a different trust-update factor for every node on the basis of its behavior. Because of the fact that the scheme is adopted and integrated with the Low Energy Adaptive Clustering Hierarchy LEACH protocol, a given node indirectly computes the trust from its reputation among its cluster members (CMs).

In addition, MATLAB simulation evaluates the performance of the new scheme. These simulations help investigate the effects of malicious node probability on the detection ratio (DR), misdetection ratio (MDR), and malicious node lifetime. The study also investigates the effect of node density on the DR and MDR. Moreover, it compares the network lifetimes under the traditional LEACH protocol and the proposed WT-MND in order to examine the power consumption.

1.2 Problem Definition and Motivation

The flexible structure and superior characteristics of WSNs have made them ideal for a diverse assortment of uses including military, industrial, environmental, health, agricultural, civilian, and smart cities. However, the infrastructureless nature of WSNs and the limited capabilities of the SNs (e.g., processing power, memory, and battery lifetime) make them susceptible to various types of attacks by what refer to as malicious nodes.

Furthermore, early literature in the field of WSNs research was mainly interested in problems regarding wireless channel access, multi-hop routing, and power consumption, disregarding the fundamental issues of network security measures [12-14]. Consequently, a trustworthy mechanism for identifying and isolating compromised and malicious nodes to safeguard the network against any harm is deemed indispensable. It is worth noting that traditional security mechanism such as authentication and confidentiality are not suitable for WSNs due to their overreliance on resources, which make them incompatible with the nature of WSNs.

Cluster heads (CHs) are special nodes in a cluster–based routing accumulate data received from SNs in the same cluster and forward it to the BS. Thus, CH needs to evaluate the trust of each of its CMs in order to avoid transmission of false data. Once deployed, clustered sensor networks increase trust, decrease system delay, save the energy during data aggregation operation and hence prolong the system's life.

WSNs endurance correlates positively with their battery efficiency. The long life of battery ensures the long life of WSNs. Many variables including data transmission and data manipulation, how SNs and the CHs are closely or widely spaced, the

transfer packet size, the energy slope of that SN relevant to its physical measuring structure, data processing as well as other factors are also related to the WSNs efficiency.

1.3 Problem Statement

SNs should be expanded and employed in unattended environments. The networks are likely to become more exposed to various types of attacks. The collected data by the SNs in the WSNs might be falsified by faults or malicious nodes. In the network, in case a few SNs are compromised, the whole network can be toppled unless the compromised nodes are immediately isolated from the network. To do so, it is necessary that the compromised nodes be identified by means of suitable algorithm and isolated.

Identifying malicious nodes in an effective and timely manner is essential for the network to function properly and reliably. Maliciously behaving nodes are usually detected and isolated by reputation and trust based schemes before they can do any harm to the network. In this research, an efficient weighted trust-based malicious node detection (WT-MND) scheme that can detect malicious nodes in a clustered WSN was proposed. Moreover, the node behaviors are realistically treated by accounting for false-positive and false-negative instances.

1.4 Research Objective

- This research aims to present a detailed description of an efficient weighted trust-based malicious node detection (WT-MND) scheme proposed to detect the compromised nodes in WSNs.

- The scheme is rendered effective by the adaptive trust-update process that implicitly performs trust recovery and computes a different trust-update factor for every node based on its behavior.

- To study the performance of the scheme proposed using extensive simulation and analysis. These simulations help investigate the effects of malicious node probability on the detection ratio (DR), misdetection ratio (MDR), and malicious node lifetime.

- To also study various parameters used in the scheme and find their optimal values.

- The power consumption is examined by comparing the network lifetimes under the traditional LEACH protocol and the proposed WT-MND scheme.

1.5 Outline of Thesis

The thesis endeavors to show the fundamental requirements of using energy-efficient scheme with a minimum BW overhead to protect WSNs from malicious attacks. Furthermore, the existing literature explaining important theoretical background of the proposed scheme and other schemes related to this research has been mentioned and illustrated. The methodology adopted for this research is explained in detail followed by simulation results and conclusion.

Chapter One explains the fundamentals of WSNs and enumerates some of the most common attacks. In addition, this chapter introduces the study and its objectives,

defines the problem in existing context, and provides the significance of the research. A brief illustration of the proposed WT-MND scheme is also provided.

Chapter Two summarizes the literature that supports the hypothesis of this thesis. Also, an explanation of the research that has been conducted so far related to WSNs, intrusion detection system (IDS), trust-based and reputation systems, security issues and conceptual frameworks are provided in detail.

The methodology of the thesis involves network architecture, node behaviour modeling, malicious node detection, and the functional block diagram of the WT-MND scheme which are illustrated in details in Chapter Three.

The outcomes of the simulation results confirm the timely identification and maliciously behaving nodes by the WT-MND scheme which are explained in Chapter Four. Adaptive trust-update process affords the effectiveness of the proposed scheme, which implicitly triggers trust recovery of temporarily malfunctioning nodes and computes a different trust-update factor for each node depending on its behavior.

The final chapter summarizes the results and conclusions of the thesis. It also proposes further extensions and implications for future researches to develop WT-MND scheme.

Chapter 2

THEORETICAL REVIEW

2.1 Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are self-configured and infrastructure-less wireless networks that monitor thermal, auditory, as well as other physical conditions including vibration, pressure, motion or pollutants. Data from multiple WSNs is collectively passed via the network to the BS where it is processed. It is possible to extract desired information from the network by submitting queries and collecting results from the BS. A typical WSN houses hundreds or thousands of SNs. These SNs use radio signals to communicate among themselves. A wireless sensor node has at its disposal various devices used for sensing and computing, radio signals transceiver, and power components. By nature, every node in a WSN is subject to resource limitation. That is to say, they are limited in terms of how fast they process data, how much data they can store, and what size of BW they have at their communication. SNs are autonomous in organizing a suitable network infrastructure using multi-hop communication through WSNs.

2.1.1 Components of a WSN

The architecture diagram in Fig. 1 illustrates the main components of a WSN. A SN typically consists of a sensing, processing, transmission, and power units. Each component is described in detail [4-6].

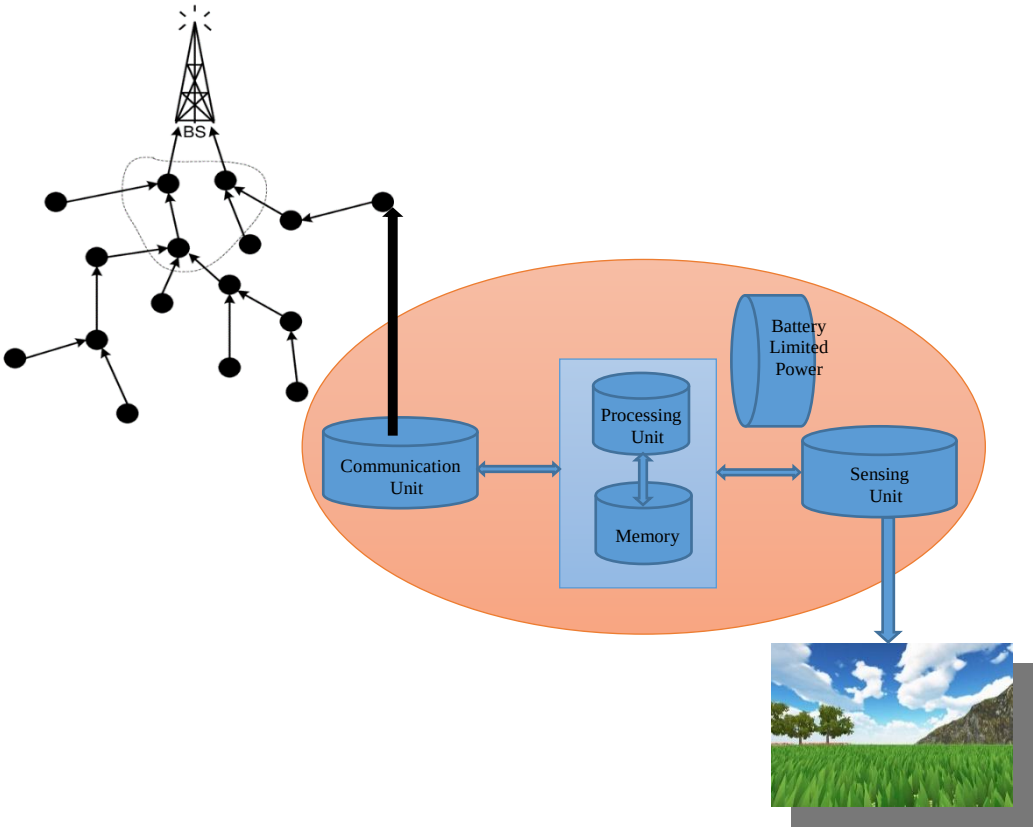

Figure 1: Architecture of Wireless Sensor Nodes

2.1.1.1 The Sensing Unit

A sensing unit is solely designed to convert physical phenomena into electrical signal and comprises two sub units. The first one is sensors and the other is analog to digital conversions [4]. A diverse variety of sensors can be employed to measure the environmental parameters such as temperature, atmospheric pressure, light intensity, sound, images, magnetic field and so on. The sensing unit is made up of a sensor attached with one end of the node whose function is to gather information at the ground level. This unit collects the raw data and transmits it to the processing unit after the data is digitized.

2.1.1.2 The Processing Unit

The SN mainly obtains the intelligence from the processing unit. A small microchip present in the processing unit performs tasks including controlling the sensors,

execution of correspondence conventions and handling the signals algorithms on the assembled sensor data. Overall, four fundamental processor states can be identified in a microprocessor. They are categorized as Off, Sleep, Idle and Active. Most interior peripherals of the CPU are activated in sleep mode and must be initiated by an external event. The CPU remains inactive in idle mode. However, different peripherals are active.

2.1.1.3 Transmission Unit

The operation of transceivers corresponds to that of microcontrollers. They can also be functional in Transmit, Receive and Sleep modes. Among these, sleep mode is an essential feature to save energy. A common observation suggests that the radios operating in idle mode consume immense amounts of energy equivalent to the amount of energy consumed in the receive mode [6]. In addition, the transient activity in the radio electronics accounts for a considerable amount of energy impoverishment when the operating mode of radio is shifted.

2.1.1.4 Battery

The battery performs a fundamental role in determining the life span of a SN by powering the entire SN. Generally, SNs are inexpensive, small, and light as they face a limitation regarding battery size. Therefore, it is important that the power generated by the battery be precisely controlled. Moreover, power consumption is the number one factor in determining the life span of a SN. As it is not possible to replace the battery repeatedly for the networks with thousands of physically embedded nodes that are unreachable, SNs are to have a life span that exceeds several months or even years.

2.1.2 Applications of WSNs

Due to their flexibility, WSNs have managed to become popular in problem solving in different application areas, with the potential to positively influence our lives in a variety of ways. A wide range of applications including military, industrial, environmental, health, agricultural, civilian, and smart cities benefit from WSNs as shown in Fig. 2. Because of their adaptable structure and exceptional characteristics, successful application of WSNs is now commonplace in various domains including the following [15-17]:

- Military applications – WSNs are indispensably utilized in military issues with its communication and computation, intelligence gathering, reconnaissance, targeting systems and battlefield surveillance.

- Area monitoring – SNs are dispatched to reconnoiter a region in which an event is under way. Once the event is picked up by the sensors, a report is transmitted to one of the BSs, triggering an appropriate response.

- Transportation – WSNs' application in real time traffic control is beneficial to feeding transportation models and providing timely warning to drivers about traffic conditions.

- Health applications – SNs are useful in supporting for patient supervision, medication in hospitals, the disabled, medical staff tracking, and monitoring of human physiological data.

- Environmental sensing – WSNs are widely used in Environmental Sensor Networks to contribute to earth science research on oceans, glaciers, volcanoes, forests, and other natural phenomena. Other relevant areas include:
  - Monitoring of air pollution
  - Detection of forest fires
  - Monitoring of greenhouse
  - Detection of landslide
- Structural monitoring – With SNs engineers can remotely perform round-the-clock supervision of activities in various structures including bridges, tunnels, and shoreline installations.

- Industrial monitoring – Another major application of WSNs takes place in industrial settings where they are used for machinery condition-based maintenance (CBM). WSNs can noticeably minimize costs while present new opportunities and applications. WSNs also eliminate the need for costly extensive wiring needed for wired sensors [16].

- Agricultural sector – With WSNs farmers won't need to pay for costly wiring, particularly in rough terrains. In addition, the same system can also be adapted to perform automatic irrigation, hence lowering waste.

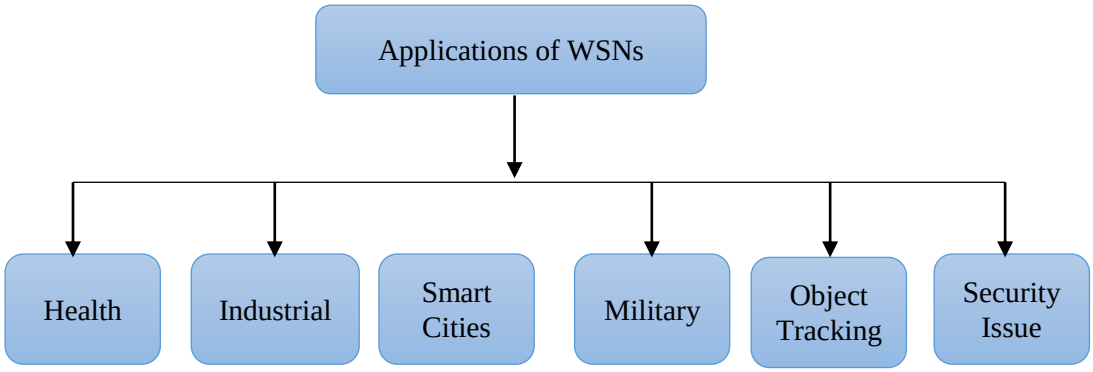

Figure 2: Applications of WSNs

2.2 LEACH

Low Energy Adaptive Clustering Hierarchy (LEACH) protocol clusters all SNs of the WSNs. Every SN cluster has a cluster head (CH) as shown by Heinzelman [27]. The sensing data is transmitted from each SN to the CH. The data collected from all cluster members (CMs) is compressed and aggregated by the CHs before being transmitted to the BS [2].

The CH has the task of collecting, aggregating, and forwarding data from all nodes as well as receiving data from BS, hence leading to more power consumption as compared to other SNs. This implies providing more energy to the CH compared to other SNs. The LEACH protocol is set to rotate the node that is selected as CH after a specific number of rounds. As the selection of CH is performed randomly, it may not be the node with maximum energy [16].

Heinzelman simulation results have demonstrated that the nodes categorized as a CH constitute less than 5% of the total number of the SNs in the WSNs. To ensure the power consumption and connectivity are kept at minimum, inter or intra cluster collisions such as DMAC, LEACH utilize a specified MAC protocol. In addition,

this algorithm presupposes the CH as centralized or semi-centralized nodes in every cluster. A sample hierarchically clustered network is shown in Fig. 5 [17].

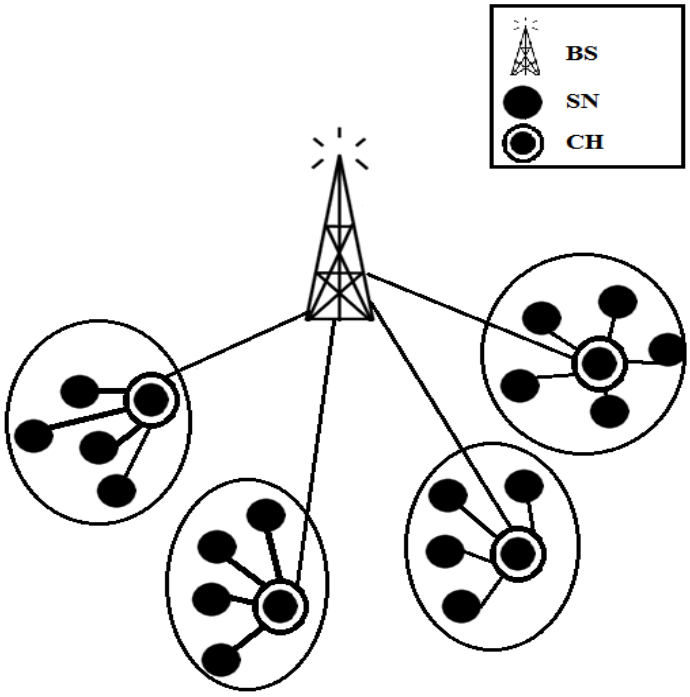

Figure 3: Sample Hierarchically Clustered Wireless Network

Set-up state and steady-state are the two phases in which LEACH protocol. In the first state, the SNs are being clustered into different clusters then for each cluster a CH will assigned. The predefined nodes are chosen as CHs in line with a to a threshold value, deepening of the percentage of this value makes the node to be considered as a CH. Each node choses a value between 0 and 1 to become a CH. Each new CH will notify all CMs to deal with it as a CH. Once the CMs are notified, a message will be submitted from the CMs [17].

Regardless of the BS, the CH can be located anywhere in the network and can be changed randomly. The network security is a major factor in improvement of the networks as the WSNs are based on negligible memory sensors and low

computational power [18]. A number of real-time problems regarding the system might be attributed to the LEACH protocol. The following are some of the main assumptions [20]:

- Transmission of all the nodes to the BS is with sufficient power as per its need.

- Each SN should have enough computational power to provide different MAC protocols.

- There is always data at nodes that awaits transmission.

- Data correlation is present between the nodes that are situated near to each other.

- The system becomes unbalanced with the death of the first node.

2.3 Misbehavior of Nodes

The lack of organizational environment and infrastructure in WSNs exposes the networks to various types of attacks. Unless proper countermeasures are defined and put in place, malicious behavior can impact the network by enforcing better service than those of cooperating nodes. It can also exploit incentive measures for financial gains or even trade sensitive information. Besides, it can save power through selfish behavior, filter other users' access to proper service, and extract data to obtain confidential information, etc [28].

The misbehavior is in the case of a mal-functioning node. Nevertheless, even in the absence of any pre-mediated malpractice, it can adversely influence the performance of a network. Different types of security threats caused by malicious nodes were identified within a WSN [12-14], which required a number of supportive measures to

counteract them. It is worth noting that the system is capable of withstanding the misbehavior of a single node if it is manifested as selfishness due to the predictable nature of this misbehavior.

The behavior of a selfish node will always lead to maximizing its advantages, and motivation can be used to guarantee that collaboration is consistently the most productive option. Yet, it is very difficult for the system network to overcome the misbehavior that is demonstrated as maliciousness. Since a malicious node inflicts damage to the system for its own advantage, it is necessary to detect and isolate such a node from the system before it can negatively impact the network.

Undeniably, reputation-dependent systems enhance the protection of a system as they are capable of managing any sort of noticeable misbehavior. Systems, either reputation- or trust-based, permit nodes to be as a decision maker regarding to future transaction neighbors. For this reason, WSNs and MANETs are being configured as reputation and trust-based systems [22].

2.4 Data Aggregation and Byzantine Problem

Here the hierarchical network is projected as a distributed information aggregation network [10]. According to data provided by SNs, FNs calculate Aggregation Information which is then committed to APs. The computed information reported by SNs forms the basis of this hypothesis. The authentication that the information is precise; needs to be performed by the FNs because of the possibility of SNs to be compromised and reporting a falsified information. The APs' ability to verify the committed information bears vital importance. In the absence of these facilities, SNs might compromise and then provide incorrect information which in turn results in

misleading the entire network. This problem of detecting incorrect data from malicious nodes, leading to identification of falsified information to the BS, is referred to as the Byzantine problem [10]. Since malicious nodes can mislead the system their identification is of paramount importance. For instance, in case a compromised SN reports false information to the levels above, the aggregator (FN) cannot receive precise aggregation information because of the impact of the misbehaving node.

2.5 Security Issues in WSNs

The autonomous nature and the limited resources of WSNs make them vulnerable to attacks. Therefore, certain protective mechanisms within the network itself must be provided to safeguard WSNs' normal operation.

2.5.1 Security Techniques in WSNs

The following are the main reasons why traditional security measures cannot be directly implemented in WSNs [21-24]:

- Their moderate cost and limitations regarding energy, memory capacity, computational ability, and communication.

- the possibility to be used in aggressive public places where there is a greater risk of physical attacks.

- exposure to hacking and loss of secret data through controlling the SNs.

- utilization of insecure communication due to their dramatically recognizable feature.

Consequently, insufficient current security measures call for new methods. WSNs are exposed to numerous attacks due to how data is transmitted as well as SNs' general shortages in terms of resources available to them. Moreover, the conditions under which they operate leave them defenseless and unprotected to attacks. Below is a list of some of the vital security schemes for WSNs:

- Using a specific key for every node: Where there are huge overheads of storing the entire network keys, this scheme may not be employed due to the insufficient capability of every node [22].

- Distribution of a preloaded key: preloading a single network key that is often done at the time of deployment is a scheme with serious disadvantages because if one key is compromised it will endanger network in general.

- Random key probabilistic distribution [21]: The scheme designs protocols for WSNs by analyzing how SNs are connected using a random graph theory. The existence of this scheme in WSN protocols works in progress in three stages: directly shared key discovery, key initialization, and path key.

2.5.2 Wireless Sensor Networks Attacks

From the perspective of history of intrusions to WSN, considerable research has been conducted on several protection schemes [6-9, 12-14]. First, a discussion on Sybil attack will be presented in this section and followed by Wormhole, Sinkhole, and Denial of Service (DoS) attacks on WSNs plus countermeasures relevant to the network layer in the protocol layer as classified in Table 1.

Table 1: Denial-of-Service Attacks and Defenses by Protocol Layer [12]

Protocol Layer	**Attacks**	**Defenses**
Application Layer	Overwhelming Sensors	Sensor tuning Data aggregation
Transport Layer	Synchronize flood	SYN cookies
	Desynchronize attack	Packet authentication
Network Layer	Spoofed, Altered, or Replayed Routıng Information	Secure cluster formation Authentication and anti-replay protection
	Hello floods	Pairwise authentication
Data Link Layer	Interrogation	Authentication and anti-replay protection
	Denial of Sleep	Detect and Sleep Broadcast attack protection Authentication and anti-replay protection
Physical Layer	Jamming	Detect and Sleep
	Node destruction	Route around jamming regions

SNs are attack-prone as they are commonly deployed in less reachable areas. As a result of this, they are more susceptible to the following main attacks [8-12]:

- JAMMING: when the radio frequencies occur in physical layer, they can be interfered by jamming. This attack could be so powerful that is capable to disrupt the entire system or even if it is not powerful enough to disturb the entire system it can still disrupt a smaller portion of the network.

- TAMPERING: here an attacker tries to obtain important information such as cryptographic keys. It happens because the physical attack is given to a node and this attack involves a physical layer. It is possible to replace the node with a compromised node.

- SYBIL ATTACK: in this attack, the node duplicates itself and represents it in various locations within the network. Authentication techniques can be the solution for this problem.

- SINK ATTACK: The compromised nodes become attractive to the other nodes in its surrounding when the attacker attracts traffic to a specific node.

- REPLICATION ATTACK: The existing node ID is duplicated and added to a node by the attacker. The aggregated values will be corrupted and will degrade the whole network performance. This attack mainly causes energy drainage.

- WORMHOLE ATTACK: The packets are identified in a location in this attack. They are sent to the tunnels later on.

- COLLISION: If two nodes attempt to transfer data at the same time with same frequency, it causes collision attack.

- EXHAUSTION: It occurs in case of repeated collisions by the attacker

- SELECTIVE FORWARDING: The attacker forms malicious nodes in this type of attack. It forwards some messages while dropping the others in the network.

- HELLO FLOOD ATTACKS: HELLO packets are used by many protocols to identify their neighbor within the radio range. However, in a sensor network, a high-powered transmitter may be utilized by an attacker and make believe that they are neighbors.

- SPOOFED, ALTERED, OR REPLAYED ROUTING INFORMATION: In this type of attack, which deals with the routing protocol in WSN, the attacker may change, deceit or replay the information in order to disturb the traffic within the network.

- DESYNCHRONIZATION: This attack implies a disturbance in an existing connection.

- SESSION-HIJACKING: Session- hijacking refers to an attack on user session over a protected network, it is a malicious attack.

- DATA CORRUPTION: Data corruption attack happens when data is transmitting. It is done either by compromised nodes or by attack on data.

Sybil attack [12] disrupts the normal operation of WSN when a malicious node produces an unlimited number of forged identities. One possible solution to overcome this problem involves creating a lightweight identity certificate that uses a two-level Merkle hash tree which will then generate a node identity certificate. This method can also decrease computational overhead.

With Wormhole attacks, malicious nodes attack packets and perforate them to a new location inside the network, before re-transmitting the information. Wormhole attacks force malicious nodes to randomly drop the packets, relocate them in the network, and resend them. This action leads to generating false scenarios pretending the sender in the vicinity of a remote location. One way to overcome this attack is to use detector nodes that are location-aware and are capable of forwarding control packets that can identify wormholes as suggested in [8].

Sinkhole attacks target the surrounding nodes and bombard them with false routing information. Here, the intruder practices selective forwarding or it changes the information that is being transmitted. An algorithm for the detection of the intruder was suggested in [20]. The primary function of this is to check data consistency and to locate a list of suspicious nodes. Once the suspicious notes have been identified, the algorithm detects the attacker in the list by means of evaluating the flow information in the network.

There are more possibilities for the sensor networks to DoS attacks. Wood and Stankovic [24, 25] have discussed various DoS attacks and preventive measures were suggested in [12]. Apart from DoS attacks, some other attacks such as Intrusion Tolerance, General Intrusion Detection and Node Clone have been discussed extensively.

Attacks of various kinds are generally differentiated by means of two factors including transmitter efficiency and energy of battery with which the attack is carried out. Different types of attacks are characterized by limited power supply and resources. While in laptop and ad-hoc networks, devices launch attacks on the

network using greater power, eligible CPU, and sensitive antenna as they are supposed to attack on the powerful devices.

On the other hand, attackers can be categorized as inside and outside. When an accredited member of the network converts into a malicious node, an inside attack is said to happen. An outsider attack, however, occurs when a third party which is not a participant of the network engages in an attack. Unlike inside attacks, outsider attacks are relatively easier to manage [15].

A Denial of Service attack aims to manipulate access from authorized users. Although this type of attack is primarily intended to prevent efficient functioning of the services either temporarily or frequently, the aim of the attack may be different for different users.

Generally, the victim is inundated by many requests to communicate, a process which renders the system unable to give response to the authorized user at all, hence the effectiveness vanishes. An attempt to reset the victim or engage almost all of its resources can severely impede the communication path. The presence of DoS can be identified through some symptoms [29]:

- Slow performance of the network
- Unavailability of some of the websites
- Accumulation or boost in spam emails
- Packets are either lost or delayed without acknowledgment

Based on the type of destruction that it causes in the system; a DoS attack can occur in the following forms:

- Utilization of resources such as memory space, processor time, BW etc.
- Routing information is either deleted or altered
- State information such as resetting of TCP session is interrupted
- Physical components are damaged or destroyed
- Hampering the communication media between authorized devices

Attacks are defined as constant interference or random, deceptive and reactive functions. Since there is a variety of DoS attacks, it is possible that each layer experiences a different kind of DoS attack. Thus, each type of attack requires different options for its defense as described hereafter:

Jamming is one of the most common DOS attacks on WSNs in which data transmission occurs in regular intervals of time in case of constant jamming. When the attacker disguises itself as an authorized node within the network and perpetually transmits data, it is referred to as a flooding attack. Also, when an attacker monitors a data transfer within the network, he then sends jam signals. A random jammer operates by switching the modes between sleeping and jamming and avoids transmitting radio signals continuously. Jamming will continue for a while before it turns off radio and initiate “sleeping mode”. However, jamming will recapitulate after an interval of sleep [29].

In case of a deceptive jammer a continuous stream of bytes is sent into the network to simulate legitimate traffic. Arranging a reactive strategy is an alternative approach to jamming wireless communication. A reactive jammer only sends a radio signal once it senses some event in the channel. Otherwise, it remains inactive when the channel is in idle mode.

There are two types of jamming defense techniques known as active and passive modes. The presence of jamming is detected by a detection module equipped with SNs. This type of jamming is termed as active jamming.

Once discovered, the affected SNs are switched into sleep mode on their own, In the meantime, the unaffected nodes re-route the traffic. On the other hand, in passive jamming mode, effort is made to minimize the volume of packets that the nodes transmit in order to alleviate jamming effects.

Rendering a resource unavailable or degrading their performance for authorized users is what both DoS and DDoS aim to achieve. The two are distinguished by the fact that in the latter there are more than one host (malicious node) that attacks a system [22].

2.6 Trust-Based and Reputation Systems

Recently, the concept of trust and reputation has been widely used to develop security schemes for WSNs. Trust refers how a node finds the ability, trustworthiness and consistency of other nodes credible. This is determined by direct or indirect observation of the nodes' behaviors. A trust that is determined through direct observation is called direct trust, whereas indirect trust or reputation is that which is determined based on other nodes observations and opinions [17, 18].

A node is considered trusted if it has a trust equal to or larger than a pre-set minimum acceptable trust (MAT); otherwise, it is called untrusted or malicious node and will be subject to isolation. In trust-based mechanisms, only trusted nodes are accepted as a source of information and data forwarding [17].

2.6.1 Trust Determination

The core of any trust-based malicious node identification and isolation application is the fairness and cost of calculating the trust of the nodes. Therefore, it is important to have an algorithm that meets these objectives.

The major requirements of trust determination algorithms are [20]:

1) Designate new arriving nodes with a preliminary trust value. This will help other nodes to either fully trust or reject the new node.

2) Run a periodic appraisal of the trusted nodes in the network based on their current behavior and reputation among their neighbors.

3) Report the initial trusts of all nodes in the network with minimum power consumption and overhead.

Trust determination models are usually applied in dynamic and multivariable environments. Therefore, it is imperative that configuration parameters be carefully adjusted at each node so that they will be suitable to the environment and ensure optimal performance of the model. These include [11, 25]:

1) Minimum Acceptable Trust (MAT). The minimum trust level for a node to be recognized as trustworthy ($0 < MAT < 1$).

2) Trust Update Time (TUT). The minimum time interval before updating the existing trust of any single node in a WSN.

2.6.2 Reputation in Wireless Communication Networks

The vital goals pertaining to reputation and trust-based systems in the field of wireless communication networks were discussed in [20]. These include providing information that allows other nodes to determine the trustworthiness of other nodes in the same network, promoting cooperation between nodes, and preventing untrustworthy nodes from participating in the network activities. To these we should add a number of other important goals. The first goal concerns the system's ability to manage any kind of observable misbehavior. Secondly, it aims to lower the damage that the insider attacks may cause.

Every Reputation/trust-based WCN system must possess three major qualities for an effective and efficient operation [18]. These characteristics include the following:

(a) The system must contain a robust existence that encourage anticipations for future connections.

(b) The system must have the ability to receive and send out feedbacks about the existing communications between its components parts and ensure the availability this information in the future.

(c) The system must utilize a feedback-based mechanism governing trust decisions.

2.6.3 Weighted Trust Evaluation Scheme

Neighbor-Weight Trust Determination (NWTD) algorithm [15, 16] is one of the trust determination algorithms that have been traditionally developed [10-14]. The NWTD algorithm is modified and used so as to establish and assess how a fair trust-based

malicious node detection and isolation scheme (FTMNDI) performs in flat WSNs to periodically update the trust of the nodes based on their reputation.

In the FTMNDI scheme, a trust value of 1 would suggest that all SNs are considered to be trusted as a whole. In case of the transmission of falsified information (i.e. a node is behaving maliciously) observed by the monitoring node, a true positive malicious node is identified. In case the trust value of a particular node drops below the set MAT, any malicious node within the network is separated. A number of simulations determine the performance of the FTMNDI scheme using the network simulator MANSim [24].

The simulations help investigate the effect of a number of monitoring nodes, trust update factor and minimum acceptable trust on the malicious node lifetime. The most dominant factor and the near optimal values of the above parameters were also found by employing data envelopment analysis.

A Trust-Based Intrusion Detection approach was proposed in [35]. In order to identify malicious nodes in the WSN, Trust-Based Intrusion Detection considers a composite trust metric that is obtained from social trust as well as quality of service (QoS) trust. With the application of intrusion detection by the CH, the SNs evaluate the trust worthiness and maliciousness of the CMs. To achieve this, peer-to-peer trust evaluation results which are gathered from different SNs are examined statistically.

2.7 Related Works

The security issues of WSNs have been subject to extensive research in recent years. These networks are susceptible to faults and malicious attacks due to their limited resources and inherent nature. Such vulnerability calls for effective detection and

isolation of malicious nodes IDS before they are able to adversely impact the network. To overcome this, most proposed security schemes are based on trust and reputation mechanisms that add more security options for decision-making in WSNs.

Idris *et al.* [7] proposed a novel model for detecting malicious node by evaluating weighted trust (WTE). They introduced hierarchical clustered network topologies that help reduce the communication overheads between two SNs. In the same vein, Hongbing *et al.*'s innovative scheme [8] used the WTE algorithm for hierarchical WSNs. To detect malicious nodes, they attached a weight-recovery value relevant to the node behavior during a specified past period. Ju *et al.* [9] proposed a new mechanism which was also based on the WTE algorithm to defend WSNs against maliciously behaving nodes by updating the weight value assigned to each node.

Sajjad *et al.* [15] used the trust level of the neighboring nodes to develop an IDS for WSNs. They studied the network statistics and malicious node behaviors and reported that selective forwarding, Hello flooding, and jamming attacks were successfully detected by their IDS. Accordingly, their scheme effectively isolated the malicious nodes.

Potukuchi and Kant [16] proposed a secure data-aggregation framework for WSNs that benefitted from a trust monitoring system (TMS) installed at the level of nodes and with an IDS next to the BS. Every network node evaluates the its neighboring nodes' trust via the TMS system and runs network activities including selecting CH, aggregating data, and sending reports to BSs which will then analyze the incoming data utilizing the IDS and report any maliciously behaving nodes back to the network.

Oh *et al*. [17] presented a scheme to identify malicious and malfunctioning nodes by using dual-WTEs in hierarchical WSNs. Their scheme was able to effectively detect nodes that were identified as malicious by means of a weighted-majority voting technique without sacrificing normal nodes that develop transient faults. Yim and Choi [18] presented a neighbor-based malicious node detection scheme for WSNs that estimated the trustworthiness of the nodes by their confidence levels and achieved a high detection accuracy with a low false alarm rate.

Umashankar *et al.* [19] introduced a detection system using a protocol layer trust-based intrusion (LB-IDS) to protect WSNs through detection of intrusions at different layers. In their proposed system, using trust deviation metrics, the SN's trust value is calculated at each layer trust—physical layer trust, media access control (MAC), and network layer trust—with respect to each attack. The trust of a SN at a particular layer is calculated by taking key trust metrics of that layer. Finally, each layer's trust value is accumulated and calculated to find out the collective SNs' trust values. By applying the trust threshold, SNs are identified as trusted or malicious. Evaluation of the performance of LB-IDS scheme is done by the accuracy of detection, and false-positive and false-negative rates.

Ramya and Basith [20] proposed an efficient WTE system for WSNs, which can detect maliciously behaving nodes by a WTE methodology and cooperation among the CHs. Their approach enhanced the system's efficiency by lowering the memory, energy, and communication overheads from those of other schemes. Babu *et al.* [21] proposed TENCR—a novel method to evaluate trust using nodes' quality of service (QoS) characteristics (the trust metrics) as well as the recommendations of adjacent nodes. The QoS characteristics give the DT of a node, whereas the reputation of the

node among its neighbors gives the IT. Their technique resulted in an efficient detection of the malicious and selfish nodes and admitted only the trustworthy nodes to the routing process.

Kumar and Dutta [22] presented a novel scheme for aggregating data using a trust-based reputation model to guarantee that aggregated data are secure and reliable. Their scheme applied linguistic fuzzy logic and fuzzy sets to their trust and reputation model, aiding the selection of secure paths from SNs to the BS. Their proposed scheme produced more accurate results with a decrease in energy consumption compared to various existing schemes.

Fu and Lui [23] proposed a novel cluster-based data fusion model to ensure security and accuracy of fusing data in WSNs. Their model integrated a clustering mechanism, trust and reputation arrangement, and data fusion algorithms. After clustering, the trust system selected two CHs for every cluster using its members' reputation. Each CH independently performs data fusion. The results were then reported to the BS which then computed the dissimilarity coefficient. In case the coefficient exceeded a preset value, the CHs were assumed to behave maliciously and other CHs were elected. In another trust evaluation model introduced by Chen *et al.* [24] they used a data fusion mechanism in WSNs that integrated the data trust (calculated by processing the sensor data), behavioral trust (calculated by monitoring the behavior of nodes when sensing and forwarding the data), and historical trust (updated trust with weighted calculations). Both models aim to make the data in WSNs more reliable and creditable.

Recently, we investigated the performance of an FTMNDI scheme [25], which periodically updates the nodes' trust in a flat network by an algorithm that uses neighbor-weight trust determination. The update is based on the node's reputation and assumes a fixed trust-update factor. However, the FTMNDI scheme ignores several major challenges in network security, that is, the malicious behavior of nodes, the network's architecture, and trust recovery issues, which are addressed in the present research. Moreover, as the FTMNDI scheme makes the malicious node already known to all other nodes, it does not compute the DR and/or MDR.

To our knowledge, most of the previously proposed weighted-trust schemes update the weight or trust value depending on the number of maliciously behaving nodes or apply a constant penalty factor. To make such schemes as effective and accurate as possible, we here consider the number of error-data reported by each node and hence introduce a dynamic trust-update factor based on the behavior of each SN.

2.8 Conceptual Framework

The performance of the enhanced WT-MND scheme for detection of malicious node in WSN is assessed by means of three identified metrics, i.e. response time, ratio of misdetection, and rate of detection.

1) Detection Ratio (DR)—the ratio of malicious nodes identified by the scheme to the total number of malicious SNs that exist in a WSN—can be utilized as an indicator of scheme effectiveness. The presence of misleading reports originated from malicious SNs and residing in the WSN can be identified if the value of DR is high, thus making it possible to eliminate them.

$$DR = \frac{\text{Number of correctly detected malicious nodes}}{\text{Total number of malicious nodes}}$$

2) Misdetection Ratio (MDR)—the ratio of misdetected nodes to the total number of all detections made by the scheme—includes both the correctly detected and incorrectly detected nodes— Misdetected nodes belong to two classes, i.e. malicious nodes considered normal as well as normal nodes considered malicious. In order to reduce false positives, the misdetection ratio of the scheme should be as low as possible.

$$MDR = \frac{\text{Number of misdetected nodes}}{\text{Total number of detections}}$$

3) Response Time (RT)—the average number of rounds required by the scheme to accurately identify malicious nodes in the WSNs—is employed to show the agility of the proposed scheme to help identify malicious nodes present in WSNs.

Chapter 3

AN EFFICIENT WEIGHTED TRUST-BASED MALICIOUS NODE DETECTION (WT-MND) SCHEME

3.1 Network Architecture

This work adopts a well-known clustering mechanism in WSNs called the LEACH protocol, which hierarchically clusters the SNs as shown in Fig. 6. Each cluster consists of several CMs and one CH [26]. The CMs sense the data and transmit them to their CH in a time-division multiple access (TDMA) manner. Each CH then aggregates the data to reduce the data redundancy and transmits them to the BS using the code-division multiple access (CDMA) technique. The LEACH protocol operates through rounds, each consisting of two phases: the setup phase and the steady-state phase. The setup phase performs three operations: CH selection, cluster formation, and TDMA/CDMA scheduling. The steady-state phase implements the data transmission [27-30].

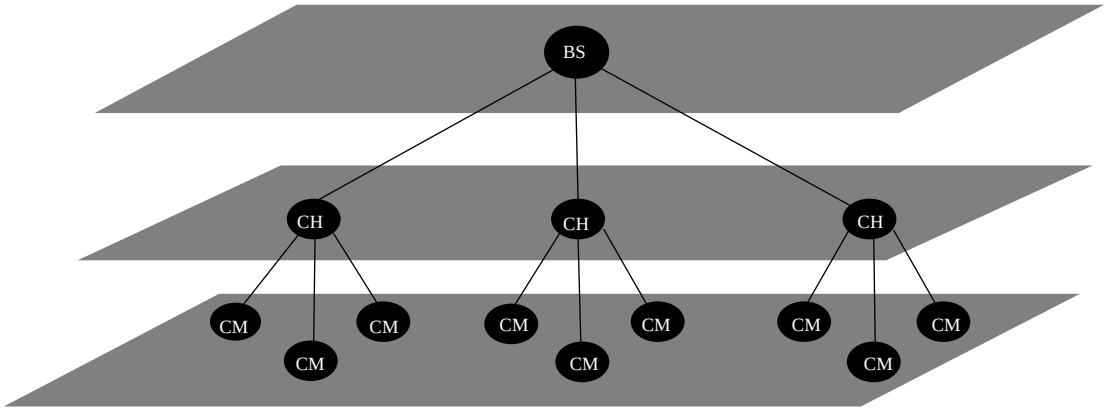

Figure 4: WSNs Architecture

LEACH minimizes energy dissipation by dividing WSNs into clusters in order to minimize the number of communicating messages and intercept a direct communication between SNs and the BS [26]. The data aggregation can exclude many redundant data to decrease the communication load on the cluster head (CH) node. The energy of the CH is rapidly exhausted because it has more duties and processes than other nodes. After a node become as a CH in a round, then it passes this role to another node in order to balance the power consumption between all nodes in the WSN. The CMs communicate with their CH by a single-hop using TDMA slots, and then each CH forwards aggregated data to the BS directly. In order to avoid internal communication collisions, CHs use a TDMA schedule for their members, and the BS classify the CHs with a CDMA schedule [27]. The operation of LEACH is divided into rounds; each round consists of two phases: the setup phase and the steady-state phase as described below.

3.1.1 The Setup Phase

Step 1: CH Selection [27]

In this step, each candidate node selects a random number between 0 and 1. Then each node will compare its random number with a threshold value T(n). If the case that the selected random number was lower than the T(n), then that candidate node will be chosen as the CH currently for that round.

$$T(n)=\begin{cases} \frac{P}{1-p\left(r \bmod \frac{1}{P}\right)} & , n \in G \\ 0 & , O.W. \end{cases}$$

Where:

P the desired percentage of cluster heads (e.g., P = 0.05)

r is the current round

G is the set of nodes that have not been cluster-heads in the last 1/P rounds.

Some of the CHs in the most recent 1/P rounds will not be acted as a CH in the current round according to the equation of T(n). The node immediately announces its CH status to all other normal nodes by broadcasting a cluster head-advertisement message (HEAD_Adv_Msg).

Step 2: Formatting of Clusters [27]

After the normal nodes received the HEAD_Adv_Msg broadcasted from the CH, then they send join-cluster message (JOIN_Clu_Msg) to the CH for which it has the highest received signal strength. The JOIN_Clu_Msg contains the CH's ID and the node's ID's.

Step 3: TDMA and CDMA Scheduling [27]

After SNs are grouped into clusters, each CH creates a time slots through a time division multiple access (TDMA) and distributes them to all its CM nodes. Each CH also selects a code division multiple access (CDMA) code that it will use to forward the aggregated data to the BS. To handle a large number of nodes, where a number of nodes simultaneously and asynchronously access a channel, CDMA is a good choice as a MAC protocol [38]. LEACH also uses CDMA so that each cluster uses a different set of CDMA codes and transmits aggregated data to the BS in the upper level of the hierarchy, to minimize interference between clusters. CDMA ensures data transmitting in parallel and reduces the delay significantly.

3.1.2 The Steady-state Phase

Step 1: Data Transmission

The CMs start sensing in their coverage areas and forward sensed data to their CH within the TDMA time slot assigned in Step 3 [27-30]. In each round, CMs will do sensing and transmitting for the sensed data to their CH n-1 times through n-1 frames; each one in its time slot in each frame. Each round consists of start-up phase

and steady state phase. While the steady state phase consists of n frames (n-1 frames are employed for transmitting the sensed data and the last frame is concerned for transmitting the trust values of all other nodes in the same cluster, which indicates how much that node is trusting each).

Step 2: Multiple Clusters [27]

The CH node will collect the sensed data from its CMs, while some nodes can be isolated from the network because of its behavior using WT-MND scheme. CH will start data fusion and do data compression and data aggregation to be transmitted to the BS. The state of the network will return to Step 1 of the setup phase and a new round is started.

The next sections will be devoted to explaining the adopted LEACH protocol using WT-MND scheme and what additional steps and improvements that we did related to the "reputation and trust based" for the purpose of detecting and isolating the malicious nodes in the clustered WSNs.

3.2 Node Behavior Modeling

A SN can be classified as a normal node or a malicious node depending on its trust value. A normal node usually sends correct data but can behave maliciously when the sensing data are faulty, when the data transmission is interrupted, or when falsified (error) data are sent. On the other hand, a malicious node usually sends falsified data but sometimes transmits correct data to protect itself from detection. For example, suppose that the SNs are sensing the temperature of an environment and transmitting the sensed data to their CHs, which then aggregate the data. Let D denote the weighted average of the temperature sensed by the nodes and δ be a small

acceptable variation from D (i.e., let $D \pm \delta$ be accepted in the data readings). When data from a node lie outside this range, they are considered as falsified or error data.

We assume that malicious and normal nodes randomly send falsified data in each TDMA slot with probabilities P_{m_err} and P_{n_err}, respectively using uniform random distribution. As the malicious nodes are randomly selected at the initialization stage of the simulation, we also define the malicious node probability P_m using uniform random distribution (i.e., the percentage of malicious nodes among all nodes in the network). Without loss of generality, we assume that the BS behaves non-maliciously in all rounds and that the CHs do not behave maliciously during their elected rounds.

3.3 Malicious Node Detection

The assumptions of the proposed WT-MND scheme are listed below:

1. Recall that a single LEACH round consists of the setup phase and the steady-state phase. The optimal number of frames in each round, which minimizes the tradeoff between the lifetime and the throughput of the network, is five [31]. Hence, the steady-state phase is assumed to hold five frames as shown in Figure 7. In each frame, the number of TDMA slots equals the number of CM nodes.

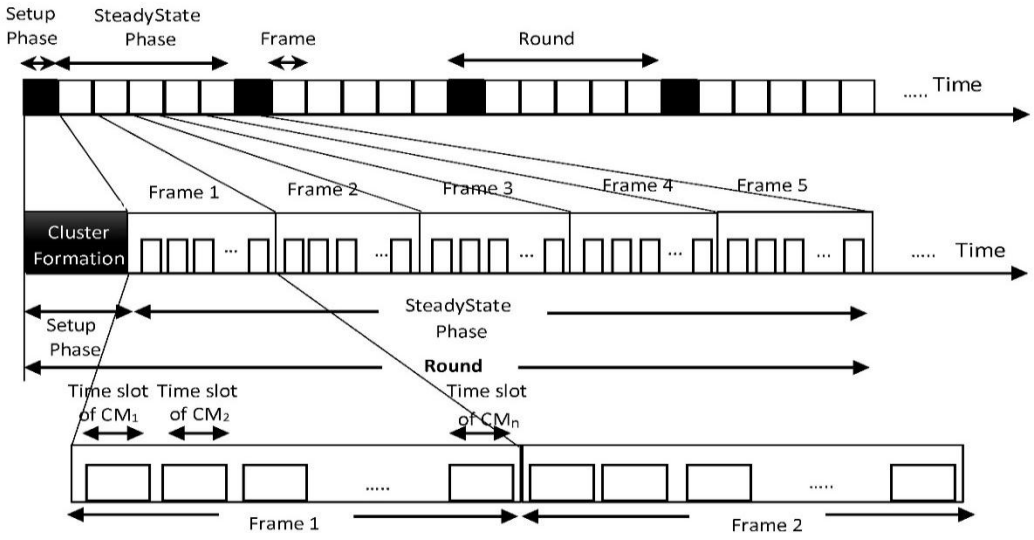

Figure 5: Format of a WT-MND Round

2. Each CM transmits its data in the corresponding time slot of each frame. However, the last frame of each round is reserved for transmitting the DT values of each CM to its CH. The DT values of the *i*th CM (CM_i) specify CM_i's level of trust in all other CMs in its cluster (i.e. Figure 8 shows that CM_1 will transmit to its CH the DT values of how much CM_1 trust CM_2, CM_3, CM_4,CM_5 and itself. The same happened for CM_2, CM_3, CM_4 and CM_5, those nodes will transmit the DT values for each of them with the all other CMs to their CH).

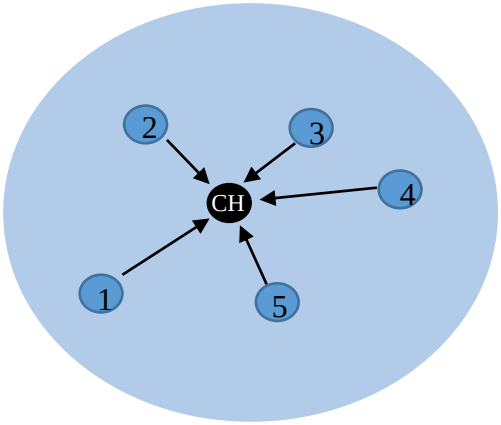

Figure 6: DT for each CM with all other CMs

3. Each node has a trust vector (TV) containing the DT values of all N nodes in the network. Each node fully trusts itself. Initially, all entries of the TV are set to 1 as shown in Table 2.

Table 2: TV of DT Values for Node *i* Initially with all other *N* SNs in the Network

DT_1	DT_2	DT_3	DT_4	DT_5	……..	DT_i	……	DT_N
1	1	1	1	1		1		1

4. The CH in each cluster works as a *monitoring* node that monitors the behaviors of its CM nodes, calculates their IT values, and decides whether to admit or to isolate each node. The calculated IT value of CM_i defines the reputation of CM_i with respect to all other CMs.

5. Each CH maintains a trust-update vector (TUV) containing the current trust-update factors of all of its CMs as shown in Table 3.

Table 3: TUV of Each CH with the Current Trust-Update Factors of all of its *k* CMs.

α_{CM_1}	α_{CM_2}	α_{CM_3}	α_{CM_4}	α_{CM_5}	α_{CM_6}	……	α_{CM_k}

Figure 9 is a functional block diagram of the WT-MND scheme based on the two main phases of the LEACH protocol. The setup phase was explained in Section 3.1, and the steady-state phase implicitly implements the WT-MND scheme. The WT-MND scheme is structured into four main components: IT calculation, malicious

node isolation, trust-update factor computation, and DT updating. These four components are explained in detail below.

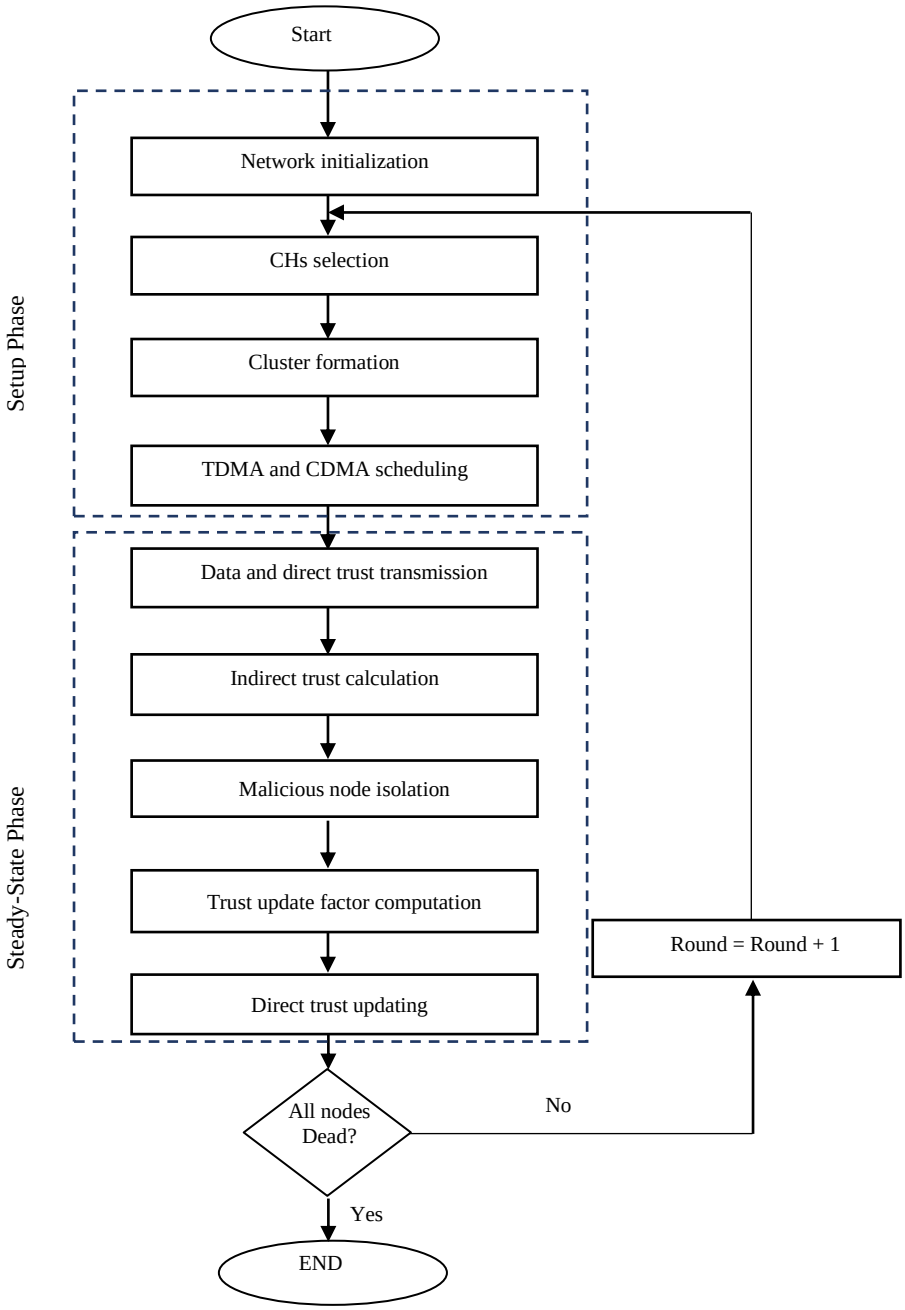

Figure 7: Functional Block Diagram of the WT-MND Scheme

3.3.1 IT Calculation

In each round, each CM_i transmits its DT values of all other CMs to its CH. The CH in each cluster then calculates the average trust, the weight, and the IT of each CM_i through the following steps.

In order to clarify these steps, we consider one cluster of three CMs (see Figure 10). The same steps are applied to all other clusters in the WSN.

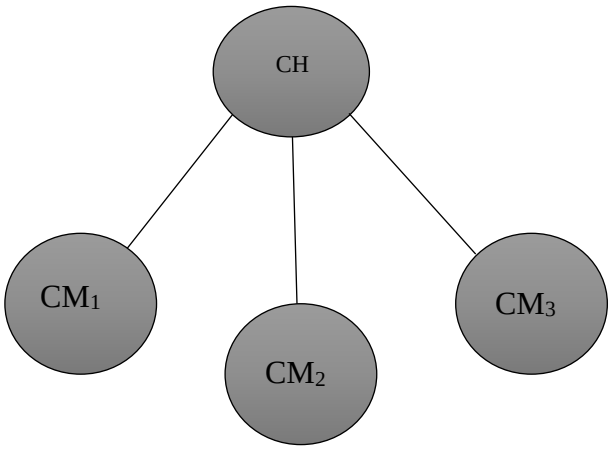

Figure 8: Example of a Cluster with Three CM

(1) The average trust ($\bar{T}$) of each CM_i is determined by all other CMs:

$$\bar{T}_{CM_i} = \frac{1}{k}\sum_{j=1}^{k} RDT_{CM_i.CM_j} \quad (i = 1 \text{ to } k), \tag{1}$$

where $RDT_{CM_i.CM_j}$ is the reverse DT of CM_i determined by CM_j (i.e., the extent to which CM_i is trusted by CM_j) and k is the number of CMs.

In the above example, this step yields the following result:

$$\bar{T}_{CM_1} = (RDT_{CM_1.CM_1} + RDT_{CM_1.CM_2} + RDT_{CM_1.CM_3})/3,$$

$$\bar{T}_{CM_2} = (RDT_{CM_2.CM_1} + RDT_{CM_2.CM_2} + RDT_{CM_2.CM_3})/3,$$

$$\bar{T}_{CM_3} = (RDT_{CM_3.CM_1} + RDT_{CM_3.CM_2} + RDTT_{CM_3.CM_3})/3.$$

(2) The CH calculates the weight of each CM_i (W_{CMi}), accounting for the average trust of all other CMs. Note that all W_{CMi}'s in a cluster sum to 1, and the individual W_{CMi}'s indicate the reputation of the CM_i's among the other CMs. A normal node is weighted more heavily than a malicious node.

$$W_{CM_i} = \frac{\bar{T}_{CM_i}}{\sum_{j=1}^{k} \bar{T}_{CM_j}} \quad (i = 1 \text{ to } k). \tag{2}$$

In the above example, W_{CM_i}'s are calculated as follows:

$$W_{CM_1} = \frac{\bar{T}_{CM_1}}{\bar{T}_{CM_1} + \bar{T}_{CM_2} + \bar{T}_{CM_3}},$$

$$W_{CM_2} = \frac{\bar{T}_{CM_2}}{\bar{T}_{CM_1} + \bar{T}_{CM_2} + \bar{T}_{CM_3}},$$

$$W_{CM_3} = \frac{\bar{T}_{CM_3}}{\bar{T}_{CM_1} + \bar{T}_{CM_2} + \bar{T}_{CM_3}}.$$

(2) The CH calculates the IT of each CM_i ($IT_{CM_i.CH}$) as follows:

$$IT_{CM_i.CH} = \sum_{j=1}^{k} W_{CM_j} \cdot RDT_{CM_i.CM_j} \quad (i = 1 \text{ to } k). \tag{3}$$

In the above example, $IT_{CM_i.CH}$ is given by

$$IT_{CM_1.CH} = W_{CM1} \cdot RDT_{CM_1.CM_1} + W_{CM2} \cdot RDT_{CM_1.CM_2} + W_{CM3} \cdot RDT_{CM_1.CM_3},$$

$$IT_{CM_2.CH} = W_{CM1} \cdot RDT_{CM_2.CM_1} + W_{CM2} \cdot RDT_{CM_2.CM_2} + W_{CM3} \cdot RDT_{CM_2.CM_3},$$

$$IT_{CM_3.CH} = W_{CM1} \cdot RDT_{CM_3.CM_1} + W_{CM2} \cdot RDT_{CM_3.CM_2} + W_{CM3} \cdot RDT_{CM_3.CM_3}.$$

3.3.2 Malicious Node Isolation

The CH compares the IT of each CM calculated by Eq. (3) with the preset MAT value. Any node with an IT value below MAT is considered a malicious node and is isolated from the network. Otherwise, the node is considered a normal node and is admitted. It should be noted that MAT lies between 0 and 1, and its value depends on how sensitively the network application must quickly isolate malicious nodes in the network [25].

3.3.3 Trust-Update Factor Computation

As mentioned above, each CM_i in each cluster sends its sensed data to its CH in the first four TDMA frames of each round. The CH aggregates the sensed data, observes the behavior of each CM_i by recording the number of error-data transmissions by CM_i per round [$T_{err}(i)$], and computes the corresponding trust-update factor $\alpha(i)$ by the following heuristic formula:

$$\alpha\,(i) = 1 - \frac{T_{err}(i)}{2*F} \qquad (i = 1 \text{ to } k), \tag{4}$$

where F is the number of data frames per round (four in our case).

Hence, at the end of each round, every CH forms its resultant TUV as [$\alpha(1)$, $\alpha(2)$, …, $\alpha(k)$]. It is worth mentioning that α of a suspected malicious node (Section 3.3.2) is replaced by the node's physical address in the TUV.

Clearly, detected falsified information reduces the $\alpha(i)$ value of CM_i. However, this falsified information may simply result from a temporary fault in the communication channel, which neither compromises nor disables the node. Therefore, continuously decreasing the trust-update values of such nodes is a nonpractical solution. Instead, the DT value should be recovered if the SN functions normally after an interruption.

The trust value can be implicitly and adaptively recovered by recalculating the α values of the CMs at each round.

For more declaration, the vector of T_{err} presents how many times each CM_i sent error data in one round. Furthermore, we assume that each round contains five frames; first four frames for data transmission and the last frame is for transmitting the DT values of CMs.

In the above example, if

- CM_1 transmits zero error messages (i.e., $\boldsymbol{T_{err(1)} = 0}$),
- CM_2 transmits one error message due to faulty sensing data (i.e., $\boldsymbol{T_{err(2)=}1}$),
- CM_3 transmits three error messages (i.e., $\boldsymbol{T_{err(3)} = 3}$),

Then, the CH will fill the vector of the $\boldsymbol{T_{err}}$ values for its CMs as shown below.

Frame #	Frame1	Frame2	Frame3	Frame4
Data Status of CM_1	√	√	√	√
Data Status of CM_2	x	√	√	√
Data Status of CM_3	√	x	x	x

0 1 3

Vector of T_{err} is Array of *1xk* indices

$\boldsymbol{T_{err}}$ for each CM_i can be varied between 0 and 4

0 means no error data sent by CM_i

4 means, CM_i sent error data all the time during the round

So, we obtain

$$\alpha\,(1) = \mathbf{1}\text{-}\frac{\boldsymbol{T_{err(1)}}}{\mathbf{2}*\boldsymbol{F}} = \mathbf{1},$$

$$\alpha\,(2) = \mathbf{1}\text{-}\frac{\boldsymbol{T_{err(2)}}}{\mathbf{2}*\boldsymbol{F}} = \mathbf{7/8},$$

$$\alpha\,(3) = \mathbf{1}\text{-}\frac{\boldsymbol{T_{err(3)}}}{\mathbf{2}*\boldsymbol{F}} = \mathbf{5/8},$$

respectively. The corresponding TUV is [1 7/8 5/8].

Each CH broadcast its obtained TUV to all *N* nodes in the network.

3.3.4 DT Updating

Based on the TUVs broadcasted by all CHs, each node updates its TV by multiplying its old DT values by the corresponding α values, as shown in the following equation:

$$\mathrm{TV}_{(i)_{\mathrm{updated}}}[j] = \alpha(j) * TV_{(i)_{\mathrm{old}}}[j], \qquad (i = 1 \text{ to } N)(j = 1 \text{ to } N). \tag{5}$$

In Table 4, it shows a sample of updated TV for i_{th} Node after applying equation (5).

Table 4: TV of DT Values for Node *i* with all other Nodes in the Network

$DT_{i,1\ Updated}$	$DT_{i,2\ Updated}$	$DT_{i,3\ Updated}$	$DT_{i,4\ Updated}$	….	$DT_{i,N\ Updated}$
$\alpha(1)$ $* DT_{i,1\ Old}$	$\alpha(2)$ $* DT_{i,2\ Old}$	$\alpha(2)$ $* DT_{i,2\ Old}$	$\alpha(3)$ $* DT_{i,3Old}$	…..	$\alpha(N)$ $* DT_{i,N\ Old}$

In the following round, the last frame of each CM again transmits the updated DT values of all other CMs to the CH (see Fig. 3). Moreover, in order to ensure that the

normal nodes fairly update their DT values, the TVs are reinitialized after a desired number of rounds (e.g., 40 rounds).

3.4 Computation and Transmission Overhead

WT-MND scheme incurs a small overhead into the network beacause of Computation and Transmission overhead:

- Computation overhead
  - ✓ CHs calculate the IT values for each CM in their clusters using equations 1, 2 and 3.
  - ✓ CHs compare IT values of all their CMs with MAT for malicious node isolation.
  - ✓ CHs compute the Trust Update Factor using equation 4.
  - ✓ CMs update DT values using equation 5.
- Transmission overhead
  - ✓ Each SN transmits its DT values with all its CMs to its CH in Frame number five in each round.
  - ✓ Each CH broadcast its TUV to all *N* nodes in the network. While this point can be solved by forwarding the TUV to the BS with the aggregated data and then the BS will broadcast the TUV to all SNs, especially that the BS is chargeable and no worry about the power consumption.

Chapter 4

RESULTS

4.1 Simulation Setup

I n this Thesis a MATLAB program is developed to experiment and simulate WT-MND scheme to get the results. The proposed approach was evaluated in a simulated $100 \times 100\ m^2$ network area containing randomly distributed nodes. The nodes were randomly distributed in a uniform random distribution and clustered under the LEACH protocol (see Fig.ure 11). All CHs and the BS were assumed to behave as normal nodes all the time. All nodes were assumed to be stationary throughout the simulations, and the BS was located at (50,175).

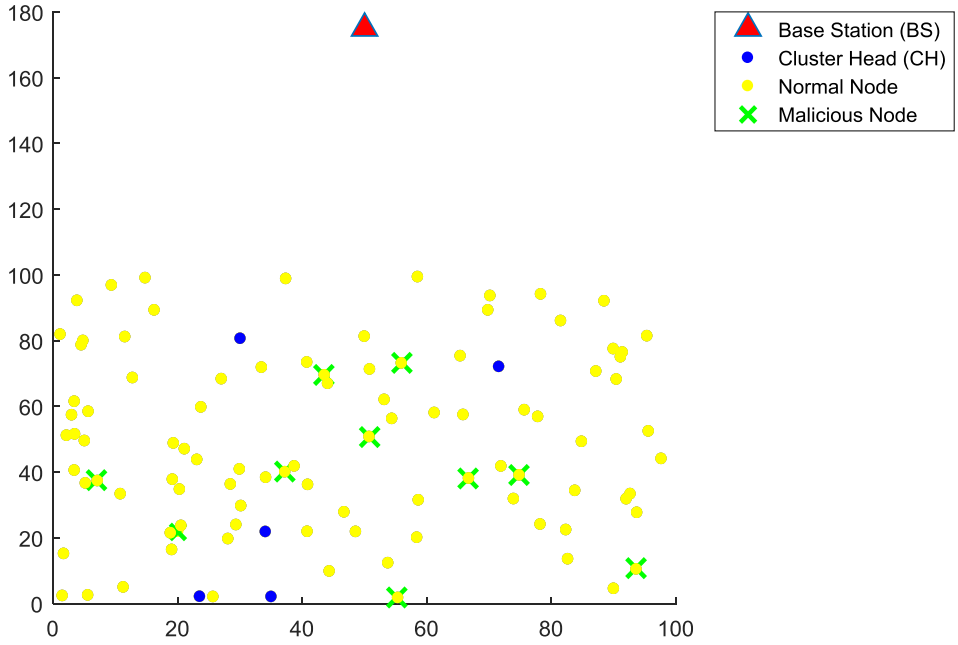

Figure 9: Sample Node Deployment in the 100 × 100 m2 Network Area

The simulation was carried out over several rounds. During each round, the CHs evaluated the behavior of their CMs and updated the trust level of any maliciously behaving node. When the IT of a node fell below the preset MAT value, that node was isolated. Thus, the malicious node lifetime can be calculated as the number of rounds before the node is isolated. The input parameters of the simulation are listed in Table 5 [10,32].

Table 5: Simulation Parameters

Parameter	**Value**
Network deployment area	$100 \times 100\ m^2$
Number of SNs, N	100–900
Number of CHs	5% of N
Location of BS	(50,175)
Initial power per node	0.5 J
Length of packets	6400 bits
E_{elec}	50 nJ/bit
Amplifier transmission (ε_{amp})	$100\ pJ/bit/m^2$
Probability of malicious nodes (P_m)	0.05–0.5
Probability of normal nodes transmitting error data ($P_{n\text{-}err}$)	0.05
Probability of malicious nodes transmitting error data ($P_{m\text{-}err}$)	0.6
MAT	0.3–0.9

In order to evaluate the simulated WT-MNDscheme, we constructed the confusion matrix [33] shown in Table 6 and calculated the DR and MDR performance metrics by Eqs. (6) and (7), respectively.

Table 6: Confusion Matrix [32]

		Detected	
		Normal	**Malicious**
Actual	**Normal**	True negative (TN)	False positive (FP)
	Malicious	False negative (FN)	True positive (TP)

$$\text{DR} = \frac{\text{Number of correctly detected malicious nodes}}{\text{Total number of malicious nodes in the network}},$$

$$\text{DR} = \frac{\text{TP}}{\text{TP+FN}} \tag{6}$$

$$\text{MDR} = \frac{\left(\begin{matrix}\text{Number ofnormal nodes falsely detected as malicious nodes} \\ + \\ \text{Number of malicious nodes falsely detected as normal nodes}\end{matrix}\right)}{\text{Total number of nodes}},$$

$$\text{MDR} = \frac{\text{FP+FN}}{\text{FP+TP+FN+TN}} \tag{7}$$

The performance of the WT-MND detection scheme was further evaluated by the *response time*, which expresses the average number of rounds before malicious nodes are correctly detected in the network.

The energy consumption of radio communication in the network was estimated by a simplified power control model [27,34-37]. The energy consumption of a node transmitting and receiving k-bit data in free space over a distance of *d* meters was computed by Eqs. (8) and (9), respectively:

$$E_{Tx}(k,d) = E_{\text{elec}} \times k + \varepsilon_{\text{amp}} \times k \times d^2 \tag{8}$$

$$E_{Rx}(k) = E_{\text{elec}} \times k \tag{9}$$

where E_{elec} and ε_{amp} denote the energy consumption of the transceiver and amplifier electronics per bit, respectively.

As is well known, most of the energy in a WSN is expended in node-to-node data transmission. The residual energy of node *i*, denoted by $E_{\text{residual}}(i)$, is the energy remaining in node *i* after data transmission or reception and/or computation.

Hence, after each round, the average residual energy of all live nodes *n* in the system is calculated as follows:

$$E_{\text{average}} = \frac{\sum_{i=1}^{n} E_{\text{residual}}(i)}{n} \tag{10}$$

4.2 Simulation Results

In order to evaluate the performance of the proposed WT-MND scheme, we varied the network environment and configuration parameters in the MATLAB simulation. During simulation, 30 runs are taken each time with duration larger than 10000000 unit of simulation time to ensure that all reported averages reach 95% confidence level. Please see Appendix A for detailed calculation.

In the Byzantine Generals' problem [8, 37], the decision of the loyal generals becomes inaccurate when the number of betrayed generals exceeds one-third of the total number. Similarly, if the number of malicious nodes in IDS exceeds 33% of the total node number, the malicious nodes will be difficult to be detected with accuracy. In order to ensure the safety of the network and similarly to previous researchers, we conservatively set the P_m value between 0.04 and 0.25 in the simulation study using uniform random distribution. The detection is terminated after 200 cycles or more than 25% of all nodes are detected as malicious nodes.

First, the proposed scheme was compared with WTE, WTE(R), and WTA [7-9]. Figures 12 and 13 plot the DRs and MDRs, respectively, as functions of the malicious node probability P_m (0.05-0.25) in a 100-node network of each type. As seen in Figure 12, the DR of all schemes slightly decreased with increasing P_m. This trend is attributed to the increasing number of malicious nodes as P_m increases; meaning that more falsified information penetrates the network. Consequently, the FN increases while the TP reduces. However, the WT-MND scheme achieved a higher DR than the other methods across the investigated range of P_m's. Note that, to highlight the differences among the four schemes, we have rescaled the *y*-axis to the range 0.8-1.0. The average DR improvements of WT-MND over WTE, WTE(R), and WTA were 3.6%, 2.8%, and 2.6%, respectively as shown in Table 7. Moreover, the MDRs were lower in WT-MND than in the other schemes across the range of P_m's (Figure 13). The average MDR improvements of WT-MND over WTE, WTE(R), and WTA were 89.7%, 31.6%, and 50.3%, respectively as shown Table 7.

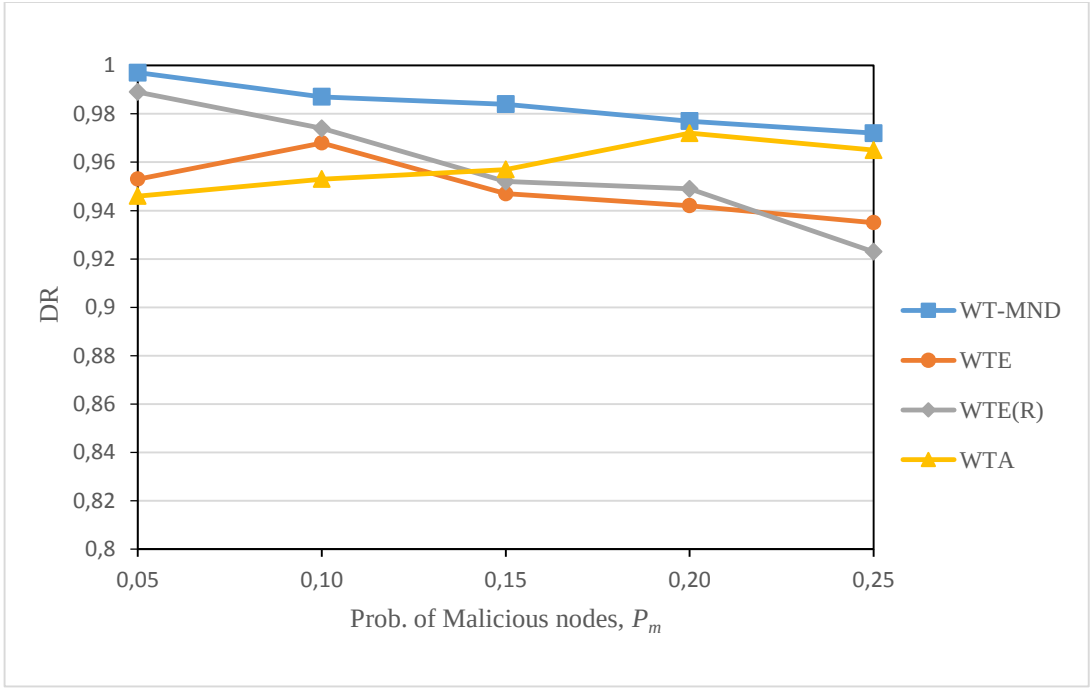

Figure 10: DR Versus Probability of Malicious Nodes in the Investigated Schemes

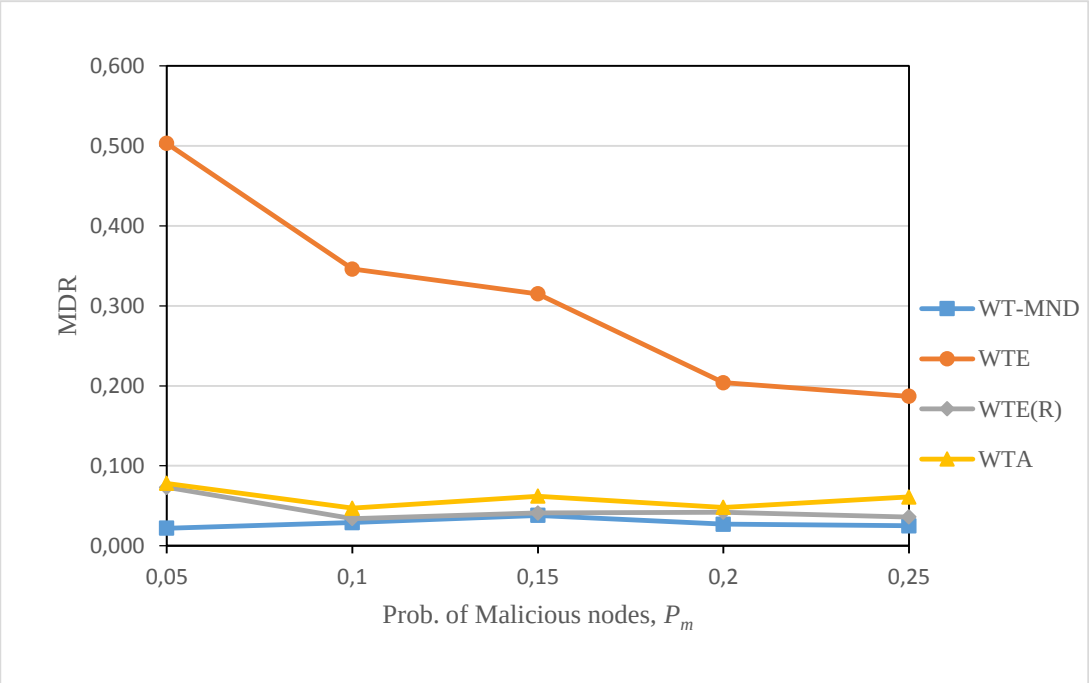

Figure 11: MDR Versus Probability of Malicious Nodes in the Investigated Schemes

Table 7: The Average DR and MDR Improvements of WT-MND over WTE, WTE(R), and WTA

DR	**Pm**					**Improvement %**
Scheme	**0.05**	**0.1**	**0.15**	**0.2**	**0.25**	
WTE	0.046170	0.019628	0.039071	0.037155	0.039572193	3.6
WTE(R)	0.008089	0.013347	0.033613	0.029505	0.053087757	2.8
WTA	0.053911	0.035677	0.028213	0.005144	0.007253886	2.6
MDR	**Pm**					**Improvement %**
Scheme	**0.05**	**0.1**	**0.15**	**0.2**	**0.25**	
WTE	0.956262	0.916185	0.879365	0.867647	0.86631016	89.7
WTE(R)	0.698630	0.147059	0.073171	0.357143	0.305555556	31.6
WTA	0.717949	0.382979	0.387097	0.437501	0.590163934	50.3

The proposed scheme was investigated with related to DR and MDR, respectively, as functions of the malicious node probability P_m (0.05-0.25) and the minimum acceptable trust MAT (0.3-0.9) in a 100-node network. As shown in Table 8, the DRs were almost the same for all MAT threshold values. This trend is attributed because the TP and FN values are almost not affected by changing the MAT values

within the same P_m. Moreover, the MDRs were slightly increased by increasing MAT values. This trend is attributed because of the increase in FP value as MAT increases (i.e. the normal nodes that were detected as malicious).

Table 8: DR and MDR on WT-MND Scheme with Different MAT and P_m Values

		MAT			
	P_m	**0.3**	**0.5**	**0.7**	**0.9**
DR	**0.05**	0.994	0.998	0.997	0.997
	0.1	0.989	0.991	0.987	0.993
	0.15	0.981	0.986	0.984	0.984
	0.2	0.976	0.982	0.977	0.978
	0.25	0.969	0.974	0.972	0.971
		MAT			
	P_m	**0.3**	**0.5**	**0.7**	**0.9**
MDR	**0.05**	0.016	0.019	0.022	0.031
	0.1	0.015	0.020	0.029	0.036
	0.15	0.019	0.023	0.038	0.044
	0.2	0.018	0.023	0.027	0.049
	0.25	0.021	0.022	0.025	0.047

Figure 14 plots the response time versus P_m in small (100-node) and large (900-node) networks. Increasing P_m (i.e., increasing the number of malicious nodes) increased the number of rounds before the malicious nodes were detected. As more malicious nodes appeared, the aggregated data were increasingly affected by the falsified information, and the malicious nodes were harder to detect. This difficulty was amplified in smaller networks, as fewer sources contributed to the reputation of the malicious nodes. However, the results demonstrate that the proposed scheme can detect and isolate malicious nodes within two to four rounds on average.

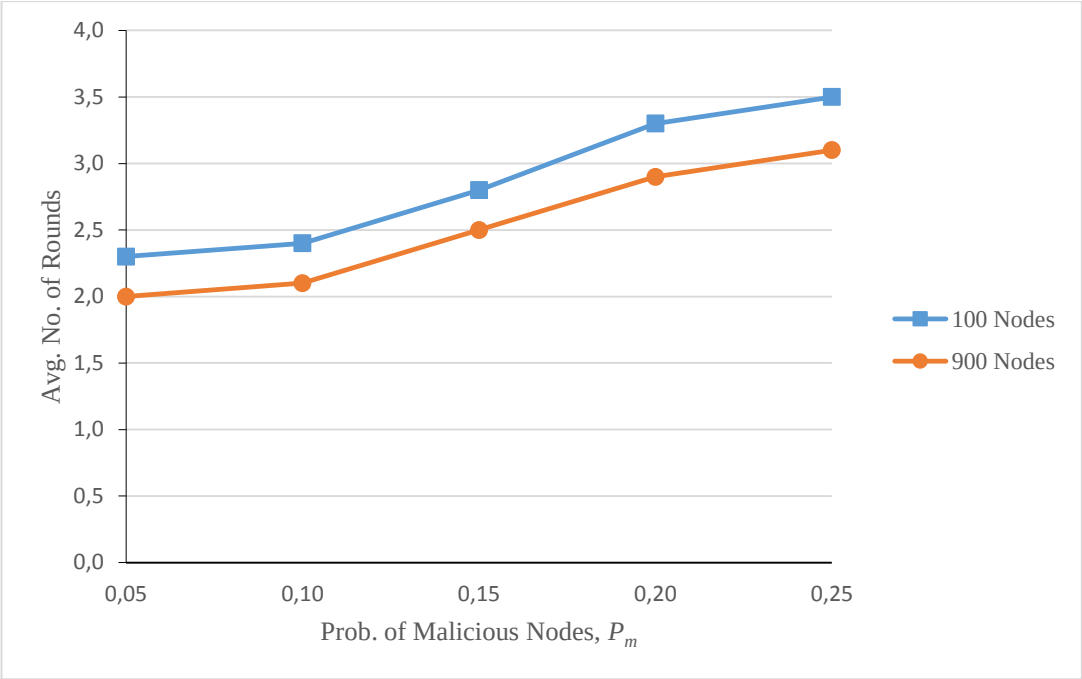

Figure 12: Response Time Versus Probability of Malicious Node Detection in the WT-MND Scheme

Next, in order to compare the scalabilities of the proposed scheme, WTE, WTE(R), and WTA, we computed the DRs and MDRs as functions of node number. Here, the number of nodes was varied from 100 to 900 while P_m was fixed at 0.04. The variations in DR and MDR were quantified by their standard deviations (SDs) indicator (i.e. the lowest in SD has the highest precision). The DR and MDR results of the four schemes are shown in Tables 9 and 10, respectively. Although the DR and MDR variations remained small as the node number was varied in all schemes, WT-MND achieved the smallest SDs among the four algorithms, confirming that the WT-MND scheme is more scalable than the other schemes. The average SD of the DR was 83.6%, 61.4%, and 82.3% lower in the WT-MND scheme than in WTE, WTE(R), and WTA, respectively (Table 9 and Figure 15). Similarly, the average SD of the MDR was 55.4%, 74.9%, and 71.0% lower in the WT-MND scheme than in WTE, WTE(R), and WTA, respectively (Table 10 and Figure 16).

Table 9: DR SD Versus Number of Nodes in the Investigated Network Security Schemes

Scheme	No. of node									SD
	100	200	300	400	500	600	700	800	900	
WT-MND	0.994	0.989	0.987	0.991	0.986	0.982	0.984	0.988	0.983	0.003887
WTE	0.894	0.979	0.963	0.958	0.957	0.953	0.941	0.958	0.959	0.023659
WTE(R)	0.993	0.989	0.982	0.984	0.982	0.974	0.972	0.967	0.963	0.010064
WTA	0.991	0.978	0.979	0.983	0.974	0.961	0.953	0.938	0.926	0.021953

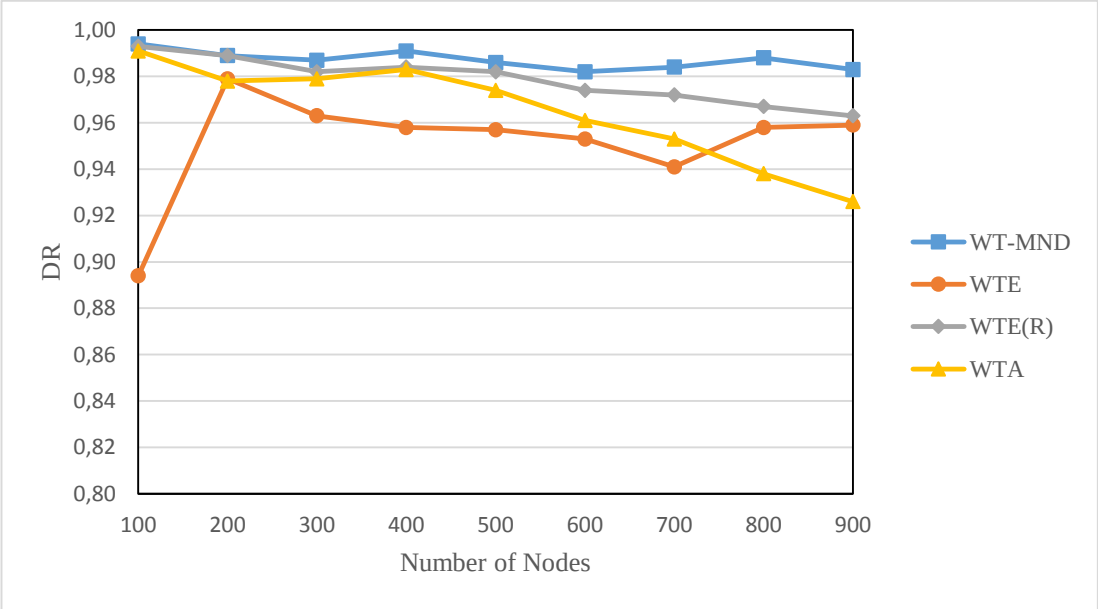

Figure 13: DR SD Versus Number of Nodes in the Investigated Network Security Schemes

Table 10: MDR SD Versus Number of Nodes in the Investigated Network Security Schemes

Scheme	No. of node									SD
	100	200	300	400	500	600	700	800	900	
WT-MND	0.012	0.018	0.026	0.028	0.021	0.032	0.037	0.035	0.031	0.008276
WTE	0.797	0.783	0.779	0.775	0.775	0.775	0.794	0.812	0.827	0.018566
WTE(R)	0.143	0.231	0.208	0.202	0.198	0.195	0.217	0.236	0.261	0.032964
WTA	0.016	0.062	0.055	0.047	0.058	0.083	0.093	0.098	0.104	0.028492

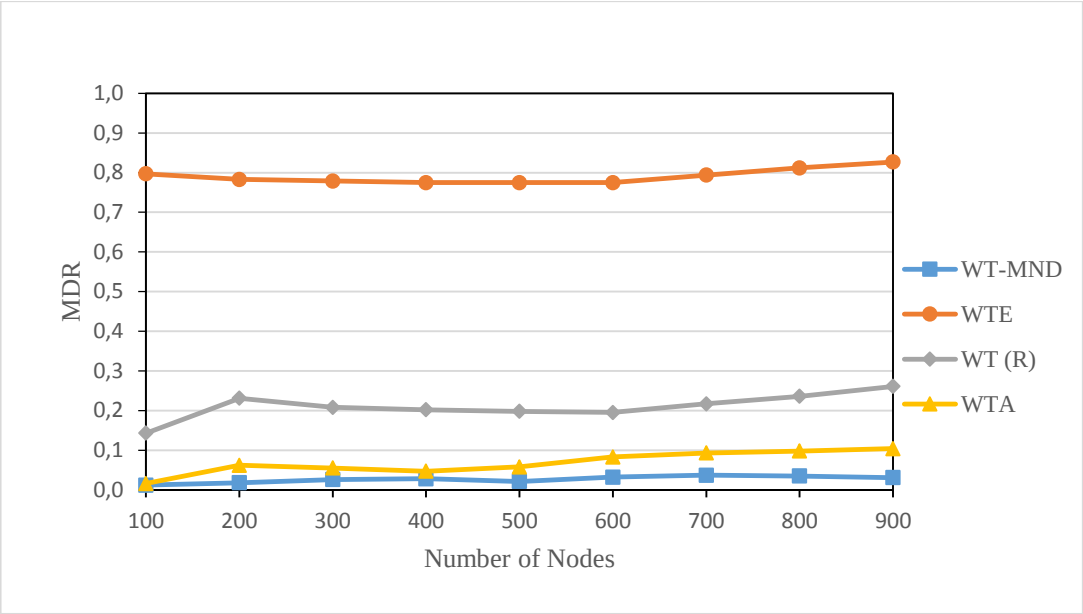

Figure 14: MDR SD Versus Number of Nodes in the Investigated Network Security Schemes

Table 11: The Average SD Improvements of the DR and MDR on WT-MND over WTE, WTE(R), and WTA

	DR		**MDR**	
Scheme	SD	Improvement %	SD	Improvement %
WT-MND	0.003887301		0.008276473	
WTE	0.023659036	83.6	0.018565949	55.4
WTE(R)	0.010063686	61.4	0.032963785	74.9
WTA	0.021953233	82.3	0.028491714	71.0

Next, in order to see the effectiveness of MAT on the scalabilities of the proposed scheme, we computed the DR and MDR values as function of MAT for the two network sizes (i.e. 100 nodes and 900 nodes) while P_m was fixed at 0.04. Table 12 shows that the number of rounds needed to find out the malicious nodes in the two network sizes is decreased by increasing MAT threshold values. That is because comparing the IT with a high MAT value takes less number of rounds for the malicious nodes to be detected than low MAT value.

Table 12: Response Time on WT-MND Scheme for Two different Size of Networks with Different MAT Values

	MAT			
No. of Nodes	**0.3**	**0.5**	**0.7**	**0.9**
100 Node	4.8	3.6	2	1.2
	5.1	4	2.1	1.5
	5.2	4.2	2.5	1.5
	5.6	4.7	2.9	1.8
	5.9	4.8	3.1	1.9
900 Node	5.2	3.8	2.3	1.4
	5.4	4.4	2.4	1.6
	5.8	4.7	2.8	1.8
	5.9	5.1	3.3	2
	6.2	5.4	3.5	2.1

Finally, Figure 17 compares the residual power of the WT-MND scheme and the traditional LEACH protocol in a 100-node network. Recall that the WT-MND scheme requires an extra frame per round for transmitting the updated trust values of a CM's associated nodes. This incurs a small overhead; consequently, the residual power and the average residual energy per node dropped by 1.5% (on average) from those of LEACH.

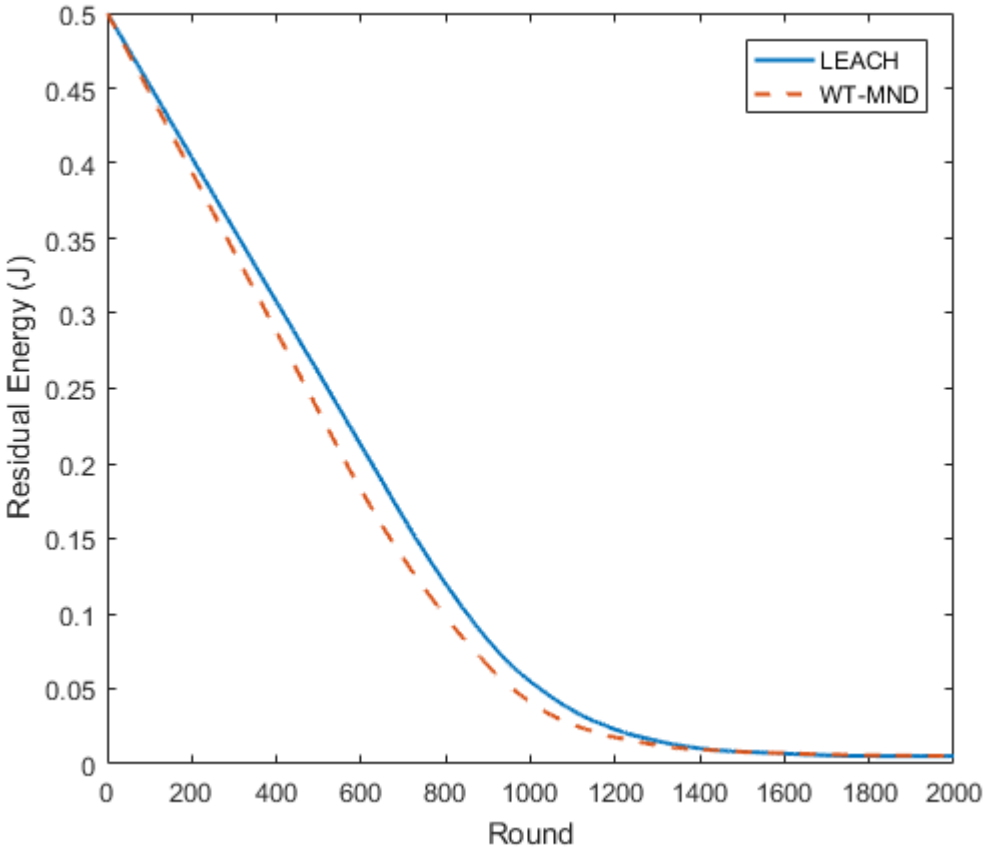

Figure 15: Residual Power Comparisons of WT-MND and LEACH

Chapter 5

CONCLUSION

In this thesis, we developed and evaluated the performance of an efficient weighted-trust-based malicious node detection and isolation scheme for WSNs, namely, the WT-MND scheme. Adopting the LEACH clustering protocol, WT-MND determines and updates the trust of the network nodes on the basis of their reputation. The scheme relies on an adaptive trust-update process that implicitly recovers the trust of temporarily malfunctioning nodes; hence, the trust-update factor calculated for each node reflects the realistic behavior of the node. A node is classified and isolated as malicious if its trust falls below the MAT. The new scheme was evaluated in a number of MATLAB simulations, and its performance was compared with those of other schemes proposed in the literature. Specifically, we determined the DR and MDR for different probabilities P_m of detecting malicious nodes. The WT-MND scheme outperformed the other schemes, achieving a higher DR and a much lower MDR over the range of P_m's. The lowest improvements of WT-MND over the other schemes were 2.6% for DR and 31.6% for MDR.

We also checked the scheme's scalability by investigating the effect of node density on the DR and MDR. As the number of nodes increased, the DR and MDR variations were lower in the WT-MND scheme than in the other tested algorithms, confirming the superior scalability of WT-MND.

We then investigated the response time of the system for varying probabilities of detecting malicious nodes P_m. The scheme detected and isolated the malicious nodes in the network within an acceptably low number of rounds. Finally, we examined the power consumption of the WT-MND scheme by comparing the network lifetimes in WT-MND and the traditional LEACH protocol. The results confirmed that the proposed scheme adds no significant power consumption penalty.

5.1 Challenges and Assumptions

The study faced a number of challenges and limitations during the implementation of the proposed scheme which led to formulation of assumptions as presented below. Despite the apparent benefits associated with MATLAB including features such as an agile computational engine, quick prototyping, rich computation and visualization, MATLAB's capabilities are restricted as it is in want of a built-in routine for WSNs. This deficiency, therefore, mandated that we design our own simulations for the proposed scheme. As regards the presence of false positives in certain clusters in which compromised nodes' number exceeded that of legitimate nodes, the system misidentified normal and malicious nodes, mistaking one for the other, triggering a hike in the ratio of mis-detected nodes. The assumptions included:

- The BS cannot be compromised by an adversary otherwise the attacker can launch any possible attack against the WSN upon taking control of the BS.

- The optimal number of frames in each round, which minimizes the tradeoff between the lifetime and the throughput of the network, is five. In each frame, the number of TDMA slots equals the number of CM nodes.

- Each CM transmits its data in the corresponding time slot of each frame.

- Each node has a trust vector (TV) containing the DT values of all *N* nodes in the network. Each node fully trusts itself. Initially, all entries of the TV are set to 1.

- The communication path over which the sensed values are propagated from the CM to the CH and then to the BS is considered to be error-free so the data reaches to the BS without modification en route.

- The CH in each cluster works as a *monitoring* node that monitors the behaviors of its CM nodes, calculates their IT values, and decides whether to admit or to isolate each node. The calculated IT value of CM_i defines the reputation of CM_i with respect to all other CMs.

5.2 Future Works

Our approach assumes that the BS and CHs can be fully trusted. In fact, if an adversary gains control over the BS and/or CHs, the WSN becomes vulnerable to attacks. This problem is pertinent and warrants further investigation. In future work, we will examine the WT-MND scheme taking into consideration geographic distances (i.e. closer nodes may be trusted more) in heterogeneous networks subjected to powerful security attacks such as Sybil attacks, wormhole attacks, Hello flooding attacks, jamming attacks, and selective forwarding attacks.

REFERENCES

[1] Butun I, Morgera SD, Sankar RA. A Survey of intrusion detection systems in wireless sensor networks. IEEE Commun Surv Tutorials 2014; 16(1):266-282.

[2] Illiano VP, Lupu EC. Illiano, and E. Lupu, Detecting malicious data injections in event detection wireless sensor networks," IEEE Trans Netw Serv Manage 2015; 12(3):496-510.

[3] Makin BA, Padha DA. "A Trust-Based Secure Data Aggregation Protocol for Wireless Sensor Networks," the IUP. J Inform Technol 2010; 6(3):7-22.

[4] Almomani I, B. Al-Kasasbeh, and M, AL-Akhras, "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks," Hindawi Publishing Corporation Journal of Sensors. 2016; 2016:1-16.

[5] Wang J, Jiang S, Fapojuwo AO. A protocol layer trust-based intrusion detection scheme for wireless sensor networks. Sensors 2017; 17(6).

[6] Hsieh M, Huang Y, Chao H. Adaptive security design with malicious node detection in cluster-based sensor networks. Comput Commun 2007; 30(11–12):2385-2400.

[7] Idris MA, Hongbing H, Yu C. Wei-Shinn, K. and Zhou, S.: malicious node detection in wireless sensor networks using weighted trust evaluation.

Proceedings of the Symposium on Simulation of Systems Security (SSSS'08), Ottawa, Canada, 2008:836-843.

[8] Rani, JayaKumar, Dhivya R, Towards Malicious node Detection Methods in Wireless Sensor Network. International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE), 2016, 0976-1353, 22 (2), 69-73.

[9] Ju L, Li H, Liu Y, Xue W, Li K, Chi Z. An Improved Intrusion Detection Scheme based on Weighted Trust Evaluation for Wireless Sensor Networks. Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications (CUTE), China, 2010:978-983.

[10] Chen Z, Tian L, Lin C. "Trust model of wireless sensor networks and its application in Data Fusion". Sensors 2017; 17(4):703.

[11] Abu Romman A, Al-Bahadili H. Performance analysis of the neighbor weight trust determination algorithm in Manets. International Journal Network Security & Its Applications (IJNSA) 2016; 8(4).

[12] Ahmed, K. Abubakar, M. Channa, K. Haseeb, and A. Khan, "A Survey on Trust Based Detection and Isolation of Malicious Nodes in Ad-Hoc and Sensor Networks," Frontiers of Computer Science Journal, Vol.9, No. 2, pp. 280-296, 2015.

[13] D. Martins and H. Guyennet, "Wireless Sensor Network Attacks and Security Mechanisms: A Short Survey". Proceedings of the 13th International Conference

on Network-Based Information Systems (NBiS), 14-16 September 2010, pp. 313-320, 2010.

[14] K. Xing, S. Srinivasan, M Rivera, J. Li, and X. Cheng, "Attacks and Countermeasures in Sensor Networks: A Survey". Book Chapter in Network Security, pp. 251-272, Springer, New York, 2010.

[15] Sajjad SM, Bouk SH, Yousaf M. Neighbor node trust based intrusion detection system for WSN. Procedia Comput Sci 2015; 63:183-188.

[16] Potukuchi RV, Kant K. Secure data aggregation and intrusion detection in wireless sensor networks. International Conference on Signal Processing and Communication (ICSC); 2015:127-131.

[17] Oh SH, Hong CO, Choi YA. A malicious and malfunctioning node detection scheme for wireless sensor networks. Wireless Sensor Netw, 2012; 4(3):84-90.

[18] Yim S, Choi Y. Neighbor-based malicious node detection in wireless sensor networks. Wireless Sensor Netw, 2015; 4(9):219-225.

[19] Ghugar U, Pradhan J, Bhoi S K, Sahoo R R. LB-IDS: Securing Wireless Sensor Network Using Protocol Layer Trust-Based Intrusion Detection System. Hindawi, Journal of Computer Networks and Communications, 2019; 2054298: 13.

[20] Ramya D, Basith P. Design of an efficient Weighted Trust Evaluation System for Wireless Sensor Networks. International Journal Engineering and Computer Science 2014; 3(2):3909-3913.

[21] Babu SS, Raha A, Naskar MK. Trust evaluation based on node's characteristics and neighboring nodes' recommendations for WSN. Wireless Sensor Network 2014; 06:157-172.

[22] Kumar M, Dutta K. LDAT: LFTM based data aggregation and transmission protocol for wireless sensor networks. J Trust Manage 2016; 3(1):2.

[23] Fu JS, Liu Y. Double cluster heads model for secure and accurate data fusion in wireless sensor networks. Sensors 2015; 15(1):2021-2040.

[24] Chen Z, Tian L, Lin C. Trust model of wireless sensor networks and its application in Data Fusion. Sensors 2017; 17(4):703.

[25] Zawaideh F, Salamah M, Al-Bahadili H. A fair trust-based malicious node detection and isolation scheme for WSNs. Processes & Systems (IT-DREPS). 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy 2017, 2017.

[26] Abbasi AA, Younis M. A survey on clustering algorithms for wireless sensor networks. Comput Commun 2007; 30(14–15):2826-2841.

[27] Heinzelman WR, Chandrakasan A, Balakrishnan H. Proceedings of the 33rd Hawaii International Conference on System Sciences. 8; 2000.

[28] More A, Raisinghani V. A survey on energy efficient coverage protocols in wireless sensor networks. J King Saud Univ Comput Inform Sci 2017; 29(4):428-448.

[29] Zeljko G, Dejan S, Overview of DOS attacks on wireless sensor networks and experimental results for simulation of interference attacks. Ingenieria Investigacion, 2018, 38 (1), 130-138.

[30] Liu X. A survey on clustering routing protocols in wireless sensor networks. Sensors 2012; 12(8):11113-11153.

[31] Li Y, N, Yu WZ, Zhao W, You X, Daneshmand M. Enhancing the performance of LEACH protocol in wireless sensor networks. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2011:223-228.

[32] Shang F, Lei Y. An energy-balanced clustering routing algorithm for Wireless Sensor Network. WSN 2010; 2(10):777-783.

[33] Bijone M. A survey on Secure Network: intrusion detection & prevention approaches. Am J Inform Syst 2016; 4(3):69-88.

[34] Ennajari H, Maissa YB, Mouline S. Energy efficient in-network aggregation algorithms in wireless sensor networks: A survey. Advances in Ubiquitous Networking 2. Singapore: Springer 2016; 397:135-148.

[35] Zaman N, Jung LT, Yasin MM. Enhancing energy efficiency of wireless sensor network through the design of energy efficient routing protocol. Hindawi Publishing Corporation. J Sens 2016.

[36] Yu J, Feng L, Jia L, Gu X, and Yu D. A local energy consumption prediction-based clustering protocol for wireless sensor networks. Sensors. 2014; 14(12):23017-23040.

[37] Chang J, Ju P. An efficient cluster-based power saving scheme for wireless sensor networks. EURASIP J Wireless Commun Netw 2012;2012(1):1.

[38] Dattaa U, Mukherjeeb A, Sahuc P and Kundu S. Resource utilization of multi-hop CDMA wireless sensor networks with efficient forwarding protocols. IConDM 2013, Procedia Engineering 64 (2013) 46 – 55.

APPENDICES

Appendix A: Confidence Interval Estimation

In this thesis, parameter DR and MDR estimated through simulation are obtained from outputs of 30 independent simulation runs, each run lasting 10000000 simulation units. The 95% confidence intervals of the estimated means of these parameters for each P_m reported in Figures 12 and Figure 13 are calculated as follows:

$$m \pm t_{29,\,0.025} * \frac{SD}{\sqrt{n}}$$

where n=30 (the number of simulation runs), m is the average and *SD* is the Standard Deviation of *DR* and *MDR* obtained from the 30 runs. $T_{29,\,0.025}$ is the 0.975 quantile of the *t* distribution with 29 degrees of freedom.

Table 13: Analysis of DR and MDR Versus Probability of Malicious Nodes in the Proposed Scheme

P_m	DR	SD	MDR	SD
0.05	0.997	0.000184	0.022	0.000125
0.1	0.987	0.000623	0.029	0.00013
0.15	0.984	0.000810	0.038	3.76E-06
0.2	0.977	0.000531	0.027	0.000783
0.2	0.972	0.000124	0.025	0.00085

The following table shows the 30 independent replications for DR and MDR for P_m=0.05.

Replication	**DR**	**MDR**
1	0.9980	0.02220
2	0.9979	0.02230
3	0.9970	0.02240
4	0.9990	0.02210
5	0.9980	0.02230
6	0.9980	0.02210
7	0.9970	0.02230
8	0.9978	0.02210
9	0.9975	0.02220
10	0.9971	0.02200
11	0.9973	0.02140
12	0.9972	0.02214
13	0.9793	0.02310
14	0.9975	0.02185
15	0.9971	0.02175
16	0.9974	0.02114
17	0.9981	0.02143
18	0.9982	0.02260
19	0.9975	0.02193
20	0.9975	0.02183
21	0.9974	0.02145
22	0.9973	0.02177
23	0.9978	0.02192
24	0.9975	0.02305
25	0.9979	0.02304
26	0.9974	0.02344
27	0.9976	0.02361
28	0.9970	0.02101
29	0.9970	0.02199
30	0.9972	0.02301
Average (m)	**0.9969**	**0.02220**

The sample variance s^2 for all detection ratio, DR is:

$s^2 = (1/(30\text{-}1))*\sum_{i=1}^{30}(m_i - m)^2$ = 1.13103E-05

The Standard deviation SD is:

SD = 0.003363078

Half-size of confidence interval is:

B= $\frac{SD}{\sqrt{30}} * t_{29,0.025}$= 0.00151132

Thus, 95% confidence interval is 0.9969 ± 0.00151132

The sample variance s^2 for all mis-detection ratio, MDR is:

$s^2 = (1/(30-1))*\sum_{i=1}^{30}(m_i - m)^2$ = 4.1E-07

The Standard deviation SD is:

SD = 0.00064

Half-size of confidence interval is:

B= $\frac{SD}{\sqrt{30}} * t_{29,0.025}$= 0.00132164

Thus, 95% confidence interval is 0.02220 ± 0.00132164

Appendix B: Simulation Code

```
% Firas Zawaideh, Firas.zawaideh@final.edu.tr & mr.zawaideh@yahoo.com


clc, clear all, close all

numNodes = 100; % number of nodes
p = 0.1;

netArch  = newNetwork(100, 100, 50, 115);
nodeArch = newNodes(netArch, numNodes);
roundArch = newRound();

plot1

par = struct;

for r = 1:roundArch.numRound
   r
   clusterModel = newCluster(netArch, nodeArch, 'leach', r, p);
   clusterModel = dissEnergyCH(clusterModel, roundArch);
   clusterModel = dissEnergyNonCH(clusterModel, roundArch);
   nodeArch     = clusterModel.nodeArch; % new node architecture after select CHs

   par = plotResults(clusterModel, r, par);
   if nodeArch.numDead == nodeArch.numNode
      break
   end
end



function hh = xlabel(varargin)
%XLABEL X-axis label.
%   XLABEL('text') adds text beside the X-axis on the current axis.
%
%   XLABEL('text','Property1',PropertyValue1,'Property2',PropertyValue2,...)
%   sets the values of the specified properties of the xlabel.
%
%   XLABEL(AX,...) adds the xlabel to the specified axes.
%
%   H = XLABEL(...) returns the handle to the text object used as the label.
%
%   See also YLABEL, ZLABEL, TITLE, TEXT.

%   Copyright 1984-2014 The MathWorks, Inc.

narginchk(1,inf);
```

```
% if the input has an xlabel property which is a text object, use it to set
% the xlabel on.
[ax,args,nargs] = labelcheck('XLabel',varargin);
if isempty(ax)
   ax = gca;
   args = varargin;
end

if nargs > 1 && (rem(nargs-1,2) ~= 0)
  error(message('MATLAB:xlabel:InvalidNumberOfInputs'))
end

string = args{1};
if isempty(string), string=''; end;
pvpairs = args(2:end);

if isappdata(ax,'MWBYPASS_xlabel')
  h = mwbypass(ax,'MWBYPASS_xlabel',string,pvpairs{:});

  %---Standard behavior
else
   h = get(ax,'XLabel');
   set(h,'FontSizeMode','auto',...
      'FontUnitsMode','auto',...
      'FontWeight',get(ax,'FontWeight'),...
      'FontAngle',get(ax,'FontAngle'),...
      'FontName',get(ax,'FontName'));
   set(h, 'String', string, pvpairs{:});

end

if nargout > 0
  hh = h;
end



%clustering architecture (optimal number of nodes per cluster)
function kOpt = clusterOptimum(netArch, nodeArch, dBS)
% calculate the optimum values for number of nodes
%
%   Input:
%      netArch     network model
%      nodeArch    nodes model
%      dBS         length from base station
%   Example:
%      dBS = sqrt(netArch.Sink.x ^ 2 + netArch.Sink.y ^ 2);
%      numClusters     = clusterOptimum(netArch, nodeArch, dBS);
%
%
```

```
    N    = nodeArch.numNode; % number of nodes
    M    = sqrt(netArch.Yard.Length * netArch.Yard.Width);
    kOpt = sqrt(N) / sqrt(2*pi) * ...
           sqrt(netArch.Energy.freeSpace / netArch.Energy.multiPath) * ...
           M / dBS ^ 2;
    kOpt = round(kOpt);
end
```

```
function createfigure(X1, Y1, Y2, Y3)
%CREATEFIGURE(X1,Y1,Y2,Y3)
%  X1:  vector of x data
%  Y1:  vector of y data
%  Y2:  vector of y data
%  Y3:  vector of y data

%  Auto-generated by MATLAB on 31-Jan-2013 18:49:05

% Create figure
figure1 = figure(2);

% Create sub-plot
subplot1 = subplot(2,2,1,'Parent',figure1);
box(subplot1,'on');
hold(subplot1,'all');

% Create plot
plot(X1,Y1,'Parent',subplot1,'LineWidth',2,'Color',[0 1 0]);

% Create x-label
xlabel('Round','FontWeight','bold','FontSize',11,'FontName','Cambria');

% Create y-label
ylabel('sum of energy','FontWeight','bold','FontSize',11,...
    'FontName','Cambria');

% Create title
title('Sum of energy of nodes vs. round','FontWeight','bold','FontSize',12,...
    'FontName','Cambria');

% Create sub-plot
subplot2 = subplot(2,2,2,'Parent',figure1);
box(subplot2,'on');
hold(subplot2,'all');
```

```
% Create plot
plot(X1,Y2,'Parent',subplot2,'LineWidth',2);

% Create x-label
xlabel('Round','FontWeight','bold','FontSize',11,'FontName','Cambria');

% Create y-label
ylabel('# of packets sent to BS nodes','FontWeight','bold','FontSize',11,...
    'FontName','Cambria');

% Create title
title('Number of packet sent to BS vs. round','FontWeight','bold',...
    'FontSize',12,...
    'FontName','Cambria');

% Create sub-plot
subplot3 = subplot(2,2,3,'Parent',figure1);
box(subplot3,'on');
hold(subplot3,'all');

% Create plot
plot(X1,Y3,'Parent',subplot3,'LineWidth',2,'Color',[1 0 0]);

% Create x-label
xlabel('Round','FontWeight','bold','FontSize',11,'FontName','Cambria');

% Create y-label
ylabel('# of dead nodes','FontWeight','bold','FontSize',11,...
    'FontName','Cambria');

% Create title
% title('Number of dead node vs. round','FontWeight','bold','FontSize',12,...
%     'FontName','Cambria');
```

```
%calculating power consumption by CH
function clusterModel = dissEnergyCH(clusterModel, roundArch)
% Calculation of Energy dissipated for CHs
%   Input:
%       clusterModel     architecture of nodes, network
%       roundArch        round Architecture
%   Example:
%       r = 10; % round no = 10
%       clusterModel = newCluster(netArch, nodeArch, 'def', r);
%       clusterModel = dissEnergyCH(clusterModel);
```

```
%

    nodeArch = clusterModel.nodeArch;
    netArch  = clusterModel.netArch;
    cluster  = clusterModel.clusterNode;

    d0 = sqrt(netArch.Energy.freeSpace / ...
            netArch.Energy.multiPath);
    if cluster.countCHs == 0
        return
    end
    n = length(cluster.no); % Number of CHs
    ETX = netArch.Energy.transfer;
    ERX = netArch.Energy.receive;
    EDA = netArch.Energy.aggr;
    Emp = netArch.Energy.multiPath;
    Efs = netArch.Energy.freeSpace;
    packetLength = roundArch.packetLength;
    ctrPacketLength = roundArch.ctrPacketLength;
    frameNum=1;   % Number of Frames in each round
    for i = 1:n
        chNo = cluster.no(i);
        distance = cluster.distance(i);
        energy = nodeArch.node(chNo).energy;
        % energy for aggregation the data + energy for transferring to BS
        if(distance >= d0)
            nodeArch.node(chNo).energy = energy - ...
                ((ETX+EDA) * (frameNum*packetLength) + Emp *
(frameNum*packetLength) * (distance ^ 4));
        else
            nodeArch.node(chNo).energy = energy - ...
                ((ETX+EDA) * (frameNum*packetLength) + Efs *
(frameNum*packetLength) * (distance ^ 2));
        end
        nodeArch.node(chNo).energy = nodeArch.node(chNo).energy - ...
            ctrPacketLength * ERX * round(nodeArch.numNode /
clusterModel.numCluster);
    end


    clusterModel.nodeArch = nodeArch;
end




%calculating the power consumption by Cluster members
function clusterModel = dissEnergyNonCH(clusterModel, roundArch)
% Calculation of Energy dissipated for CHs
%   Input:
```

```
%       clusterModel     architecture of nodes, network
%       roundArch        round Architecture
%   Example:
%       r = 10; % round no = 10
%       clusterModel = newCluster(netArch, nodeArch, 'def', r);
%       clusterModel = dissEnergyCH(clusterModel);
%

    nodeArch = clusterModel.nodeArch;
    netArch  = clusterModel.netArch;
    cluster  = clusterModel.clusterNode;
    if cluster.countCHs == 0
       return
    end
    d0 = sqrt(netArch.Energy.freeSpace / ...
            netArch.Energy.multiPath);
    ETX = netArch.Energy.transfer;
    ERX = netArch.Energy.receive;
    EDA = netArch.Energy.aggr;
    Emp = netArch.Energy.multiPath;
    Efs = netArch.Energy.freeSpace;
    packetLength = roundArch.packetLength;
    ctrPacketLength = roundArch.ctrPacketLength;

    locAlive = find(~nodeArch.dead); % find the nodes that are alive
    frameNum=1;   % Number of Frames in each round
    for i = locAlive % search in alive nodes
       %find Associated CH for each normal node
       if strcmp(nodeArch.node(i).type, 'N') &&  ...
          nodeArch.node(i).energy > 0

          locNode = [nodeArch.node(i).x, nodeArch.node(i).y];
          countCH = length(clusterModel.clusterNode.no); % Number of CHs
          % calculate distance to each CH and find smallest distance
          [minDis, loc] = min(sqrt(sum((repmat(locNode, countCH, 1) - cluster.loc)' .^
2)));
          minDisCH =  cluster.no(loc);

          if (minDis > d0)
             nodeArch.node(i).energy = nodeArch.node(i).energy - ...
                ctrPacketLength * ETX + Emp * (frameNum*packetLength) * (minDis
^ 4);
          else
             nodeArch.node(i).energy = nodeArch.node(i).energy - ...
                ctrPacketLength * ETX + Efs * (frameNum*packetLength) * (minDis ^
2);
          end
          %Energy dissipated
          if(minDis > 0)
```

```
            nodeArch.node(minDisCH).energy = nodeArch.node(minDisCH).energy -
...
                ((ERX + EDA) * packetLength );
          end
        end % if
    end % for
    clusterModel.nodeArch = nodeArch;
end




%clustering (CH election)
function clusterModel = newCluster(netArch, nodeArch, ...
                    clusterFun, clusterFunParam, p_numCluster)
% Create the network architecture with desired parameters
%
%   Input:
%       clusterFun          Function name for clustering algorithm.
%       clusterFunParam     Parameters for the cluster function
%       numCluster          Number of clusters (CHs)
%       netArch             Network model
%       nodeArch            Nodes model
%   Example:
%       clusterModel = newCluster();
%
    % set the parameters
    if ~exist('clusterFun','var')
        clusterFun = 'leach'; % default for clustering the node is leach algorithm
    end
    if strcmp(clusterFun, 'def')
        clusterFun = 'leach'; % default for clustering the node is leach algorithm
    end
    clusterModel.clusterFun = clusterFun;

    if ~exist('clusterFunParam','var')
        clusterFunParam = [];
    end
    clusterModel.clusterFunParam = clusterFunParam;

    if ~exist('netArch','var')
        netArch = newNetwork();
    end
    clusterModel.netArch = netArch;

    if ~exist('nodeArch','var')
        nodeArch = newNodes();
    end
    clusterModel.nodeArch = nodeArch;
```

```
    if ~exist('p_numCluster','var')
        dBS       = sqrt((netArch.Sink.x - netArch.Yard.Length) ^ 2 + ...
                         (netArch.Sink.y - netArch.Yard.Width) ^ 2);
        numCluster = clusterOptimum(netArch, nodeArch, dBS);
        p = 1 / numCluster;
    else
        if p_numCluster < 1
            p = p_numCluster;
            numCluster = 1 / p;
        else
            numCluster = p_numCluster;
            p = 1 / numCluster;
        end
    end
    %p = Optimal Election Probability of a node to become cluster head
    clusterModel.numCluster = numCluster;
    clusterModel.p          = p;

    % run the clustering algorithm
    addpath Cluster % put the clustering algorithm in the cluster folder
    [nodeArch, clusterNode] = feval(clusterFun, clusterModel, clusterFunParam); %
execute the cluster function

    clusterModel.nodeArch = nodeArch;       % new architecture of nodes
    clusterModel.clusterNode = clusterNode; % the CHs
end




%calculation of power consumption
function NetArch = newNetwork(Length, Width, sinkX, sinkY, initEnergy...
    , transEnergy, recEnergy, fsEnergy, mpEnergy, aggrEnergy)
% Create the network architecture with desired parameters
%
%   Input:
%       Length      Length of the yard
%       Width       Width of the yard
%       sinkX       x coordination of base station
%       sinkY       y coordination of base station
%       initEnergy  Initial energy of each node
%       transEnergy Energy for transferring of each bit (ETX)
%       recEnergy   Energy for receiving of each bit (ETX)
%       fsEnergy    Energy of free space model
%       mpEnergy    Energy of multi path model
%       aggrEnergy  Data aggregation energy
%   Example:
%       NetArch = createNetwork();
%
    %%%% Create the yard
```

```
Yard.Type = 'Rect'; % Rectangular
if ~exist('Length','var')
   Yard.Length = 100; % default of the yard is 100 in x coordination
else
   Yard.Length = Length;
end
if ~exist('Width','var')
   Yard.Width = 100; % default of the yard is 100 in y coordination
else
   Yard.Width = Width;
end

%%%% Create base station
% x and y Coordinates of the base station
% default of the base station is in the centre of the yard
if ~exist('sinkX','var')
   Sink.x = Yard.Length / 2;
else
   Sink.x = sinkX;
end
if ~exist('sinkY','var')
   Sink.y = Yard.Width / 2;
else
   Sink.y = sinkY;
end

%%%% Energy Model (all values in Joules)
% Initial Energy
if ~exist('initEnergy','var')
   Energy.init = 0.5;
else
   Energy.init = initEnergy;
end

% Energy for transferring of each bit (ETX)
if ~exist('transEnergy','var')
   Energy.transfer = 50*0.000000001;
else
   Energy.transfer = transEnergy;
end
if ~exist('recEnergy','var')
   Energy.receive = 50*0.000000001;
else
   Energy.receive = recEnergy;
end

% Transmit Amplifier types
if ~exist('recEnergy','var')
   Energy.freeSpace = 10*0.000000000001;
else
```

```
    Energy.freeSpace = fsEnergy;
  end
  if ~exist('recEnergy','var')
    Energy.multiPath = 0.0013*0.000000000001;
  else
    Energy.multiPath = mpEnergy;
  end

  %Data Aggregation Energy
  if ~exist('recEnergy','var')
    Energy.aggr = 5*0.000000001;
  else
    Energy.aggr = aggrEnergy;
  end

  NetArch = struct('Yard',   Yard, ...
                   'Sink',   Sink, ...
                   'Energy', Energy);
end



%node architecture (normal and malicious nodes) number of nodes
%initial trust value .. type of dead node of isolated node of malicious
%node
function nodeArch = newNodes(netArch, numNode)
% Create the node model randomly
%
%   Input:
%       netArch     Network architecture
%       numNode     Number of Nodes in the field
%   Output:
%       nodeArch    Nodes architecture
%       nodesLoc    Location of Nodes in the field
%   Example:
%       netArch  = createNetwork();
%       nodeArch = createNodes(netArch, 100)
%
MaliciousRatio=0.1;

  if ~exist('netArch','var')
    netArch = newNetwork();
  end

  if ~exist('numNode','var')
    numNode = 100;
  end
  for i = 1:numNode
    % x coordination of node
```

```
        nodeArch.node(i).x      =   rand * netArch.Yard.Length;
        nodeArch.nodesLoc(i, 1) =   nodeArch.node(i).x;
        % y coordination of node
        nodeArch.node(i).y      =   rand * netArch.Yard.Width;
        nodeArch.nodesLoc(i, 2) =   nodeArch.node(i).y;
        % the flag which determines the value of the indicator function? Ci(t)
        nodeArch.node(i).G      =   0;
        % initially there are no cluster heads, only nodes
        nodeArch.node(i).type   =   'N'; % 'N' = node (nun-CH)
        nodeArch.node(i).energy =   netArch.Energy.init;
        nodeArch.node(i).CH     = -1; % number of its CH ?
        nodeArch.dead(i)        = 0; % the node is alive
    end
    for i=1:numNode
        for j=1:numNode
         nodeArch.node(i).Trustnode(j)  = 1; % Values of initial trust
        end
    end


    nodeArch.numNode = numNode; % Number of Nodes in the field
    nodeArch.numDead = 0; % number of dead nodes
    maliciousNodes=round(1+rand([1 MaliciousRatio*numNode])*(99)); % Assign
Malicious Nodes
    for i=1:size(maliciousNodes,2)
        nodeArch.node(maliciousNodes(i)).Malicious=1;
    end

      for i=1:numNode
         nodeArch.node(i).Isolation  = 1; % If the node is isolated by trust algoritm (-1).
if it is available (1).
      end
end




%packet length from CH to BS .. packet length from Node to CH .. stoping round ..
function NetRound = newRound(numRound, packetLength, ctrPacketLength)
% Create the round architecture for specific parameters
%
%   Input:
%       numRound          Number of rounds
%       packetLength      Length of packet that sent for CH to BS
%       ctrPacketLength    Length of packet that sent for nodes to CH
%   Example:
%       NetRound = newRound();

    if ~exist('numRound','var')
        NetRound.numRound = 3000; % default of the maximum round is 9999
```

```
    else
        NetRound.numRound = numRound;
    end
    if ~exist('packetLength','var')
        NetRound.packetLength = 6400; % default of the packet length is 6400
    else
        NetRound.packetLength = packetLength;
    end
    if ~exist('ctrPacketLength','var')
        NetRound.ctrPacketLength = 200;
    else
        NetRound.ctrPacketLength = ctrPacketLength;
    end
end


function par = plotResults(clusterModel, r, par)
    nodeArch = clusterModel.nodeArch;
    netArch = clusterModel.netArch;


    %%%%% number of packets sent from CHs to BS
    if r == 1
        par.packetToBS(r) = clusterModel.numCluster;
    else
        par.packetToBS(r) = par.packetToBS(r-1) +
clusterModel.clusterNode.countCHs;
    end
    % Figure packet to BS
%     fig(par.packetToBS, r, 1, '# of packets sent to BS nodes', ...
%         'Number of packet sent to BS vs. round');

    %%%%% Number of dead neurons
    par.numDead(r) = nodeArch.numDead;
    % Figure number of dead node
%     fig(par.numDead, r, 2, '# of dead nodes', 'Number of dead node vs. round');

    %%%%% Energy
    par.energy(r) = 0;
    node = clusterModel.nodeArch;
    for i = find(~node.dead)
        if node.node(i).energy > 0
            par.energy(r) = par.energy(r) + node.node(i).energy;
        end
    end
%     fig(par.energy, r, 3, 'sum of energy', 'Sum of energy of nodes vs. round');
    save('doublefig.mat','par')
createfigure(1:r, par.energy, par.packetToBS, par.numDead);
end
%MAT value .. recovery after 10 rounds .. Pn-err .. Pm-err .. Pm .. Pn
```

```
function clusterModel=WeightCalc( clusterModel,r )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
    NumberOfFrames=4;
    nodeArch = clusterModel.nodeArch;
    netArch  = clusterModel.netArch;
    cluster  = clusterModel.clusterNode;

MAT =0.7; % Minimum Acceptable Trust.
resetVariable=10;  % Reset Value of trust vectore.
pn=0.04;%round(rand(1),1); % Ratio of normal nodes to send malicious data.
pnerr= nodeGen(pn,clusterModel); % Generate randorm node
pm=round(rand(1),1); % Ratio of malicious nodes to send malicious data.
pmerr =0.85;%rand(1); % Probability for Malicious node to send maliciuos data
mnum=0.1; % Number of malicius nodes.

% Assign nodes to be malicious.
if r==1
    for i=1:nodeArch.numNode
      nodeArch.node(i).Malicious=0;
    end
w=round(1+rand([1 mnum*nodeArch.numNode])*(nodeArch.numNode-1)); %
Randomly generate node IDs to be Malicious
for i=1:size(w,2)
    if nodeArch.node(w(1,i)).type ~='C'
    nodeArch.node(w(1,i)).Malicious=1;
    figure(1);
    p1= plot(nodeArch.node(w(1,i)).x,nodeArch.node(w(1,i)).y,'x',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','g',...
    'MarkerFaceColor',[0.5,0.5,0.5]);
    end
end
save('Malicious.mat','w')
else
    load Malicious.mat
end



%% Normal Nodes Direct Trust Calulations
for i=1:pn*nodeArch.numNode
NumberOfErrorFrames=round(1+rand(1)*(1)); % Vector broadcasted by CH.
alpha= 1-(NumberOfErrorFrames/(3*NumberOfFrames));
if nodeArch.node(pnerr(i)).Malicious ==0
   TrustUpdate= alpha*nodeArch.node(pnerr(i)).Trustnode(pnerr(i));
   for n=1:nodeArch.numNode  % Update the node trust values in all othe nodes.
    nodeArch.node(n).Trustnode(pnerr(i)) =  TrustUpdate;
   end
```

```
end
end

%% %% Malicious Nodes Direct Trust Caclulations
for i=1:round(pmerr*size(w,2))

NumberOfErrorFrames=round(2+rand(1)*(2)); % Vector broadcasted by CH.
alpha= 1-(NumberOfErrorFrames/(3*NumberOfFrames));
if nodeArch.node(w(1,i)).Malicious ==1
    w(1,i);
   TrustUpdate= alpha*nodeArch.node(w(1,i)).Trustnode(w(1,i));
   for n=1:nodeArch.numNode  % Update the node trust values in all othe nodes.
    nodeArch.node(n).Trustnode(w(1,i)) =  TrustUpdate;
   end

end
end

%% End of trust calculations
% You need to elect a cluster head and avoide using its trust in the
% quations to make sure it is pure normal node.
sumall=[]; %Denom for Weights eq2
sumall(1)=0;
 for i=2:nodeArch.numNode
       sumnode=[];   % for Equation 1.
       for j=2:nodeArch.numNode
       sumnode=[sumnode,nodeArch.node(j).Trustnode(i)];
       end
       tcm=sum(sumnode)/(nodeArch.numNode-1); % Isolation Equation 1.
       sumall=[sumall,tcm];  %Denom for Weights eq2
 end

weit=[]; % Weights for equation 2
weit(1)=1;
for i=2:nodeArch.numNode
   weit=[weit,sumall(i)/sum(sumall)];
end
save('weit.mat','weit');

%%%%%%Weight Update Factor
% equation (6) of thaita and Kn

Theta=1/((nodeArch.numNode-2)*1.75)


% % Isolation Equation 3
% ITcmi_chSUM=[]; %Denom for Weights eq2
%  ITcmi_chSUM(1)=1;
%  for i=2:nodeArch.numNode
%        ITcmi_ch=[];
```

```
%        ITcmi_ch(1)=1;
%        for j=2:nodeArch.numNode
%        ITcmi_ch=[ITcmi_ch,weit(j)*nodeArch.node(1).Trustnode(i)];
%        end
%        ITSum=sum(ITcmi_ch);
%        ITcmi_chSUM=[ITcmi_chSUM,ITSum];
%  end

 for i=2:nodeArch.numNode
   if weit(i)<Theta
%        if i>2
%        set(p2,'Visible','off')
%        end
      nodeArch.node(i).Isolation=-1;
   figure(1);
   p2=plot(nodeArch.node(i).x,nodeArch.node(i).y,'x',...
   'LineWidth',2,...
   'MarkerSize',10,...
   'MarkerEdgeColor','r',...
   'MarkerFaceColor',[1,1,0.5]);
   end
 end



%% End of trust isolation equations.

   if mod(r,resetVariable)==0    % Reset trust value after each 50 round.
    for i=1:nodeArch.numNode
      for j=1:nodeArch.numNode
       nodeArch.node(i).Trustnode(j)  = 1; % Values of initial trust
      end
   end
   end


   clusterModel.nodeArch = nodeArch;

end

function pnerr= nodeGen(pn,clusterModel)
% This function to not allow the random function to choose the same normal
% within specific number of rounds (resetValue) of trust vectore.

nodeArch = clusterModel.nodeArch;

pnerr =round(1+rand([1 pn*nodeArch.numNode])*(nodeArch.numNode-1)); %
Probability for normal node to send maliciuos data
for i=1:size(pnerr,2)
if nodeArch.node(pnerr(i)).Trustnode(pnerr(i))<1
```

```
      pnerr= nodeGen(pn,clusterModel);
end
end


% for o=1:nodeArch.numNode
%     if nodeArch.node(o).Malicious==1
%         figure(1);
%         plot(nodeArch.node(o).x,nodeArch.node(o).y,'x',...
%     'LineWidth',2,...
%     'MarkerSize',10,...
%     'MarkerEdgeColor','g',...
%     'MarkerFaceColor',[0.5,0.5,0.5])
%     end
%
% end

% for q=1:nodeArch.numNode
%     if nodeArch.node(q).type=='C'
%         figure(1);
%         plot(nodeArch.node(q).x,nodeArch.node(q).y,'bo',...
%     'LineWidth',2,...
%     'MarkerSize',10,...
%     'MarkerEdgeColor','r',...
%     'MarkerFaceColor',[1,1,0.5])
%     else
%         figure(1);
%         plot(nodeArch.node(q).x,nodeArch.node(q).y,'.', 'MarkerSize',15);
%     end
%
% end
end


function fig(data, r, n, yLabel, Title)
% plot data vs. round

   figure(2);
   subplot(1, 3, n);
   plot(1:r, data);
   xlabel('Round');
   ylabel(yLabel);
   title(Title);
end


figure(1), hold on
p1=plot(netArch.Sink.x, netArch.Sink.y,'^', ...
   'MarkerSize',10, 'MarkerFaceColor', 'r','DisplayName','Base Station');
p2=plot(nodeArch.nodesLoc(:, 1), nodeArch.nodesLoc(:, 2),...
```

```
    'b.', 'MarkerSize',15,'DisplayName','Normal Node');

p3= plot(20, 22,'y.', 'MarkerSize',15);
p4= plot(20, 22,'x',...
    'LineWidth',2,...
    'MarkerSize',8,...
    'MarkerEdgeColor','g',...
    'MarkerFaceColor',[0.5,0.5,0.5]);
% p2=plot(0, 0,'x',...
%     'LineWidth',2,...
%     'MarkerSize',10,...
%     'MarkerEdgeColor','r',...
%     'MarkerFaceColor',[1,1,0.5]);



legend ([p1,p2,p3,p4],'Base Station (BS)','Cluster Head (CH)','Normal
Node','Malicious Node')
legend('Location','northeastoutside')
% This commad to draw the legend outside.



function par = plotResults(clusterModel, r, par)
    nodeArch = clusterModel.nodeArch;
    netArch = clusterModel.netArch;


    %%%%% number of packets sent from CHs to BS
    if r == 1
        par.packetToBS(r) = clusterModel.numCluster;
    else
        par.packetToBS(r) = par.packetToBS(r-1) +
clusterModel.clusterNode.countCHs;
    end
    % Figure packet to BS
%     fig(par.packetToBS, r, 1, '# of packets sent to BS nodes', ...
%         'Number of packet sent to BS vs. round');

    %%%%% Number of dead neurons
    par.numDead(r) = nodeArch.numDead;
    % Figure number of dead node
%     fig(par.numDead, r, 2, '# of dead nodes', 'Number of dead node vs. round');

    %%%%% Energy
    par.energy(r) = 0;
    node = clusterModel.nodeArch;
    for i = find(~node.dead)
        if node.node(i).energy > 0
            par.energy(r) = par.energy(r) + node.node(i).energy;
```

```
        end
    end
%     fig(par.energy, r, 3, 'sum of energy', 'Sum of energy of nodes vs. round');

save('doublefig.mat','par')
createfigure(1:r, par.energy, par.packetToBS, par.numDead);
end
```

Appendix C: Related Publication

An Efficient Weighted Trust-Based Malicious Node Detection Scheme for Wireless Sensor Networks

1. Introduction

Wireless sensor networks (WSNs) are distributed networks with a large number of extremely small, inexpensive, low-battery-powered devices called sensor nodes (SNs), which are densely embedded in a physical environment. Owing to their ad hoc nature, distributed architecture, and wireless communication paradigm, WSNs are deployable in a wide assortment of applications, such as environmental monitoring, traffic, healthcare, smart grids, home automation, smart cities, and military applications. The main function of an SN is to sense and report data to the base station (BS) that provides gateway functionality to another network, or an access point to a human interface.

As SNs are often deployed in hostile and unattended environments, they are vulnerable and easily compromised by malicious nodes entering through different types of attacks. Among the more common attacks are Denial of Service attacks, Hello flooding attacks, wormhole attacks, sinkhole attacks, black hole attacks, and Sybil attacks[1-4]. Furthermore, unlike traditional wireless networks, WSNs have very limited computation, communication, power, and memory resources. Therefore, lightweight, energy-efficient security schemes with minimum bandwidth overhead are required to protect WSNs from malicious attacks[5].

The above problems are usually overcome by reputation and trust-based schemes. Reputation describes the observations accumulated by a node about another node's behavior through direct and indirect inspection of its past behaviors, whereas trust is the subjective anticipation of a node about another node's future performance with respect to a specific behavior[6-9]. Therefore, trust can be determined through direct or indirect observation of other

nodes' behaviors. Direct trust (DT) and indirect trust (IT) are determined from direct observations and from the reputations of other nodes, respectively. Reputation and trust-based mechanisms mainly aim to differentiate between normal and abnormal (malicious) nodes in the network and isolate the malicious ones before they can damage the network. A node is trustable if its trust value equals or exceeds the preset minimum acceptable trust (MAT); otherwise, it is called a malicious node and will be isolated. In trust-based mechanisms, only the trusted nodes are accepted as sources of information and data forwarding[10,11].

Most of the weighted trust-based schemes proposed in the literature update the trust value using a constant penalty factor, which undermines the node credibility. Moreover, the trust recovery of those nodes that lost weight due to a transient fault with time was not considered. Subsequently, malicious nodes may be detected after some normal nodes are sacrificed, which can in turn lead to lack of network connectivity and sensing coverage. Motivated with the above arguments, in this paper, we propose an efficient weighted trust-based malicious node detection (WT-MND) scheme that can detect malicious nodes in WSNs. In this scheme, the SNs report their readings to a cluster head (CH) for data aggregation. The node behaviors are realistically treated by accounting for both false-positive and false-negative instances. The proposed scheme is rendered effective by the adaptive trust-update process that implicitly performs trust recovery and computes a different trust-update factor for every node on the basis of its behavior. As the scheme is based on the low energy adaptive clustering hierarchy (LEACH) protocol, the trust of a given node is indirectly computed from its reputation among its cluster members (CMs).

The performance of the new scheme is evaluated in MATLAB simulations. These simulations help investigate the effects of malicious node probability on the detection ratio (DR), misdetection ratio (MDR), and malicious node lifetime. The effect of node density on the DR and MDR is also investigated. In addition, the power consumption is examined by

comparing the network lifetimes under the traditional LEACH protocol and the proposed WT-MND scheme.

The remainder of this paper is structured as follows. Section 2 reviews related research on trust determination and malicious node detection and isolation in WSNs, and Section 3 describes the proposed WT-MND scheme. The simulation environment, performance results, and result analysis are presented in Section 4. Section 5 presents the research conclusions and recommends ideas for future research and development.

2. Literature Review

The security issues of WSNs have been extensively researched in recent years. Owing to their limited resources and inherent nature, these networks are vulnerable to faults and malicious attacks. Hence, malicious nodes must be effectively detected and isolated by an intrusion detection system (IDS) before they can harm the network. Most of the proposed security schemes are based on trust and reputation mechanisms, which add more security options for decision-making in WSNs.

Idris et al.[7] proposed a novel scheme for malicious node detection based on weighted-trust evaluation (WTE). They adopted a hierarchical clustered network topology that reduces the communication overheads between two SNs. Hongbing et al.[8] proposed a novel scheme based on the WTE algorithm for hierarchical WSNs. In order to detect malicious nodes, they appended a weight-recovery value that depends on the node behavior during a specified past period. Ju et al.[9] developed a new mechanism, also based on the WTE algorithm, which protects WSNs from maliciously behaving nodes by updating the weight value assigned to each node.

Sajjad et al.[12] developed an IDS for WSNs based on the trust level of the neighboring nodes. They investigated the network statistics and malicious node behaviors and reported the

successful detection of selective forwarding, Hello flooding, and jamming attacks by their IDS. Accordingly, their scheme effectively isolated the malicious nodes.

Potukuchi and Kant[13] proposed a secure data-aggregation framework for WSNs with a trust monitoring system (TMS) installed at the node level and an IDS at the BS side. Each node in the network evaluates the trust of its neighboring nodes via the TMS system and performs network activities such as CH selection, data aggregation, and reporting to the BS. The BS then analyzes the received data utilizing the IDS and reports any maliciously behaving nodes back to the network.

Oh et al.[14] presented a malicious and malfunctioning node detection scheme using dual-WTE in a hierarchical WSN. Malicious nodes were effectively detected by a weighted-majority voting technique without sacrificing normal nodes that develop transient faults. Yim and Choi[15] presented a neighbor-based malicious node detection scheme for WSNs. Their scheme estimated the trustworthiness of the nodes by their confidence levels and achieved a high detection accuracy with a low false alarm rate.

Ramya and Basith[16] proposed an efficient WTE system for WSNs, which can detect maliciously behaving nodes by a WTE methodology and cooperation among the CHs. Their approach increased the system's efficiency by reducing the memory, energy, and communication overheads from those of other schemes. Babu et al.[17] proposed TENCR, a new trust evaluation method based on the node's quality of service (QoS) characteristics (the trust metrics) and the neighboring nodes' recommendations. The QoS characteristics give the DT of a node, whereas the reputation of the node among its neighbors gives the IT. Their technique efficiently detected the malicious and selfish nodes and admitted only the trustworthy nodes to the routing process.

Kumar and Dutta[18] presented a novel data-aggregation scheme based on a trust and reputation model, which ensures the security and reliability of aggregated data. This scheme applies linguistic fuzzy logic and fuzzy sets to their trust and reputation model, aiding the selection of secure paths from SNs to the BS. Their proposed scheme is more accurate and consumes less energy than various existing schemes.

Fu and Lui[19] proposed a novel cluster-based data fusion model for secure and accurate data fusion in WSNs. Their model integrates a clustering mechanism, trust and reputation arrangement, and data fusion algorithms. After clustering, the trust system selects two CHs for each cluster on the basis of the reputation of its members. Each CH independently performs data fusion. The results are then sent to the BS, where the dissimilarity coefficient is computed. If the coefficient exceeds a preset value, the CHs are assumed to behave maliciously and other CHs are elected. Another trust evaluation model based on a data fusion mechanism in WSNs was proposed by Chen et al.[20]. Their model integrates the data trust (calculated by processing the sensor data), behavioral trust (calculated by monitoring the behavior of nodes when sensing and forwarding the data), and historical trust (updated trust with weighted calculations). Both models aim to ensure the reliability and credibility of the data in WSNs.

Recently, we investigated the performance of a fair-trust-based malicious node detection and isolation (FTMNDI) scheme[21], which periodically updates the trust of the nodes in a flat network by a neighbor-weight trust determination algorithm. The update is based on the node's reputation and assumes a fixed trust-update factor. However, the FTMNDI scheme ignores several major challenges in network security, that is, the malicious behavior of nodes, the network's architecture, and trust recovery issues, which are addressed in the present research. Moreover, as the FTMNDI scheme makes the malicious node already known to all other nodes, it does not compute the DR and/or MDR.

To our knowledge, most of the previously proposed weighted-trust schemes update the weight or trust value depending on the number of maliciously behaving nodes or apply a constant penalty factor. In order to increase the effectiveness and accuracy of such schemes, we here consider the number of error-data sends by each node and hence introduce a dynamic trust-update factor based on the behavior of each SN.

3. The Proposed WT-MND Scheme

In this section, we model the proposed WT-MND scheme on a clustered WSN.

3.1. Network Architecture

This work adopts a well-known clustering mechanism in WSNs called the LEACH protocol, which hierarchically clusters the SNs. As shown in Fig. 1, each cluster consists of several CMs and one CH[22]. The CMs sense the data and transmit them to their CH in a time-division multiple access (TDMA) manner. Each CH then aggregates the data to reduce the data redundancy and transmits them to the BS using the code-division multiple access (CDMA) technique. The LEACH protocol operates through *rounds*, each consisting of two phases: the set-up phase and the steady-state phase. The set-up phase performs three operations: CH selection, cluster formation, and TDMA/CDMA scheduling. The steady-state phase implements the data transmission[23-26].

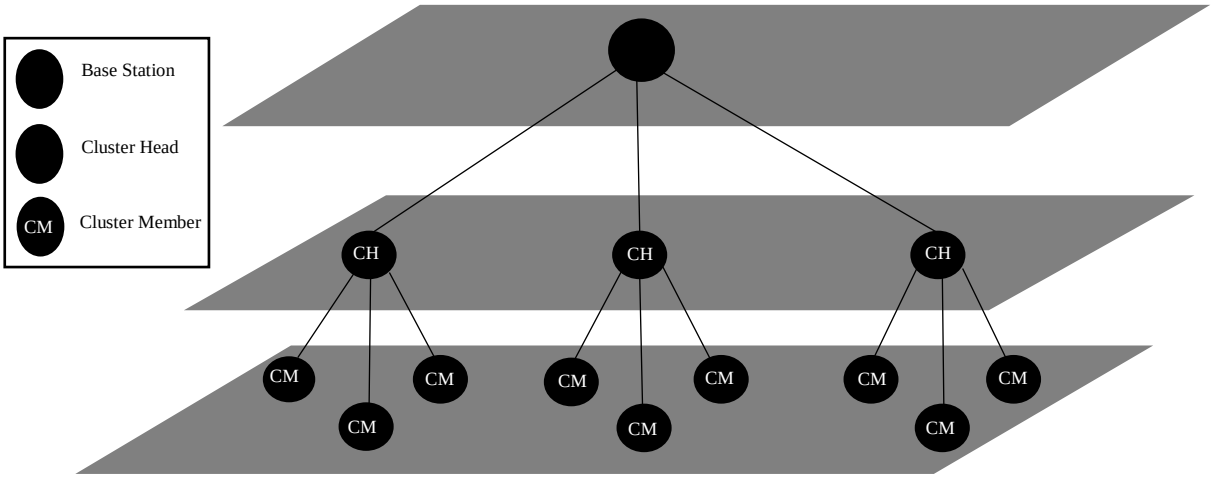

Figure 1: Hierarchical WSN architecture.

3.2. Node Behavior Modeling

An SN can be classified as a normal node or a malicious node depending on its trust value. A normal node usually sends correct data but can behave maliciously when the sensing data are faulty, when the data transmission is interrupted, or when falsified (error) data are sent. On the other hand, a malicious node usually sends falsified data but sometimes transmits correct data to protect itself from detection. For example, suppose that the SNs are sensing the temperature of an environment and transmitting the sensed data to their CHs, which then aggregate the data. Let D denote the weighted average of the temperature sensed by the nodes and δ be a small acceptable variation from D (i.e., let $D \pm \delta$ be accepted in the data readings). When data from a node lie outside this range, they are considered as falsified or error data.

We assume that malicious and normal nodes randomly send falsified data in each TDMA slot with probabilities P_{m_err} and P_{n_err}, respectively. As the malicious nodes are randomly selected at the initialization stage of the simulation, we also define the malicious node probability P_m (i.e., the percentage of malicious nodes among all nodes in the network). Without loss of generality, we assume that the BS behaves nonmaliciously in all rounds and that the CHs do not behave maliciously during their elected rounds.

3.3. Malicious Node Detection and Isolation

The assumptions of the proposed WT-MND scheme are listed below:

1. Recall that a single LEACH round consists of the set-up phase and the steady-state phase. The optimal number of frames in each round, which minimizes the tradeoff between the lifetime and the throughput of the network, is five[27]. Hence, the steady-state phase is assumed to hold five

frames as shown in Fig. 2. In each frame, the number of TDMA slots equals the number of CM nodes.

2. Each CM transmits its data in the corresponding time slot of each frame. However, the last frame of each round is reserved for transmitting the DT values of each CM to its CH. The DT values of the *i*th CM (CM_i) specify CM_i's level of trust in all other CMs in its cluster,where DT value lies between zero and one (zero means no trust and one means fully trusted).

3. Each node has a trust vector (TV) containing the DT values of all *N* nodes in the network. Each node fully trusts itself. Initially, all entries of the TV are set to 1.

4. The CH in each cluster works as a *monitoring* node that monitors the behaviors of its CM nodes, calculates their IT values, and decides whether to admit or to isolate each node. The calculated IT value of CM_i defines the reputation of CM_i with respect to all other CMs, where IT value ranges from zero to one as in the DT case.

5. Each CH maintains a trust-update vector (TUV) containing the current trust-update factors of all of its CMs.

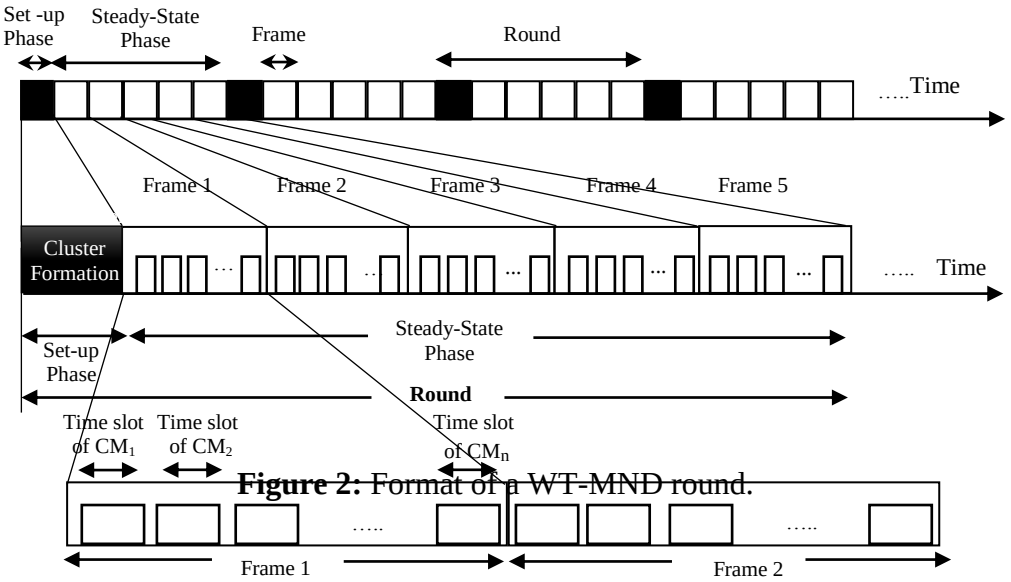

Figure 2: Format of a WT-MND round.

Figure 3 is a functional block diagram of the WT-MND scheme based on the two main phases of the LEACH protocol. The set-up phase was explained in Section 3.1, and the steady-state phase implicitly implements the WT-MND scheme. The WT-MND scheme is structured into four main components: IT calculation, malicious node isolation, trust-update factor computation, and DT updating. These four components are explained in detail below.

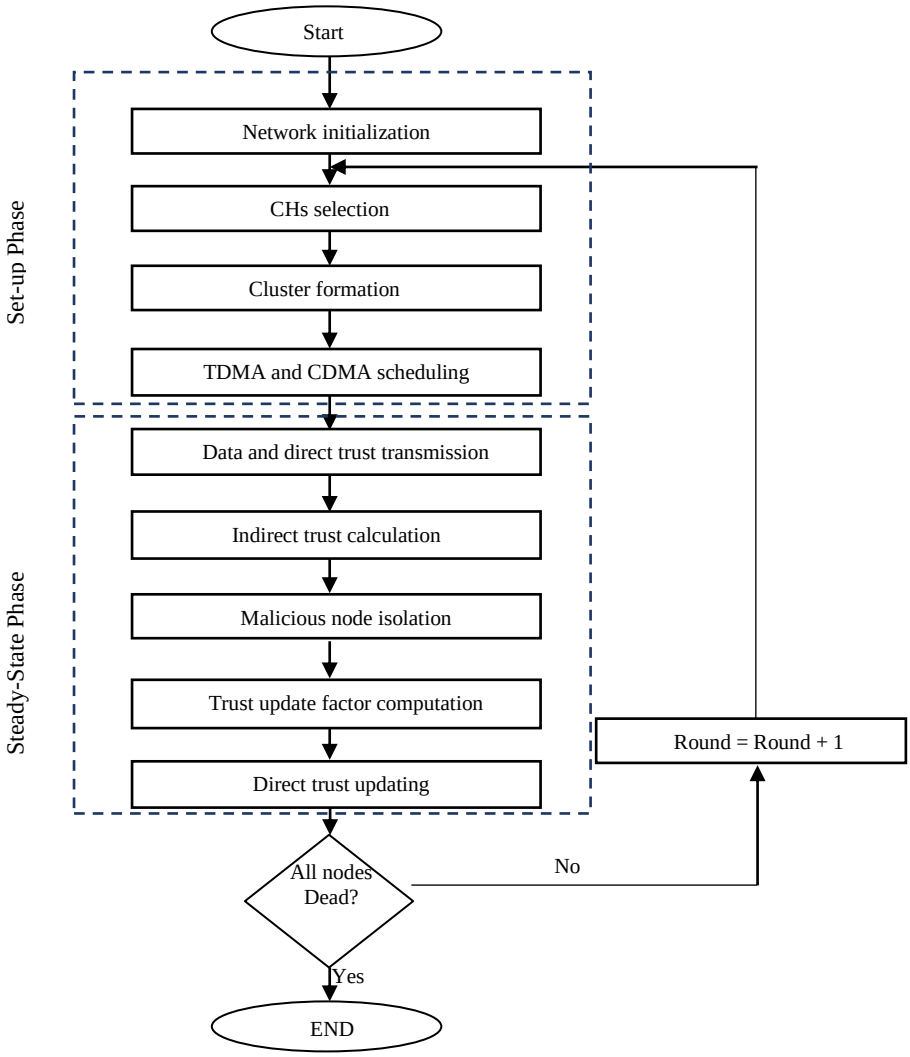

Figure 3: Functional block diagram of the WT-MND scheme.

3.3.1. IT Calculation

In each round, each CM_i transmits its DT values of all other CMs to its CH. The CH in each cluster then calculates the average trust, the weight, and the IT of each CM_i through the following steps.

In order to clarify these steps, we consider one cluster of three CMs ((CM_1, CM_2, CM_3) and one CH. The same steps are applied to all other clusters in the WSN.

(1) The average trust ($\bar{T}$) of each CM_i is determined by all other CMs:

$$\bar{T}_{CM_i} = \frac{1}{k}\sum_{j=1}^{k} RDT_{CM_i.CM_j} \qquad (i = 1 \text{ to } k), \tag{1}$$

where $RDT_{CM_i.CM_j}$ is the reverse DT of CM_i determined by CM_j (i.e., the extent to which CM_i is trusted by CM_j) and k is the number of CMs.

In the above example, this step yields the following result:

$$\bar{T}_{CM_1} = (RDT_{CM_1.CM_1} + RDT_{CM_1.CM_2} + RDT_{CM_1.CM_3})/3,$$

$$\bar{T}_{CM_2} = (RDT_{CM_2.CM_1} + RDT_{CM_2.CM_2} + RDT_{CM_2.CM_3})/3,$$

$$\bar{T}_{CM_3} = (RDT_{CM_3.CM_1} + RDT_{CM_3.CM_2} + RDTT_{CM_3.CM_3})/3.$$

(2) The CH calculates the weight of each CM_i (W_{CMi}), accounting for the average trust of all other CMs. Note that all W_{CMi}'s in a cluster sum to 1, and the individual W_{CMi}'s indicate the reputation of the CM_i's among the other CMs. A normal node is weighted more heavily than a malicious node.

$$W_{CM_i} = \frac{\bar{T}_{CM_i}}{\sum_{j=1}^{k} \bar{T}_{CM_j}} \qquad (i = 1 \text{ to } k). \tag{2}$$

In the above example, W_{CM_i}'s are calculated as follows:

$$W_{CM_1} = \frac{\bar{T}_{CM_1}}{\bar{T}_{CM_1} + \bar{T}_{CM_2} + \bar{T}_{CM_3}},$$

$$W_{CM_2} = \frac{\bar{T}_{CM_2}}{\bar{T}_{CM_1} + \bar{T}_{CM_2} + \bar{T}_{CM_3}},$$

$$W_{CM_3} = \frac{\bar{T}_{CM_3}}{\bar{T}_{CM_1} + \bar{T}_{CM_2} + \bar{T}_{CM_3}}.$$

(3) The CH calculates the IT of each CM_i ($IT_{CM_i.CH}$) as follows:

$$IT_{CM_i.CH} = \sum_{j=1}^{k} W_{CM_j} \cdot RDT_{CM_i.CM_j} \qquad (i = 1 \text{ to } k). \tag{3}$$

In the above example, $IT_{CM_i.CH}$ is given by

$$IT_{CM_1.CH} = W_{CM1}.\ RDT_{CM_1.CM_1} + W_{CM2}.RDT_{CM_1.CM_2} + W_{CM3}.RDT_{CM_1.CM_3},$$

$$IT_{CM_2.CH} = W_{CM1}.RDT_{CM_2.CM_1} + W_{CM2}.RDT_{CM_2.CM_2} + W_{CM3}.RDT_{CM_2.CM_3},$$

$$IT_{CM_3.CH} = W_{CM1}.RDT_{CM_3.CM_1} + W_{CM2}.RDT_{CM_3.CM_2} + W_{CM3}.RDT_{CM_3.CM_3}.$$

3.3.2. Malicious Node Isolation

The CH compares the IT of each CM calculated by Eq. (3) with the preset MAT value. Any node with an IT value below MAT is considered a malicious node and is isolated from the network. Otherwise, the node is considered a normal node and is admitted. It should be noted that MAT lies between 0 and 1, and its value depends on how sensitively the network application must quickly isolate malicious nodes in the network[21].

3.3.3. Trust-Update Factor Computation

As mentioned above, each CM_i in each cluster sends its sensed data to its CH in the first four TDMA frames of each round. The CH aggregates the sensed data, observes the behavior of each CM_i by recording the number of error-data transmissions by CM_i per round [$T_{err}(i)$], and computes the corresponding trust-update factor $\alpha(i)$ by the following heuristic formula:

$$\alpha\ (i) = 1 - \frac{T_{err}(i)}{2*F}\ (i = 1 \text{ to } k), \qquad (4)$$

where F is the number of data frames per round (four in our case).

Hence, at the end of each round, every CH forms its resultant TUV as [$\alpha(1)$, $\alpha(2)$, …, $\alpha(k)$]. It is worth mentioning that α of a suspected malicious node (Section 3.3.2) is replaced by the node's physical address in the TUV.

Clearly, detected falsified information reduces the $\alpha(i)$ value of CM_i. However, this falsified information may simply result from a temporary fault in the communication

channel, which neither compromises nor disables the node. Therefore, continuously decreasing the trust-update values of such nodes is a nonpractical solution. Instead, the DT value should be recovered if the SN functions normally after an interruption. The trust value can be implicitly and adaptively recovered by recalculating the α values of the CMs at each round.

In the above example, if

- CM_1 transmits zero error messages (i.e., $\boldsymbol{T_{err(1)} = 0}$),
- CM_2 transmits one error message due to faulty sensing data (i.e., $\boldsymbol{T_{err(2)=}1}$),
- CM_3 transmits three error messages (i.e., $\boldsymbol{T_{err(3)} = 3}$),

we obtain

$$\alpha\,(1) = \mathbf{1}\text{-}\frac{\boldsymbol{T_{err(1)}}}{\mathbf{2}*\boldsymbol{F}} = \mathbf{1},$$

$$\alpha\,(2) = \mathbf{1}\text{-}\frac{\boldsymbol{T_{err(2)}}}{\mathbf{2}*\boldsymbol{F}} = \mathbf{7/8},$$

$$\alpha\,(3) = \mathbf{1}\text{-}\frac{\boldsymbol{T_{err(3)}}}{\mathbf{2}*\boldsymbol{F}} = \mathbf{5/8},$$

respectively. The corresponding TUV is [1 7/8 5/8].

3.3.4. DT Updating

Based on the TUVs broadcasted by all CHs, each node updates its TV by multiplying its old DT values by the corresponding α values, as shown in the following equation:

$$\mathrm{TV}_{(i)_{\mathrm{updated}}}[j] = \alpha(j) * TV_{(i)_{\mathrm{old}}}[j], (i = 1 \text{ to } N)(j = 1 \text{ to } N). \quad (5)$$

In the following round, the last frame of each CM again transmits the updated DT values of all other CMs to the CH (see Fig. 3). Moreover, in order to ensure that the

normal nodes fairly update their DT values, the TVs are reinitialized after a desired number of rounds (e.g., 40 rounds).

4. Performance Evaluation

4.1. Simulation Setup

The proposed approach was evaluated in a simulated $100 \times 100\ m^2$ network area containing randomly distributed nodes. The nodes were distributed and clustered under the LEACH protocol (see Fig. 4). All CHs and the BS were assumed to behave as normal nodes. All nodes were assumed to be stationary throughout the simulations, and the BS was located at (50,175).

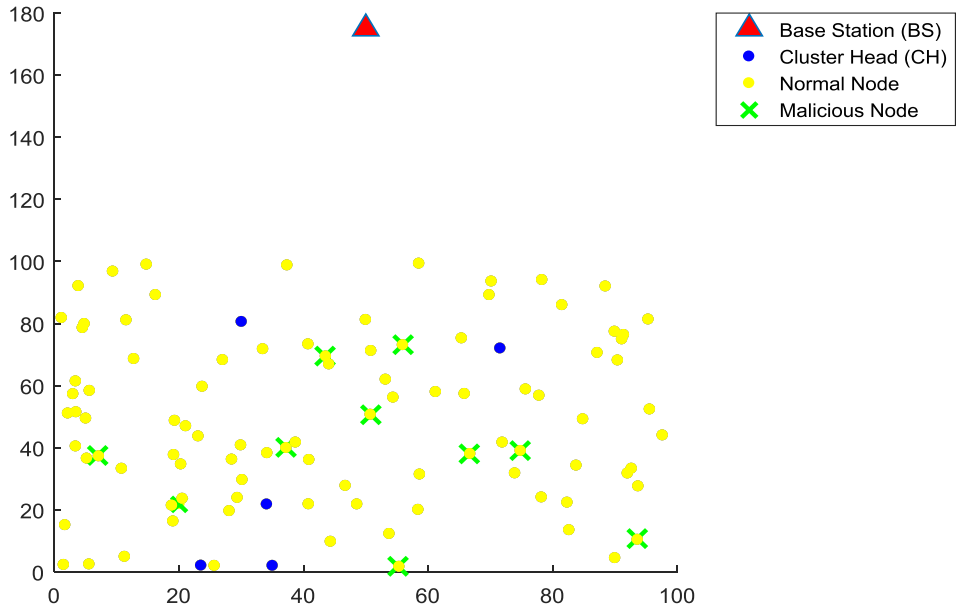

Figure 4: Sample node deployment in the $100 \times 100\ m^2$ network area.

The simulation was carried out over several rounds. During each round, the CHs evaluated the behavior of their CMs and updated the trust level of any maliciously behaving node. When the IT of a node fell below the preset MAT value, that node was isolated. Thus, the malicious node lifetime can be calculated as the number of

rounds before the node is isolated. The input parameters of the simulation are listed in Table 1[10,28].

Table 1: Simulation parameters.

Parameter	Value
Network deployment area	100×100 m^2
Number of SNs, N	100–900
Number of CHs	5% of N
Location of BS	(50,175)
Initial power per node	0.5 J
Length of packets	6400 bits
E_{elec}	50 nJ/bit
Amplifier transmission (ε_{amp})	100 pJ/bit/m^2
Probability of malicious nodes (P_m)	0.05–0.5
Probability of normal nodes transmitting error data ($P_{n\text{-}err}$)	0.05
Probability of malicious nodes transmitting error data ($P_{m\text{-}err}$)	0.6
MAT	0.7

In order to evaluate the simulated WT-MND scheme, we constructed the confusion matrix[29] shown in Table 2 and calculated the DR and MDR performance metrics by Eqs. (6) and (7), respectively.

Table 2: Confusion matrix[29].

		Detected	
		Normal	Malicious
Actual	Normal	True negative (TN)	False positive (FP)
	Malicious	False negative (FN)	True positive (TP)

$$\mathrm{DR} = \frac{Number\ of\ correctly\ detected\ malicious\ nodes}{Total\ number\ of\ malicious\ nodes\ in\ the\ network},$$

$$\mathrm{DR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}. \tag{6}$$

$$\mathrm{MDR} = \frac{\begin{pmatrix}\text{Number ofnormal nodes falsely detected as malicious nodes} \\ + \\ \text{Number of malicious nodes falsely detected as normal nodes}\end{pmatrix}}{\text{Total number of nodes}},$$

$$\mathrm{MDR} = \frac{\mathrm{FP+FN}}{\mathrm{FP+TP+FN+TN}}. \tag{7}$$

The performance of the WT-MND detection scheme was further evaluated by the *response time*, which expresses the average number of rounds before malicious nodes are correctly detected in the network.

The energy consumption of radio communication in the network was estimated by a simplified power control model[23,30-33]. The energy consumption of a node transmitting and receiving k-bit data in free space over a distance of *d* meters was computed by Eqs. (8) and (9), respectively:

$$E_{Tx}(k,d) = E_{\mathrm{elec}} \times k + \varepsilon_{\mathrm{amp}} \times k \times d^2, \tag{8}$$

$$E_{Rx}(k) = E_{\mathrm{elec}} \times k, \tag{9}$$

where E_{elec} and ε_{amp} denote the energy consumption of the transceiver and amplifier electronics per bit, respectively.

As is well known, most of the energy in a WSN is expended in node-to-node data transmission. The residual energy of node *i*, denoted by $E_{\text{residual}}(i)$, is the energy remaining in node *i* after data transmission or reception and/or computation.

Hence, after each round, the average residual energy of all live nodes, *n*, in the system is calculated as follows:

$$E_{\text{average}} = \frac{\sum_{i=1}^{n} E_{\text{residual}}(i)}{n}. \quad (10)$$

4.2. Simulation Results

In order to evaluate the performance of the proposed WT-MND scheme, we varied the network environment and configuration parameters in the MATLAB simulation. Averages of 30 runs were considered to achieve 95% confidence in the results.

In the Byzantine Generals' problem[8,33], the decision of the loyal generals becomes inaccurate when the number of betrayed generals exceeds one-third of the total number. Similarly, if the number of malicious nodes in an IDS exceeds 33% of the total node number, it will become difficult to accuretaly detect the malicious nodes. In order to ensure the safety of the network, we conservatively set the P_m value between 0.04 and 0.25 in the simulation study in a similar manner to that of previous researchers. Further, the detection is terminated after 200 cycles or more than 25% of all the nodes are detected as malicious nodes.

First, the proposed scheme was competed against WTE, WTE(R), and WTA[7-9]. Figures 5 and 7 plot the DRs and MDRs, respectively, as functions of the malicious node probability P_m (0.05–0.25) in a 100-node network of each type. As seen in Fig. 5, the DR of all schemes slightly decreased with increasing P_m. This trend is attributed to the increasing number of malicious nodes as P_m increases, meaning that more falsified information penetrates the network. Consequently, the FN increases while the TP reduces. However, the WT-MND scheme achieved a higher DR than the other methods across the investigated range of P_m's. Note that, to highlight the differences among the four schemes, we have rescaled the y-axis to the range 0.8–1.0. The average DR improvements of WT-MND over WTE, WTE(R), and WTA were 3.6%, 2.8%, and 2.6%, respectively. Moreover, the MDRs were lower in WT-MND than in the other schemes across the range of P_m's (Fig. 6). The average MDR improvements of WT-MND over WTE, WTE(R), and WTA were 89.7%, 31.6%, and 50.3%, respectively.

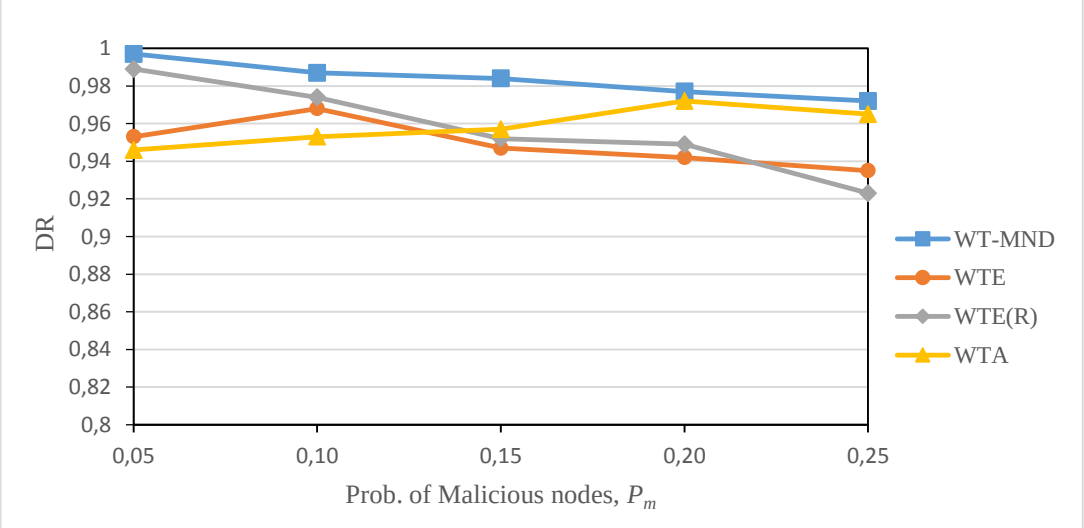

Figure 5: DR versus probability of malicious nodes in the investigated schemes.

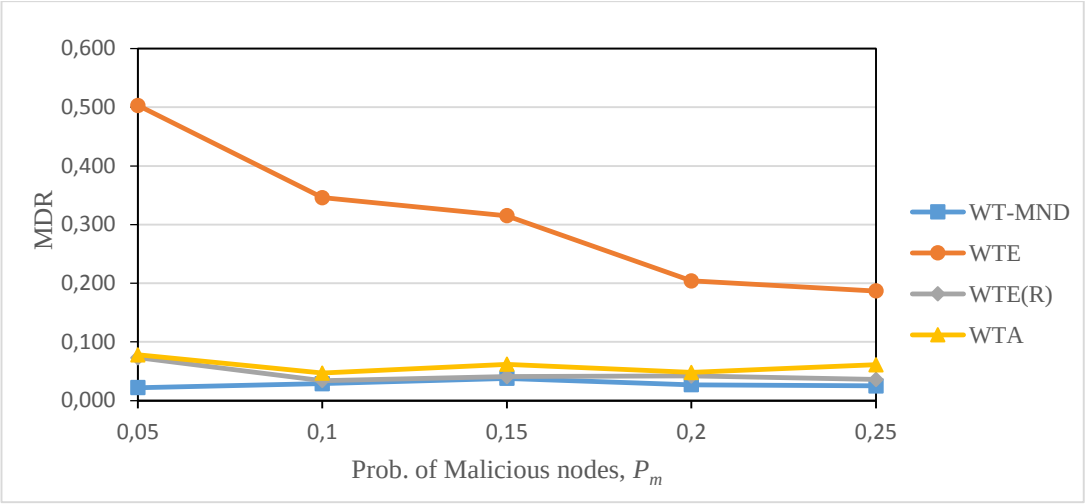

Figure 6: MDR versus probability of malicious nodes in the investigated schemes.

Figure 7 plots the response time versus P_m in small (100-node) and large (900-node) networks. Increasing P_m (i.e., increasing the number of malicious nodes) increased the number of rounds before the malicious nodes were detected. As more malicious nodes appeared, the aggregated data were increasingly affected by the falsified information, and the malicious nodes were harder to detect. This difficulty was amplified in smaller networks, as fewer sources contributed to the reputation of the malicious nodes. However, the results demonstrate that the proposed scheme can detect and isolate malicious nodes within two to four rounds on average.

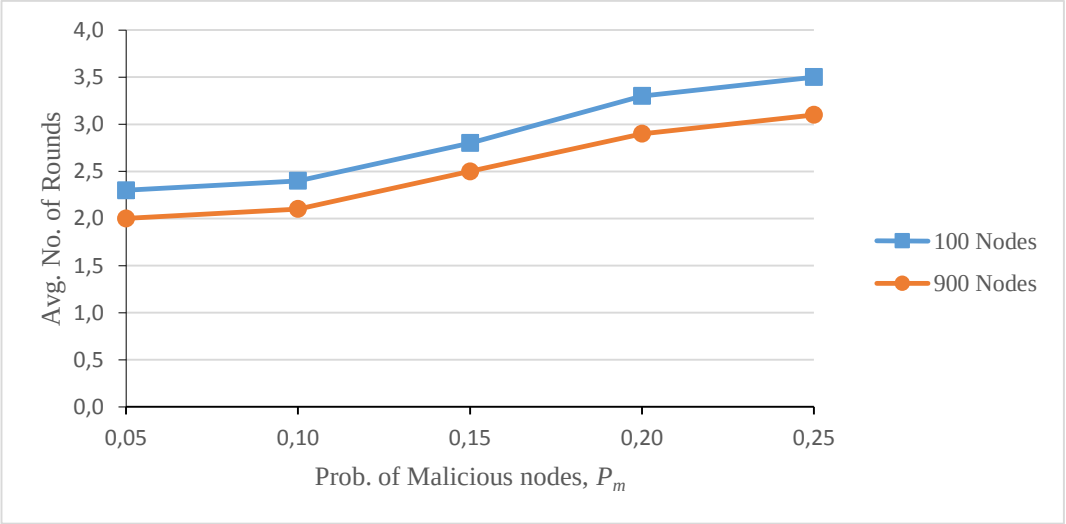

Figure 7: Response time versus probability of malicious node detection in the WT-MND scheme.

Next, in order to compare the scalabilities of the proposed scheme, WTE, WTE(R), and WTA, we computed the DRs and MDRs as functions of node number. Here, the number of nodes was varied from 100 to 900 while P_m was fixed at 0.04. The variations in DR and MDR were quantified by their standard deviations (SDs). The DR and MDR results of the four schemes are shown in Tables 3 and 4, respectively. Although the DR and MDR variations remained small as the node number was varied in all schemes, WT-MND achieved the smallest SDs among the four algorithms, confirming that the WT-MND scheme is more scalable than the other schemes. The average SD of the DR was 83.6%, 61.4%, and 82.3% lower in the WT-MND scheme than in WTE, WTE(R), and WTA, respectively (Table 3). Similarly, the average SD of the MDR was 55.4%, 74.9%, and 71.0% lower in the WT-MND scheme than in WTE, WTE(R), and WTA, respectively (Table 4).

Table 3: DR SD versus number of nodes in the investigated network security schemes.

No. of nodes	100	200	300	400	500	600	700	800	900	SD
Scheme										
WT-MND	0.994	0.989	0.987	0.991	0.986	0.982	0.984	0.988	0.983	0.003887
WTE	0.894	0.979	0.963	0.958	0.957	0.953	0.941	0.958	0.959	0.023659
WTE(R)	0.993	0.989	0.982	0.984	0.982	0.974	0.972	0.967	0.963	0.010064
WTA	0.991	0.978	0.979	0.983	0.974	0.961	0.953	0.938	0.926	0.021953

Table 4: MDR SD versus number of nodes in the investigated network security schemes.

No. of nodes	100	200	300	400	500	600	700	800	900	SD
Scheme										
WT-MND	0.012	0.018	0.026	0.028	0.021	0.032	0.037	0.035	0.031	0.008276
WTE	0.797	0.783	0.779	0.775	0.775	0.775	0.794	0.812	0.827	0.018566
WTE(R)	0.143	0.231	0.208	0.202	0.198	0.195	0.217	0.236	0.261	0.032964
WTA	0.016	0.062	0.055	0.047	0.058	0.083	0.093	0.098	0.104	0.028492

Finally, Figure 8 compares the power consumption overheads of the WT-MND scheme and the traditional LEACH protocol in a 100-node network. Recall that the WT-MND scheme requires some computations and an extra frame per round for transmitting the updated trust values of a CM's associated nodes. This incurs a small overhead; consequently, the residual power and the average residual energy per node dropped by 7% (on average) from those of LEACH.

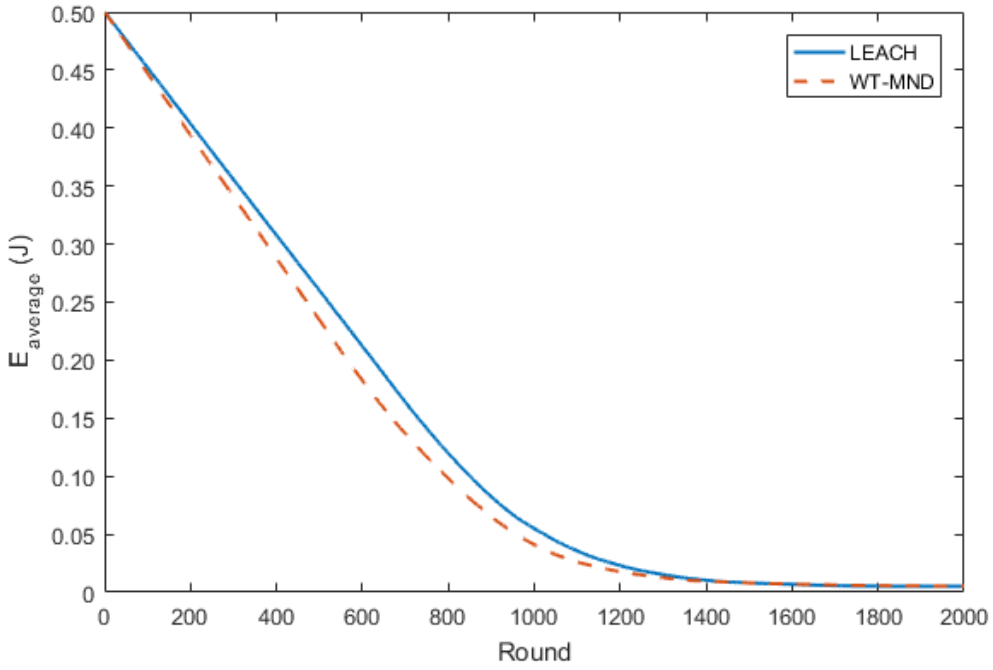

Figure 8: Power consumption comparisons of WT-MND and LEACH.

5. Conclusion

In this paper, we developed and evaluated the performance of an efficient weighted trust-based malicious node detection scheme for WSNs, namely, the WT-MND scheme. Adopting the LEACH clustering protocol, WT-MND determines and updates the trust of the network nodes on the basis of their reputation. The scheme relies on an adaptive trust-update process that implicitly recovers the trust of temporarily malfunctioning nodes; hence, the trust-update factor calculated for each node reflects the realistic behavior of the node. A node is classified and isolated as

malicious if its trust falls below the MAT. The new scheme was evaluated in a number of MATLAB simulations, and its performance was compared with those of other schemes proposed in the literature. Specifically, we determined the DR and MDR for different probabilities P_m of detecting malicious nodes. The WT-MND scheme outperformed the other schemes, achieving a higher DR and a much lower MDR over the range of P_m's. The lowest improvements of WT-MND over the other schemes were 2.6% for DR and 31.6% for MDR.

We also checked the scheme's scalability by investigating the effect of node density on the DR and MDR. As the number of nodes increased, the DR and MDR variations were lower in the WT-MND scheme than in the other tested algorithms, confirming the superior scalability of WT-MND.

We then investigated the response time of the system for varying probabilities of detecting malicious nodes P_m. The scheme detected and isolated the malicious nodes in the network within an acceptably low number of rounds. Finally, we examined the power consumption of the WT-MND scheme by comparing the network lifetimes in WT-MND and the traditional LEACH protocol. The results confirmed that the proposed scheme adds no significant power consumption penalty.

Our approach assumes that the BS and CHs can be fully trusted. In fact, if an adversary gains control over the BS and/or CHs, the WSN becomes vulnerable to attacks. This problem is pertinent and warrants further investigation. In future work, we will examine the WT-MND scheme in heterogeneous networks subjected to powerful security attacks such as Sybil attacks, wormhole attacks, Hello flooding attacks, jamming attacks, and selective forwarding attacks.

References

[1] Butun I, Morgera SD, Sankar RA. A Survey of intrusion detection systems in wireless sensor networks. IEEE Commun Surv Tutorials 2014;16(1):266-282.

[2] Illiano VP, Lupu EC. Illiano, and E. Lupu, Detecting malicious data injections in event detection wireless sensor networks,". IEEE Trans Netw Serv Manage 2015;12(3):496-510.

[3] Makin BA, Padha DA. "A Trust-Based Secure Data Aggregation Protocol for Wireless Sensor Networks," the IUP. J Inform Technol 2010;6(3):7-22.

[4] Almomani I, B. Al-Kasasbeh, and M, AL-Akhras, "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks," Hindawi Publishing Corporation Journal of Sensors. 2016;2016:1-16.

[5] Wang J, Jiang S, Fapojuwo AO. A protocol layer trust-based intrusion detection scheme for wireless sensor networks. Sensors 2017;17(6).

[6] Hsieh M, Huang Y, Chao H. Adaptive security design with malicious node detection in cluster-based sensor networks. Comput Commun 2007;30(11–12):2385-2400.

[7] Idris MA, Hongbing H, Yu C. Wei-Shinn, K. and Zhou, S.: malicious node detection in wireless sensor networks using weighted trust evaluation. Proceedings of the Symposium on Simulation of Systems Security (SSSS'08), Ottawa, Canada, 2008:836-843.

[8] Hongbing H. Weighted trust evaluation-based malicious node detection for wireless sensor networks. Int J Inform Comput Sec, 2009;3(2):132-149.

[9] Ju L, Li H, Liu Y, Xue W, Li K, Chi Z. An Improved Intrusion Detection Scheme based on Weighted Trust Evaluation for Wireless Sensor Networks. Proceedings of the 5th International Conference on Ubiquitous Information Technologies and Applications (CUTE), China, 2010:978-983.

[10] Chen Z, Tian L, Lin C. Trust model of wireless sensor networks and its application in Data Fusion. Sensors 2017;17(4):703.

[11] Abu Romman A, Al-Bahadili H. Performance analysis of the neighbor weight trust determination algorithm in Manets. International Journal Network Security & Its Applications (IJNSA) 2016;8(4).

[12] Sajjad SM, Bouk SH, Yousaf M. Neighbor node trust based intrusion detection system for WSN. Procedia Comput Sci 2015;63:183-188.

[13] Potukuchi RV, Kant K. Secure data aggregation and intrusion detection in wireless sensor networks. International Conference on Signal Processing and Communication (ICSC); 2015:127-131.

[14] Oh SH, Hong CO, Choi YA. A malicious and malfunctioning node detection scheme for wireless sensor networks. Wireless Sensor Netw, 2012;4(3):84-90.

[15] Yim S, Choi Y. Neighbor-based malicious node detection in wireless sensor networks. Wireless Sensor Netw, 2015;4(9):219-225.

[16] Ramya D, Basith P. Design of an efficient Weighted Trust Evaluation System for Wireless Sensor Networks. International Journal Engineering and Computer Science 2014;3(2):3909-3913.

[17] Babu SS, Raha A, Naskar MK. Trust evaluation based on node's characteristics and neighboring nodes' recommendations for WSN. Wireless Sensor Netw 2014;06:157-172.

[18] Kumar M, Dutta K. LDAT: LFTM based data aggregation and transmission protocol for wireless sensor networks. J Trust Manag 2016;3(1):2.

[19] Fu JS, Liu Y. Double cluster heads model for secure and accurate data fusion in wireless sensor networks. Sensors 2015;15(1):2021-2040.

[20] Chen Z, Tian L, Lin C. Trust model of wireless sensor networks and its application in Data Fusion. Sensors 2017;17(4):703.

[21] Zawaideh F, Salamah M, Al-Bahadili H. A fair trust-based malicious node detection and isolation scheme for WSNs. Processes & Systems (IT-DREPS). 2nd International

Conference on the Applications of Information Technology in Developing Renewable Energy 2017, 2017.

[22] Abbasi AA, Younis M. A survey on clustering algorithms for wireless sensor networks. Comput Commun 2007;30(14–15):2826-2841.

[23] Heinzelman WR, Chandrakasan A, Balakrishnan H. Proceedings of the 33rd Hawaii International Conference on System Sciences. 8; 2000.

[24] More A, Raisinghani V. A survey on energy efficient coverage protocols in wireless sensor networks. J King Saud Univ Comput Inform Sci 2017;29(4):428-448.

[25] Zaman N, Jung LT, Yasin MM. Enhancing energy efficiency of Wireless Sensor Network through the design of energy efficient routing protocol. J Sens 2016;2016.

[26] Liu X. A survey on clustering routing protocols in wireless sensor networks. Sensors 2012;12(8):11113-11153.

[27] Li Y, N, Yu WZ, Zhao W, You X, Daneshmand M. Enhancing the performance of LEACH protocol in wireless sensor networks. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2011:223-228.

[28] Shang F, Lei Y. An energy-balanced clustering routing algorithm for Wireless Sensor Network. WSN 2010;2(10):777-783.

[29] Bijone M. A survey on Secure Network: intrusion detection & prevention approaches. Am J Inform Syst 2016;4(3):69-88.

[30] Ennajari H, Maissa YB, Mouline S. Energy efficient in-network aggregation algorithms in wireless sensor networks: A survey. Advances in Ubiquitous Networking 2. Singapore: Springer 2016;397:135-148.

[31] Zaman N, Jung LT, Yasin MM. Enhancing energy efficiency of wireless sensor network through the design of energy efficient routing protocol. Hindawi Publishing Corporation. J Sens 2016.

[32] Yu J, Feng L, Jia L, Gu X, and Yu D. A local energy consumption prediction-based clustering protocol for wireless sensor networks. Sensors. 2014;14(12):23017-23040.

[33] Chang J, Ju P. An efficient cluster-based power saving scheme for wireless sensor networks. EURASIP J Wireless C

A Fair Trust-Based Malicious Node Detection and Isolation Scheme for WSNs

1. Introduction

A WSN is a collection of low power wireless sensor nodes distributed into a network area to monitor and collect data on some physical or environmental parameters (e.g., temperature, sound, vibration, pressure, motion, pollutants, fire, smoke, etc.), and to collaboratively pass the collected data to a pre-defined central node known as the sink node or Base Station (BS) [1]. Due to their flexible structure and superior characteristics, WSNs have been used in a wide range of applications; including: military, industrial, environmental, health, and agricultural applications [2].

The infra-structure-less nature of the WSNs and limited capabilities (processing power, memory, and battery power) of the sensor nodes make them susceptible to various types of attacks. Generally, there are two main types of misbehaving nodes in WSNs: selfish nodes and malicious nodes. A selfish node is a node that does not perform the packet forwarding properly due to low memory, or low battery power, or an intentional effect of an adversary node. On the other hand, the objective of a malicious node is to destruct the network badly as it could perform different types of harmful behavior, such as: Denial-of-Service (DoS) attack, packet dropping, memory overflow, data/route modification, etc. [3]. An extensive survey on WSNs attacks and their classification can be found in [4]. Furthermore, early WSN research assumed a friendly and cooperative WSN environment and focused on problems such as wireless channel access, multi-hop routing, and power consumption, while ignoring network security measures [2-6].

Therefore, it is very essential to have a reliable mechanism for identifying and isolating malicious nodes, before they can do any harm to the network. Traditional security mechanisms such as authentication, encryption and cryptography are not suitable for WSNs as these schemes require extensive computation, communication and storage, which are incompatible with the nature of WSNs. Recently, the concept of trust and reputation has been popularly used to develop security schemes for WSNs. Trust is defined as the confidence and faith of a node in the ability, consistency, and trustworthiness of other nodes. Trust can be determined based on direct or indirect observation of the nodes' behaviors. A trust determined based on direct observation is called direct trust, while that determined based on other nodes observations and opinions is called indirect trust or reputation [7, 8].

A node is called trusted node if it has a trust equal to or larger than a pre-set minimum acceptable trust (MAT); otherwise it is called untrusted or malicious node and will be isolated. In trust-based mechanisms, only trusted nodes are accepted as a source of information and data forwarding [9].

A number of trust determination algorithms have been developed throughout the years [10-14]. One of these algorithms is the Neighbor-Weight Trust Determination (NWTD) algorithm [15, 16]. In this algorithm, each node in the network periodically broadcasts messages containing the IDs and trusts of its one-hop neighbors; therefore, each node may receive different trusts for the same one-hop neighbor from different nodes. The receiving node calculates the average of the received trusts using a weighted-average formula to represent the new trust of the node. The weight here is the weight of the one-hop neighbors. The receiving node participates in the averaging process by giving itself a trust and weight of unity.

In this paper, we modify and utilize the NWTD algorithm to develop and evaluate the performance of a fair trust-based malicious node identification and isolation scheme (FTMI) in flat WSNs. In this scheme, all sensor nodes are initially assumed fully trusted with their trust value set to 1. Then, if the monitoring node notices one of the nodes transmitting falsified information (i.e. a node is behaving maliciously), it identifies it as a true-positive malicious node. Then any malicious node within the network is isolated if its trust becomes equal to or less than a pre-set minimum acceptable trust (MAT). The performance of the FTMI scheme is evaluated through a number of simulations using the network simulator MANSim [17]. These simulations investigate the effect of number of monitoring nodes, trust update factor and minimum acceptable trust on the malicious node lifetime. Using data envelopment analysis, we also found the most dominant factor and the near optimal values of the above parameters.

The rest of this paper is structured as follows. Section 2 reviews some of the most recent research on trust determination and malicious node detection and isolation in WSNs. The NWTD algorithm is explained in section 3. Section 4 presents the detail description of the FTMI scheme. The simulation environment, results and discussions are described in Section 5. Finally, section 6 presents research conclusions and points-out some recommendations for future research and development.

2. Literature Review

Many trust determination models have been developed for peer-to-peer systems [9], which are based on sharing recommendation information to establish trust and reputation. However, most of these models are not applicable for WSNs as they

cause significant network overhead due to the additional information exchanged, and requirement of a trusted third party (or a computationally expensive public key infrastructure (PKI)), which are against the nature of WSNs. Later on, many trust determination models have been developed for WSNs.

Sylvia et al. [7] proposed a trust based routing scheme for MANETs under adverse environment and compare the performance of the proposed scheme against existing schemes. The simulation results demonstrate that in adverse environment, the proposed scheme improves network throughput and delay; however, it incurs higher overhead. Sajjad et al. [8] developed an intrusion detection system (IDS) for WSN based on the trust level of the neighboring nodes. Based on their investigations of the network statistics and malicious node behaviors, their IDS can successfully detect selective forwarding, Hello flood, and Jamming attacks and isolate the malicious node accordingly.

Ferdous et al. [9] developed a novel trust management scheme in MANETs based on the responsibility of each node to build its trust level and tracking the node trust level, namely, the Network Trust Management (NTM) scheme. Atakli et al. [10] developed a novel scheme based on weighted trust estimation in order to detect and isolate the compromised nodes in hierarchical clustered WSN structure. In this scheme, they select some nodes as Forwarding Nodes to give a trust values for all of the cluster nodes. Afterwards, they decrease the node's trust level for all nodes that sent a meaningless/wrong information.

Cordasco and Wetzel [11] compared the performance of two MANET routing protocols; these are: the Secure AODV (SAODV) and Trusted AODV (TAODV),

which address routing security through cryptographic and trust-based means respectively. Gong et al. [12] presented a routing protocol for the purpose of energy efficiency and security in WSNs, named, Secure and Energy Aware Routing Protocol (ETARP). The main contribution point in ETARP is route discovering and selection based on both the maximum utility concept. ETARP scheme takes into consideration the energy efficiency and the trustworthiness in routing protocol, which may sustain more complexity and overhead on the network compared to AODV routing protocol.

Rani et al. [13] developed a trust scheme for determining the trustworthiness of the sensor nodes. They proposed a dynamic and decentralized system unlike most of other existing trust mechanisms that are centralized and suffer the single point failure problem. Another trust-based intrusion detection system (TIDS) scheme for wireless ad-hoc networks was presented by Deb and Chaki [14]. TIDS can detect intrusions and routes messages to avoid the compromised nodes. Trust is evaluated as the weighted sum of direct evaluation of the neighboring nodes and from indirect evaluations.

Pirzada and McDonald [18] developed a trust determination mechanism, where nodes calculate situational trust according to observed events and then use an aggregated general trust for routing decisions. Sun et al. [19] presented an information theoretic framework to quantitatively measure trust. They developed four axioms and based on these axioms, they presented two trust models: entropy-based and probability-based models, which satisfy all the axioms. Simulations showed that these models could significantly improve the network throughput as well as effectively detect malicious behaviors in MANETs.

A dynamic trust prediction models to evaluate the trustworthiness of nodes were also developed by K. Haldar et al. [20], Park et al. [21], Saini and Gautam [22], Xia et al. [23], Gowda and Hiremath [24], and Patil et al. [25]. A trust establishment and management framework for hierarchical wireless sensor networks was developed by Zhang et al. [26], and also a trust management framework (TMF) for MANETs was developed by Guo et al. [27].

3. Trust Determination

The core of any trust-based malicious node identification and isolation application is the fairness and cost of calculating the trust of the nodes. Therefore, it is important to have an algorithm that meets these objectives. The major requirements of trust determination algorithms are [7]:

a. Determine an initial trust value for any new arriving node, so that it will be either trusted by other nodes in the network or rejected.

b. Periodically determine the trust of the nodes in the network based on their current behavior and reputation among their neighbors.

c. Send the newly determined trusts to other nodes in the network with minimum overheads.

Trust determination models are usually used in dynamic and multivariable environment, so that it is important to define some configuration parameters that should be carefully attuned at each node to suit the environment and to optimize the performance of the trust determination model. These parameters include[7, 10, 11]:

a. Minimum Acceptable Trust (MAT). The minimum trust a node should have in order to be trusted by other nodes ($0 < MAT \leq 1$).

b. Trust Update Time (TUT). The minimum duration before updating current trust of any node in the network.

c. Minimum Trustable Participants (MTPs). The minimum number of trustable neighbors that should participate in determining the trust of any other node in the network. For example if MTP=2, then for Node *i* to determine the trust of Node *j* ($T_{i,j}$), Node *i* should get trusts for Node *j* from at least 1 other trustable node in the network.

3.1. The NWTD Algorithm

In this work, we use a modified version of the neighbor-weight trust determination (NWTD) algorithm [15], because of its fairness, fast convergence, and minimum overheads. In the NWTD algorithm, each node periodically broadcasts an extended HELLO (e-HELLO) message to its first-hop neighbors. The e-HELLO message has the same format of the standard HELLO message, and some additional data appended at the end of the message including the IDs of the first-hop neighbors and the trust of each of them.

On the other hand, the receiving node performs the following:

- Compare the trust of the broadcasting node with its pre-set MAT. If it is ≥MAT, then the node is trustable and message is accepted, otherwise, the node is not trustable and the message is discarded.
- Extract the IDs and trusts of the first-hop neighbors received from the broadcasting nodes that pass the above test (i.e., trustable).
- Construct a table listing the first-hop neighbor(s) and the trusts they have for each other. The node is fully trusted by itself (i.e., $T_{x,x}$=1)

- Determine the trust of the first-hop neighbors using the mathematical model described below.
- Remove from the routing table any node for which the determined trust is <MAT.
- Broadcast the updated trusts to the first-hop neighbors.

Each node $s(x)$ calculates the trust of its first-hop neighbors as follows:

a. Calculate the average trust of the first-hop neighbors of x using the following formula:

$$\bar{T}_{s(x),s(j)} = \frac{1}{k_j} \sum_{i=0}^{k_j} T_{s(i),s(j)} \quad (j=1 \text{ to } n)\ (k_j \geq \text{MTP}) \tag{1}$$

Where

$T_{s(i),s(j)}$ Trust of $s(j)$ as determined by $s(i)$, where $s(i)$ and $s(j)$ are first-hop neighbors.

$\bar{T}_{s(x),s(j)}$ Average trust of $s(j)$ as calculated by $s(x)$.

k_j Number of nodes that have trust for $s(j)$.

n Number of the one-hop neighbors of $s(x)$.

MTP Minimum trustable participants.

b. Calculate the trust of $s(j)$ as the sum of the product of the trust of $s(j)$ as determined by $s(i)$ and the weight of $s(i)$ as determined by $s(x)$ using the following mathematical formula:

$$T_{s(x),s(j)} = \sum_{i=0}^{k_j} w_{s(x),s(i)}^{s(j)} \cdot T_{s(i),s(j)}\ (j=1 \text{ to } n)\ (k_j \geq \text{MTP}) \tag{2}$$

Where $w_{s(x),s(i)}^{s(j)}$ is the weight of $s(i)$ as determined by $s(x)$ for a particular $s(j)$.

The value of $w_{s(x),s(i)}^{s(j)}$ is calculated as:

$$w_{s(x),s(i)}^{s(j)} = \frac{\bar{T}_{s(i),s(j)}}{\sum_{m=0}^{k_j} \bar{T}_{s(m),s(j)}} \quad (i=1 \text{ to } k_i,\ j=1 \text{ to } n)\ (k_j \geq \text{MTP}) \tag{3}$$

The average trusts assist to compute the weights of the nodes that determine the new trust of their first-hop neighbors, which means that the first-hop neighbors share their ideas before deciding the trust of any of their neighbors, and the contribution of each neighbor depends on its weight as determined by the receiving node.

3.2. Master and Monitoring Nodes

The NWTD algorithm defines two types of nodes: master and monitoring nodes. A master node is a pre-selected node that can take the responsibility of testing new arriving nodes, determines their initial trusts using certain testing procedures, and then broadcasts the initial trust to its first-hop neighbors. The monitoring node on the other hand, monitors the behavior of its first-hop neighbors, and for any malicious behaviors, it reduces the trust of the maliciously behaving node using the following simple linear equation:

$$T_{updated} = \alpha \; T_{current} \tag{4}$$

Where $T_{current}$ and $T_{updated}$ are the trust of the first-hop neighbor before and after the update; and α is the update factor ($0 \leq \alpha \leq 1$). The value of α can be determined dynamically by the monitoring node for each of its first-hop neighbor, based on its current trust and reputation among the neighbors; which means different values of α can be determined for different nodes at the same time.

Any node in the network can be nominated and act as the monitoring node as long as it has the capabilities to monitor the behavior of its neighbors, detect their malicious behavior, and update their trust accordingly. Furthermore, it can be easily recognized that the trust of the node depends on the number of its first-hop neighbors, the trusts of the node as determined by its first-hop neighbors, and the weight of the first-hop neighbors.

4. The FTMI Scheme

This section presents description of the proposed fair trust-based malicious node identification and isolation (FTMI) scheme. It consists of three phases:

Phase I: Trust Initialization

As we described before, the algorithm relies on one or more master nodes to test and determine the initial trust of any new arriving nodes, and then broadcasts the initial trust to the first-hop neighbors. This process consumes some of the nodes' power and requires some processing capabilities. The NWTD algorithm is originally developed for trust determination in MANETs, where nodes have more processing and power resources than the sensor nodes in WSNs, which are characterized by their limited processing and power resources. Hence, in the FTMI scheme, all sensor nodes are initially assumed to be fully trustable and this eliminates the need for the master node.

Phase II: Trust Determination

In this phase, each node uses the NWTD algorithm to determine the trust of all nodes on its list using Eqns. (1) to (3). Furthermore, if the node is a monitoring node, it first examines the behavior of its first-hop-neighbors, and if it identifies any malicious behavior for the node, it updates its trust using Eqn. (4) with pre-set update factor □.

Phase III: Malicious Node Isolation

In this phase, each node after determining the trust of all nodes on its list, compares the current trust of each node with the pre-set MAT, and if the trust of any node on the list is less than MAT, the node is dropped from the list and isolated. Afterwards, the node broadcasts the new determined trusts to its first-hop neighbors.

In summary, all existing nodes or newly arriving nodes are initially fully trustable, their behavior will be evaluated by their first-hop neighbors, new average trusts are collaboratively determined, and if the new trust of any node is less than the preset MAT, the node is dropped from the list. This continues throughout the network lifetime to ensure continuous monitoring, detection of malicious behavior, and isolation of any malicious node to maintain network stability.

5. Simulation Environment and Results

The modified NWTD algorithm described above is implemented and integrated with the Mobile Ad Hoc Network Simulator (MANSim). MANSim is a network simulator written with C++ programming language for evaluating the performance of various MANETs protocols [17].

In this paper, we consider a simulation environment that illustrates a network area of 150×150 m with 50 nodes each having 30 m transmission radius. The nodes are randomly distributed across the network as shown in Figure 1. The nodes are assumed to be stationary throughout the simulations.

The simulation is carried-out for a number of cycles *C*. During each cycle, the monitoring node evaluates the behavior of its first-hop neighbors, and if it detects any malicious behavior, it degrades the trust of the maliciously behaving node with a pre-set update factor □, and broadcasts the trust of its first-hop neighbors. Practically, the node is isolated if its trust becomes <MAT. Thus, the malicious node lifetime can be expressed as the number of cycles until the node is isolated.

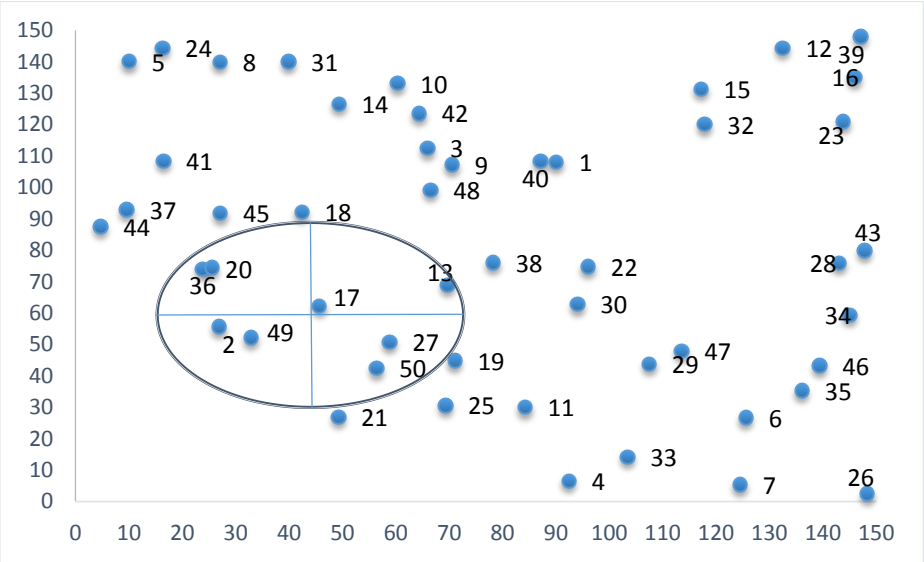

Figure 1. Nodes distribution in 150 ×150m network area for all scenarios.

In order to evaluate the performance of the FTMI scheme, the MANSim simulator has been used to simulate two scenarios considering different network environments and configuration parameters.

5.1 Scenario #1: Trust Estimation

In this scenario we determine the trust of a malicious node with different numbers of monitoring nodes *m* assuming a fixed value of α (i.e α=0.5). It is assumed that node 17 (N_{17}) behaves maliciously. Figure 1 shows that N_{17} has 7 first-hop neighbors; which are: N_2, N_{13}, N_{20}, N_{27}, N_{36}, N_{49}, and N_{50}.

The trusts of N_{17} as determined by its first-hop neighbors after 60 cycles are listed in Table 1 for different values of *m*. The results in Table 1 are also plotted in Figure 2. It can be clearly seen from the simulation results in Table 1 and Figure 2 that, after 60 cycles, the trust of N_{17} decreases as *m* increases, enabling early isolation for the malicious node. For example, after 60 cycles, the trust of N_{17} as determined by N_2 ($T_{2,17}$) is 0.762 when N_2 is the only monitoring node (i.e. *m*=1), and $T_{2,17}$ is 0.309 when all seven first-hop neighbors of N_{17} are monitoring nodes (i.e. *m*=7). This

means that the trust and lifetime of the malicious node decreases as the number of the first-hop monitoring nodes increases. The reason for this significant reduction in the trust with increasing *m*, is due to the fact that all of first-hop monitoring nodes reduce the trust of N_{17} simultaneously and consequently the average trust of N_{17} is also reduced.

Table 1 - Trust of N_{17} as determined by its first-hop neighbors ($T_{x,17}$) after 60 cycles.

#	Monitoring Nodes	$T_{2,17}$	$T_{13,17}$	$T_{17,17}$	$T_{20,17}$	$T_{27,17}$	$T_{36,17}$	$T_{49,17}$	$T_{50,17}$
1	N_2	0.762	0.909	1.000	0.768	0.887	0.768	0.818	0.887
2	N_2, N_{13}	0.697	0.696	1.000	0.705	0.710	0.705	0.713	0.710
3	N_2, N_{13}, N_{20}	0.543	0.647	1.000	0.548	0.648	0.556	0.603	0.648
4	N_2, N_{13}, N_{20}, N_{27}	0.487	0.492	1.000	0.494	0.500	0.503	0.505	0.510
5	N_2, N_{13}, N_{20}, N_{27}, N_{36}	0.387	0.471	1.000	0.395	0.470	0.395	0.437	0.480
6	N_2, N_{13}, N_{20}, N_{27}, N_{36}, N_{49}	0.321	0.427	1.000	0.333	0.415	0.333	0.368	0.427
7	N_2, N_{13}, N_{20}, N_{27}, N_{36}, N_{49}, N_{50}	0.309	0.347	1.000	0.321	0.342	0.321	0.323	0.342

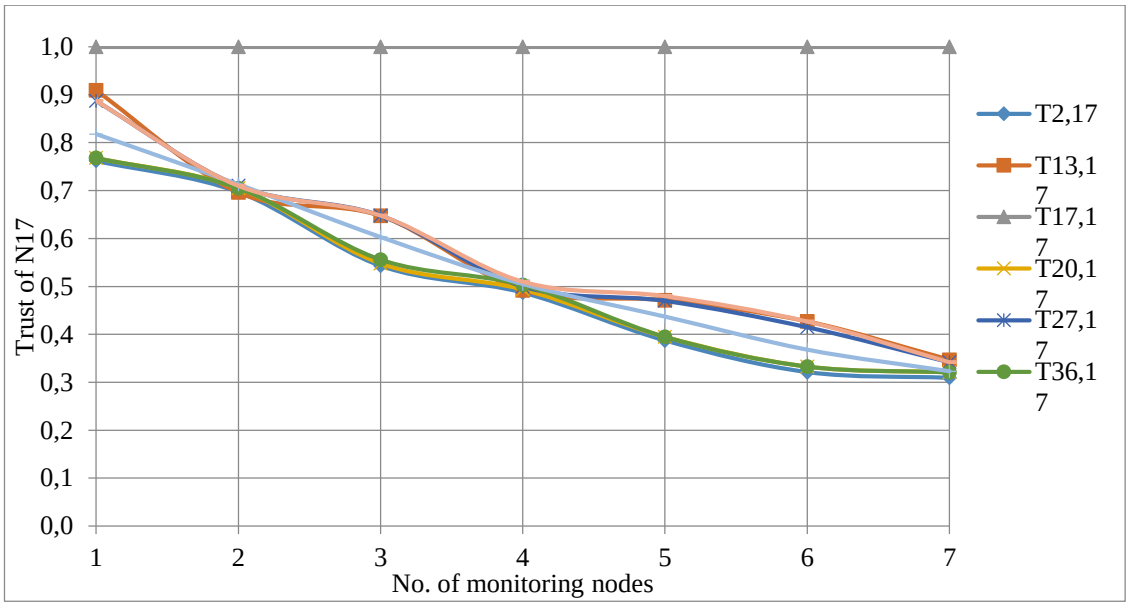

Figure 2. Trust of N_{17} as determined by its first-hop neighbors ($T_{x,17}$) after 60 cycles.

5.2 Scenario #2: Lifetime Estimation

In this scenario, the effect of α and MAT on the malicious node lifetime is investigated for several values of *m*. Five values of α (0.5, 0.6, 0.7, 0.8, and 0.9) and four values of MAT (0.3, 0.5, 0.7, and 0.9) are considered.

For each set of α and MAT, we run the simulator to determine the number of cycles required to isolate the malicious node N_{17} considering different values of *m*. A node is isolated when its trust becomes <MAT. The lifetime of the malicious node in number of cycles is given in Table 2. The results for the smallest and largest values of *m* (i.e. *m*=1 and *m*= 7) in Table 2 are also plotted in Figures 3-(a) and 3-(b) respectively. Further, the results for the smallest and largest values of α (i.e. α=0.5 and α = 0.9) in Table 2 are also plotted in Figures 4-(a) and 4-(b) respectively.

The results obtained demonstrate the following:

- For any set of *m* and MAT, the lifetime of the malicious node increases with increasing the trust update factor α as shown in Figures 3-(a) and 3-(b). This

is because higher α means that the trust of the node is slightly reduced due to the node malicious behavior, which takes longer time for the trust of the maliciously behaving node to become <MAT. Since the trend of the Figures is same for all values of *m*, we only showed the figures for the smallest and largest values of *m* (i.e. *m*=1 and *m*= 7). In addition, each of these figures shows the variation for the different values of MAT (0.3, 0.5, 0.7, and 0.9).

- For any set of □ and MAT, the lifetime of the malicious node decreases with increasing *m* (in this paper, *m* varies between 1 and 7) as shown in Figures 4-(a) and 4-(b). This is because as *m* increases, the number of nodes which identify the malicious node is increased, and simultaneously the trust of the maliciously behaving node is quickly reduced to become <MAT. Since the trend of the Figures is same for all values of α, we only showed the figures for the smallest and largest values of α (i.e. α=0.5 and α= 0.9). Furthermore, each of these figures shows the malicious node lifetime for various values of MAT (0.3, 0.5, 0.7, and 0.9).

The range of the malicious node lifetime is higher for smaller values of *m* and MAT. For example, for *m*=1 and MAT=0.3, the lifetime increased from 320 cycles for α=0.5 to 1333 cycles for α=0.9, while for *m*=7 and MAT=0.9, the lifetime is only 4 cycles for α=0.5 to 9 cycles for α=0.9. The simulation results demonstrate that a proper adjustment of the network and configuration parameters (*m*, α and MAT) is crucial to ensure instant isolation of any malicious node in the network and maintain network stability.

It can be clearly recognized that *C* depends on *m*, α, and MAT, which can be mathematically expressed as:

$C=f(m, \alpha, \text{MAT})$

C is a dependent variable representing the number of cycles, and m, α, and MAT are the independent variables representing the network and configuration parameters. These parameters must be carefully selected to ensure quick isolation for any malicious node in the network. This is a multi-variable optimization problem, and it is necessary to rate the influence of these parameters on the malicious node lifetime. Applying the CCR (Charnes-Cooper-Rhodes) model of the data envelopment analysis tool [28] on our data of Table 2, we found that the trust update factor α is the relatively efficient factor in determining the number of cycles C. Moreover, the near optimal values of the above parameters were found to be as m=1, α=0.5, and MAT=0.9.

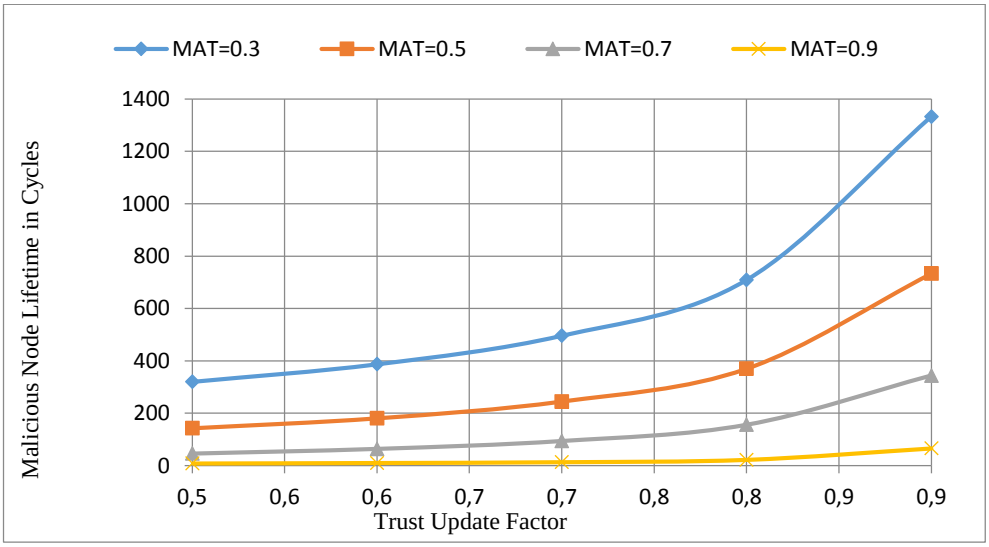

Figure 3 (a). Malicious node lifetime (m=1).

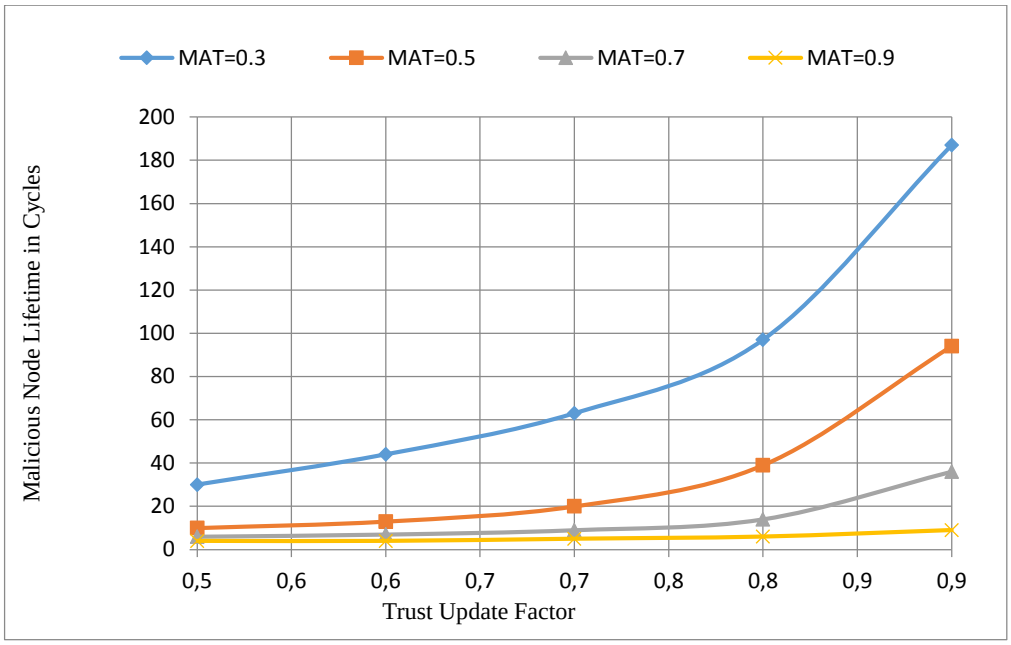

Figure 3 (b). Malicious node lifetime (m=7).

Figure 3. Malicious node lifetime against α for various m and MAT.

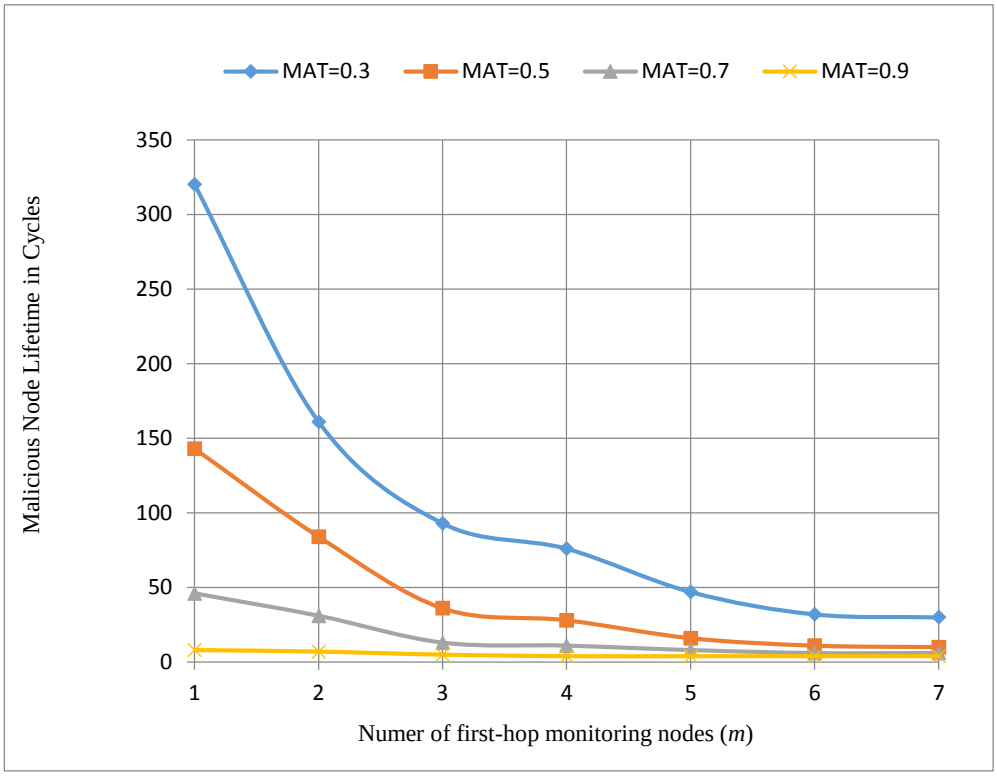

Figure 4 (a). Malicious node lifetime (α =0.5).

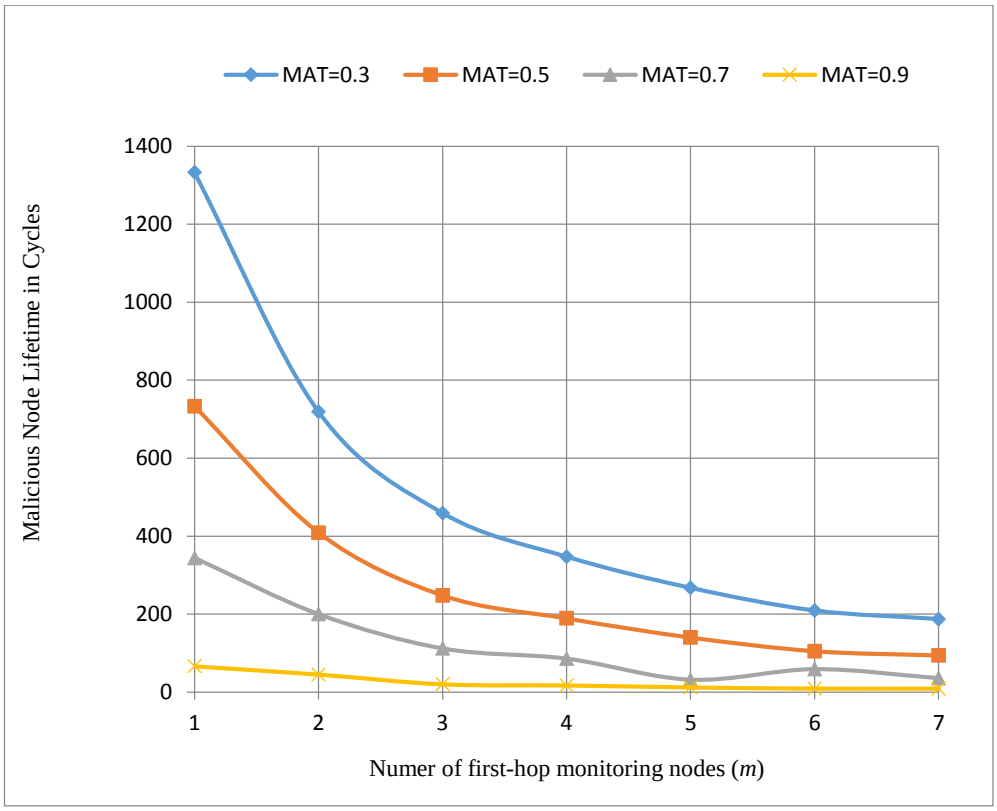

Figure 4 (b). Malicious node lifetime (α =0.9).

Figure 4. Malicious node lifetime against m for various α and MAT.

Table 4. Malicious node lifetime against m for various α and MAT.

Table 2 - Malicious node lifetime for various m, α, and MAT.						
m	Monitoring Nodes	a	MAT			
			0.3	0.5	0.7	0.9
1	N_2	0.5	320	143	46	8
		0.6	387	181	64	10
		0.7	496	245	94	13
		0.8	709	370	156	22
		0.9	1333	733	344	66
2	N_2, N_{13}	0.5	161	84	31	7
		0.6	197	105	41	8
		0.7	257	139	59	12
		0.8	373	207	95	19
		0.9	719	409	200	45
3	N_2, N_{13}, N_{20}	0.5	93	36	13	5
		0.6	117	50	17	6
		0.7	156	73	25	7
		0.8	232	117	46	10
		0.9	459	248	112	20
4	N_2, N_{13}, N_{20}, N_{27}	0.5	76	28	11	4
		0.6	95	39	15	5
		0.7	125	57	21	6
		0.8	182	92	36	8
		0.9	347	189	86	17
5	N_2, N_{13}, N_{20}, N_{27}, N_{36}	0.5	47	16	8	4

		0.6	64	22	10	5
		0.7	88	32	13	6
		0.8	134	62	22	7
		0.9	267	140	32	12
6	N_2, N_{13}, N_{20}, N_{27}, N_{36}, N_{49}	0.5	32	11	6	4
		0.6	46	14	8	5
		0.7	67	22	10	5
		0.8	105	43	15	6
		0.9	209	105	59	9
7	N_2, N_{13}, N_{20}, N_{27}, N_{36}, N_{49}, N_{50}	0.5	30	10	6	4
		0.6	44	13	7	4
		0.7	63	20	9	5
		0.8	97	39	14	6
		0.9	187	94	36	9

6. Conclusions

This paper develops and evaluates the performance of a new efficient and reliable fair trust-based malicious node identification and isolation scheme in WSNs, namely, the FTMI scheme. The new scheme utilizes the NWTD algorithm to determine and update the trust of the network nodes. A node is classified as malicious node and isolated if its trust becomes <MAT. The performance of the new scheme is evaluated through a number of simulations using the network simulator MANSim. We investigated the effect of the configuration parameters on the system performance. Specifically, the number of first-hop monitoring nodes (m), the update factor (α), and the minimum acceptable trust (MAT).

The simulation results demonstrate that the scheme can identify and isolate any malicious node in the network with a lifetime that can be controlled by proper adjustment of the above three parameters. For example, for m=1, α=0.5, and MAT=0.9, a single monitoring node requires only 8 cycles to isolate the malicious node and restore network stability. This could be reduced to 4 cycles only if m=7. On the other hand, the number of cycles and consequently the malicious node lifetime could be as high as 1333 cycles for m=1, □=0.9, and MAT=0.3. Using data envelopment analysis, it is found that the trust update factor □ is the most efficient

factor in determining the number of cycles *C*. The results demonstrate the effectiveness of the FTMI scheme in identifying and isolating malicious nodes in a fairly and timely manner.

The scheme developed in this paper is still fresh research and require extensive research and evaluation. For example, investigating the effect of nodes density and mobility on the malicious node lifetime for a certain set of network and configuration parameters is a worthy study.

References

[1] H. M. A. Fahmy. Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis. Springer, 2016.

[2] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor Network Security: A Survey". IEEE Communications Surveys and Tutorials, Vol. 11, No. 2, pp. 52-73, 2009.

[3] N. Mukherjee, S. Neogy, and S. Roy. "Building Wireless Sensor Networks: Theoretical and Practical Perspectives". CRC Press, 2015.

[4] A. Ahmed, K. Abubakar, M. Channa, K. Haseeb, and A. Khan, "A Survey on Trust Based Detection and Isolation of Malicious Nodes in Ad-Hoc and Sensor Networks," Frontiers of Computer Science Journal, Vol.9, No. 2, pp. 280-296, 2015.

[5] D. Martins and H. Guyennet, "Wireless Sensor Network Attacks and Security Mechanisms: A Short Survey". Proceedings of the 13th International Conference on Network-Based Information Systems (NBiS), 14-16 September 2010, pp. 313-320, 2010.

[6] K. Xing, S. Srinivasan, M Rivera, J. Li, and X. Cheng, "Attacks and Countermeasures in Sensor Networks: A Survey". Book Chapter in Network Security, pp. 251-272, Springer, New York, 2010.

[7] D. Sylvia, J. Katiravan, and D. Srinivasa Rao, "Trust-based Routing in Wireless Ad Hoc Networks under Adverse Environment". International Journal of Computer Applications (IJCA), Vol. 136, No.10, pp. 23-28, 2016.

[8] Syed Muhammad Sajjad, Safdar Hussain Bouk, and Muhammad Yousaf, "Neighbor Node Trust Based Intrusion Detection System for WSN". Procedia Computer Science, Vol. 63, pp. 183-188, 2015.

[9] R. Ferdous, V. Muthukkumarasamy, and A. Sattar, "A Node-based Trust Management Scheme for Mobile Ad-Hoc Networks". Proceedings of the 4th International Conference on Network and System Security (NSS), pp. 275–280, 2010.

[10] Idris M. Atakli, Hongbing Hu, Yu Chen, Wei-Shinn Ku, and Zhou Su, "Malicious Node Detection in Wireless Sensor Networks using Weighted Trust Evaluation". Proceedings of the Symposium on Simulation of Systems Security (SSSS'08), Ottawa, Canada, April 14 –17, pp. 836-843, 2008.

[11] J. Cordasco and S. Wetzel, "Cryptographic Versus Trust-based Methods for MANET Routing Security". Electronic Notes in Theoretical Computer Science, Vol. 197, No. 2, pp. 131–140, 2008.

[12] P. Gong, T. M. Chen, and Q. Xu, "ETARP: An Energy Efficient Trust-Aware Routing Protocol for Wireless Sensor Networks". Journal of Sensors, Vol. 2015, Article ID 469793, 10 pages, 2015.

[13] Rani, JayaKumar and Divya, "Trust Aware Systems in Wireless Sensor Networks", Proceedings of the International Conference on Computing and Communications Technologies (ICCCT), pp. 174-179, 2015.

[14] N. Deb and N. Chaki, "TIDS: Trust-Based Intrusion Detection System for Wireless Ad-hoc Networks", CISIM 2012: Computer Information Systems and Industrial Management, vol 7564, pp 80-91, 2012.

[15] H. Al-Bahadili, "Trust Determination in Wireless Ad Hoc Networks". Book Chapter in Handbook of Research on Threat Detection and Countermeasures in Network Security, Chapter 18, pp. 330-348, 2015.

[16] A. Abu Romman, "Evaluating the Performance of the Novel Neighbor Weight-Based Trust Determination Algorithm in Wireless Ad Hoc Networks". MS Thesis, Princes Sumaya University of Science and Technology, Faculty of Information Technology. Amman, Jordan, 2013.

[17] H. Al-Bahadili, "On the Use of Discrete-Event Simulation in Computer Networks Analysis and Design". In Handbook of Research on Discrete-Event Simulation Environments: Technologies and Applications). Information Science Reference, Chapter 19, pp. 418-442, 2010.

[18] A. A. Pirzada and C. McDonald, "Establishment in Pure Ad-Hoc Networks". Wireless Personal Communications, Vol. 37, No. 1-2, pp. 139–168, 2006.

[19] Y. L. Sun, S. Member, Z. Han, and K. J. R. Liu, "Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks". IEEE Journal on Selected Area in Communications, Vol. 24, pp. 305–317, 2006.

[20] K. Haldar, N. Narayan and B. K. Mishra, "Mathematical Model on Selfishness and Malicious Behavior in Trust based Cooperative Wireless Networks". International Journal of Computer Network and Information Security, Vol. 10, pp. 15-22, 2015.

[21] S.-S.Park, J.-H.Lee, and T.-M. Chung, "Cluster-Based Trust Model against Attacks in Ad-Hoc Networks". Proceedings of the 3rd International Conference on Convergence and Hybrid Information Technology (ICCIT '08),Vol. 1, pp. 526–532, 2008.

[22] R. Saini and R. K. Gautam, "Establishment of Dynamic Trust among Nodes in Mobile Ad-Hoc Network". Proceedings of 2011 International Conference on Computational Intelligence and Communication Networks (CICN), pp. 346–349, 2011.

[23] H. Xia, Z. Jia, X. Li, L. Ju, and E. Sha, "Trust Prediction and Trust-Based Source Routing in Mobile Ad Hoc Networks". Journal of Ad Hoc Network, Vol. 11, Issue 7, pp. 2096-2114, 2013.

[24] S. R. Gowda and P. S. Hiremath, "Secure Routing Schema for MANET with Probabilistic Node-to-Node Forwarding". International Journal of Computer Science Issues (IJCSI), Vol. 10, No. 3, pp. 51-58, 2013.

[25] K. Patil, P. Bhanodia, and S. Joshi, "A Novel Paradigm RIP (Reputation Index Protocol) for MANET". International Journal of Engineering Research & Technology (IJERT), Vol. 2, No. 1, pp. 1-5, 2013.

[26] J. Zhang, R. Shankaran, M. A. Orgun, V. Varadharajan, and A. Sattar, "A Dynamic Trust Establishment and Management Framework for Wireless Sensor

Networks". Proceedings of the 2010 IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC), pp. 484–491, 2010.

[27] J. Guo, A. Marshall, and B. Zhou, "A New Trust Management Framework for Detecting Malicious and Selfish Behaviour for Mobile Ad Hoc Networks". Proceddings of 2011 IEEE 10thInternational Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 142–149, 2011.

[28] M. Martic, M. Novakovic, and A. Baggia, "Data Envelopment Analysis- Basic Models and their Utilizations," Organizacija Journal, Vol. 42, No. 2, pp. 37-43, 2009.