Security of Software-defined Wireless Sensor Networks

by

Manaf Abdulrahman Bin-Yahya (Ben Yahya)

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2022

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Vojislav Misic Professor Ryerson University
Supervisor(s):	Xuemin (Sherman) Shen Professor University of Waterloo
Internal Member:	Xiaodong Lin Professor University of Guelph
Internal Member:	Oleg Michailovich Associate Professor University of Waterloo
Internal-External Member:	Jun Liu Associate Professor University of Waterloo

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Wireless Sensor Network (WSN) using Software Defined Networking (SDN) can achieve several advantages such as flexible and centralized network management and efficient routing. This is because SDN is a logically centralized architecture that separates the control plane from the data plane. SDN can provide security solutions, such as routing isolation, while handling the heterogeneity, scalability, and the limited resources of WSNs. However, such centralized architecture brings new challenges due to the single attack point and having non-dedicated channels for the control plane in WSNs. In this thesis, we investigate and propose security solutions for software-defined WSNs considering energy-efficiency and resource-preservation. The details are as follows.

First, the functionality of software-defined WSNs can be affected by malicious sensor nodes that perform arbitrary actions such as message dropping or flooding. The malicious nodes can degrade the availability of the network due to in-band communications and the inherent lack of secure channels in software-defined WSNs. Therefore, we design a hierarchical trust management scheme for software-defined WSNs (namely TSW) to detect potential threats inside software-defined WSNs while promoting node cooperation and supporting decision-making in the forwarding process. The TSW scheme evaluates the trustworthiness of involved nodes and enables the detection of malicious behavior at various levels of the software-defined WSN architecture. We develop sensitive trust computational models to detect several malicious attacks. Furthermore, we propose separate trust scores and parameters for control and data traffic, respectively, to enhance the detection performance against attacks directed at the crucial traffic of the control plane. Additionally, we develop an acknowledgment-based trust recording mechanism by exploiting some built-in SDN control messages. To ensure the resilience and honesty of the trust scores, a weighted averaging approach is adopted, and a reliability trust metric is also defined. Through extensive analyses and numerical simulations, we demonstrate that TSW is efficient in detecting malicious nodes that launch several communication and trust management threats such as black-hole, selective forwarding, denial of service, bad and good mouthing, and ON-OFF attacks.

Second, network topology obfuscation is generally considered a proactive mechanism for mitigating traffic analysis attacks. The main challenge is to strike a balance among energy consumption, reliable routing, and security levels due to resource constraints in sensor nodes. Furthermore, software-defined WSNs are more vulnerable to traffic analysis attacks due to the uncovered pattern of control traffic between the controller and the nodes. As a result, we propose a new energy-aware network topology obfuscation mechanism, which maximizes the attack costs and is efficient and practical to be deployed. Specifically, first, a

route obfuscation method is proposed by utilizing ranking-based route mutation, based on four different critical criteria: route overlapping, energy consumption, link costs, and node reliability. Then, a sink node obfuscation method is introduced by selecting several fake sink nodes that are indistinguishable from actual sink nodes, according to the k-anonymity model. As a result, the most suitable routes and sink nodes can be selected, and a highest obfuscation level can be reached without sacrificing energy efficiency. Finally, extensive simulation results demonstrate that the proposed methods strongly mitigate traffic analysis attacks and achieve effective network topology obfuscation for software-defined WSNs. In addition, the proposed methods reduce the success rate of the attacks while achieving lower energy consumption and longer network lifetime.

Last, security networking functions, such as trust management and Intrusion Detection System (IDS), are deployed in WSNs to protect the network from multiple attacks. However, there are many resource and security challenges in deploying these functions. First, they consume tremendous nodes' energy and computational resources, which are limited in WSNs. Another challenge is preserving the security at a sufficient level in terms of reliability and coverage. Watchdog nodes, as one of the main components in trust management, overhear and monitor other nodes in the network. Accordingly, a secure and energy-aware watchdog placement optimization solution is studied for software-defined WSNs. The solution balances the required energy consumption, computational resource, and security in terms of the honesty of the watchdog nodes. To this end, a multi-population genetic algorithm is proposed for the optimal placement of the watchdog function in the network given the comprehensive aspects of resources and security. Finally, simulation results demonstrate that the proposed solution robustly preserves security levels and achieves energy-efficient deployment.

In summary, reactive and proactive security solutions are investigated, designed, and evaluated for software-defined WSNs. The novelty of these proposed solutions is not only efficient and robust security but also their energy awareness, which allows them to be practical on resource-constrained networks. Thus, this thesis is considered a significant advancement toward more trustworthy and dependable software-defined WSNs.

Acknowledgements

My unreserved gratitude and praises are for Allah, the Most Compassionate and the Most Merciful. He has blessed me with His bounties, and He has given me the strength and courage to reach my goals during the course of this research.

I would like to express my deepest gratitude to Professor Xuemin (Sherman) Shen, my advisor, for his continued guidance, invaluable advice, deep insight, and support during my Ph.D. journey at the University of Waterloo. His sharp sense of research direction and great enthusiasm has been a tremendous force in the completion of this thesis.

I would also like to express my extreme appreciation to the examining committee members of this thesis, Professor Xiaodong Lin, Professor Oleg Michailovich, Professor Jun Liu, and Professor Vojislav Misic. Their insightful comments have significantly improved the quality and presentation of this thesis.

With great appreciation, I acknowledge Hadhramout Foundation for granting me the scholarship and giving me the opportunity to enhance my experience and education.

I would like to extend my sincere thanks to Professor Ladan Tahvildari for being a crucial part of my journey at the University of Waterloo. I feel very fortunate and honored to be one of her course's teaching assistantships.

I would also like to thank my colleagues and friends at BBCR Lab, especially in the Security subgroup. My discussions with Dr. Cheng Huang, Dr. Dongxiao Liu, Dr. Jianbing Ni, and many other current and former BBCR members have given me many inspirations.

My special regards and love to Waterloo friends: Omar Sababha, Mohammad Shahab, Maged Aldhaeebi, Abdulrahman Hamid, Mohammed Alhasani, Khalid Aldubaikhy, Amr Salah, and Hossam Amer. There are many other people whose names are not mentioned due to the limited space. It does not mean that they are not important and I have forgotten or ignored their help. It is a privilege for me to share life with so many bright, energetic and helpful people.

Forever I shall remain indebted to my family for their unconditional love, continuous support, and sincere prayers. Special thanks to my parents and my sisters. Without you, I would not be the person I am today.

Above all, I would like to thank my beloved wife, Hala Al-Kaf, for always believing in me and supporting me. Your unconditional love, encouragement, and understanding have been and will always be a great source of motivation in my life. But most of all, thank you for being my best friend. I owe you everything. To my lovely kids, Tuqa and Hashem, you always fill my life with joy and happiness.

Dedication

To my beloved parents, Abdulrahman and Fatima

To my beloved wife, Hala

To my beloved kids, Tuqa and Hashem

To my beloved sisters, Manal, Manar, and Maram

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CH Cluster Head 18

DoS Denial of Service 4

IDS Intrusion Detection System 10, 17

IoT Internet of Things 2

NTO Network Topology Obfuscation 8

QoS Quality of Service 3

SDN Software Defined Network 1, 2, 10

WSN Wireless Sensor Network 1, 2, 10

Chapter 1

Introduction

In this chapter, we give a brief description of Wireless Sensor Network (WSN) and how Software Defined Network (SDN) platform was introduced to it to form the Software-Defined WSNs. Then, we present Software-Defined WSNs' challenges and mainly focus on security challenges. Finally, we describe our motivations and contributions of to protect the software-defined WSNs using trust management and network obfuscation methods.

1.1 Software-defined WSNs

WSN is a term that refers to a group of sensors that can perform multiple functions, mainly sensing and forwarding. Sensing is the primary function where each sensor, depending on its respective type, contributes to the system by providing certain readings and detecting some events about its environment. Examples include temperature sensors, sound sensors, and motion sensors [1]. Then, the sensed data is transferred wirelessly through the network to reach the destination. For applications involving randomly deployed sensors over a wide area, the data is often forwarded in a multi-hop manner. This results in the participation of multiple sensors in the forwarding process. To perform the aforementioned latter functionality, the sensor node must communicate with other nodes and perform the forwarding function based on the routing policy. Basically, the routing algorithm determines how the data should be routed through multi-hop transmission. WSN applications include military, manufacturing, tracking and movement monitoring, health monitoring, home intelligence, smart grid, and environmental [1].

SDN paradigm introduces solutions that add programmable and flexible features to WSNs [2]. In SDN, a centralized controller acts as an intelligent element in the architecture

whereby (over-the-top) services are implemented above this controller [3]. SDN paradigm intends to split the data and the control planes [4]. As a result, SDN-enabled devices (such as sensors) become non-intelligent devices in terms of making routing decisions. The controller provides the sensor nodes with flow rules for message forwarding for the routing function. The controller collects information about the network to create the routing map. To assign a path to a particular flow, the controller uses the shortest path algorithm to select the best route based on the hop counts from all feasible routes in the network [5]. The SDN controller acts as the intelligent player in the architecture where most services are implemented above this controller. Lately, the WSN and Internet of Things (IoT) networks have attracted attention for using SDN to achieve different benefits such as flexible and centralized network management, efficient routing, mobility and localization management, and security.

In software-defined WSNs, SDN handles the routing decisions, while in conventional WSNs, the nodes themselves must determine the routing algorithms. The SDN controller must determine the data forwarding rules based on its supervisory role in the network. This utilizes the power resources of the network efficiently and extends the network lifetime. Furthermore, SDN can provide security solutions for WSN and IoT networks while handling the heterogeneity, scalability, and limited resource environments of these networks. On the other hand, by using SDN-enabled devices, different sensor nodes from different vendors and technologies can be used in the network. The interoperability and heterogeneity of WSN components can be handled by the SDN [6]. As with general SDN, software-defined WSN consists of three main layers: data layer, control layer, and application layer [7]. As shown in Fig. 1.1, one of the main differences between the architecture of general SDN and the architecture of software-defined WSN is the network nodes. These nodes are the sensor nodes with limited resources. Also, these nodes are connected through a limited range of wireless communication. Additionally, the SDN controller communicates indirectly with the sensor nodes through multi-hop paths (control plane).

1.2 Challenges of Software-defined WSNs

1.2.1 Networking Challenges

As mentioned before, a key aspect of SDN technology is that it can simplify network management [8]. However, the network management in WSNs is very complicated due to the dynamic network conditions and restricted by the limited resources of network components. Transferring the control from the nodes to a centralized element with powerful

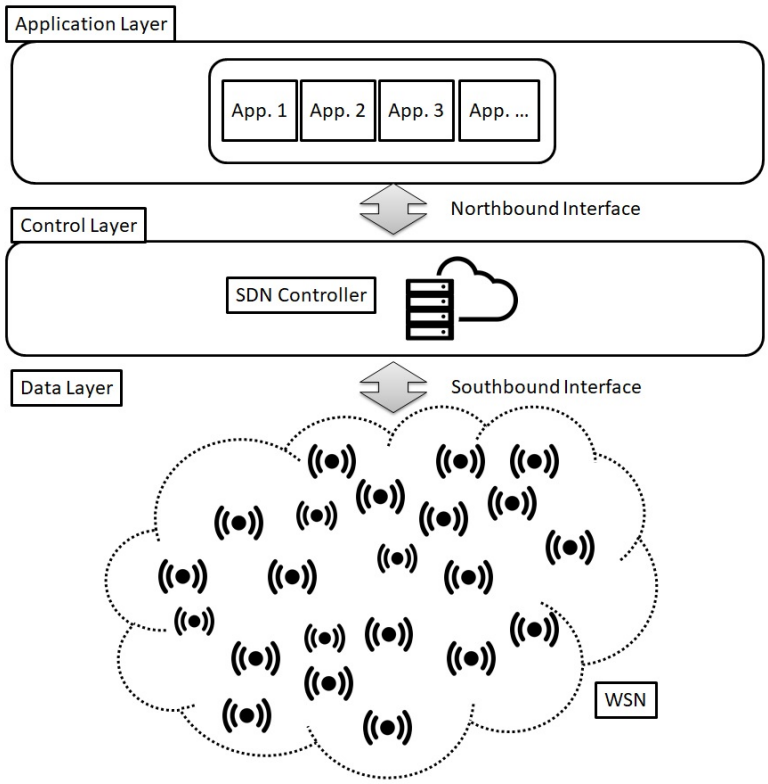


Figure 1.1: Software-defined WSN architecture.

resources utilizes the energy distribution and provides efficient network and resource management. Furthermore, SDN manages all aspects of scalability, mobility, and localization. Sensor nodes have a short network lifetime as they are often battery-powered. Therefore, the lifetime of the network depends mainly on the power consumption of the network sensors [9]. However, these nodes are required to perform different functions such as sensing, forwarding, and processing. The main challenge here is to efficiently utilize power resources without degrading the functionality of the whole network. Several solutions have been proposed in the literature, such as duty-cycling, data aggregation, sporadic sensing, and other solutions that depend on the topology of the network. Another challenging aspect is the classical routing problem due to its distributed manner and limited communication range [10]. Even the control logic of routing decisions is handled by the controller, many issues are encountered when solving routing problems, such as transmission overhead, scalability, and security. In some cases, the data must be transmitted based on Quality of Service (QoS) requirements. However, obtaining routes (flow rules) by the controller

to meet specific requirements costs transmission overhead due to the background traffic caused by the increase of control messages. Consequently, there is increased congestion and delay in the data network.

1.2.2 Security Challenges

Due to its open environment and limited resources, WSN faces several security challenges and vulnerabilities [11]. Several solutions with specific design requirements are proposed to solve the security issues of WSN, such as cryptographic and behavioral monitoring solutions. Attackers primarily attempt to impair different properties of network security, such as confidentiality, integrity, and availability. For example, with the help of malicious or compromised nodes, the attacker can launch selective forwarding, flooding and Denial of Service (DoS) attacks, and traffic analysis attacks. SDN can provide central management of security aspects by enabling proactive and reactive actions against incident attack [12]. Furthermore, removing most of the decisive actions from the node to the controller leads to omitting different types of WSN attacks. In addition, by using SDN flexibility and programmability features, the manual faults of a network configuration that cause significant vulnerabilities can be avoided. Additionally, SDN can provide intelligent routing with the advantage of injecting detection and mitigation techniques for potential attacks such as distributed DoS attacks. SDN-enabled devices provide the SDN controller with traffic information and statistics. Using this data, the controller will have the ability to detect threats and provide mitigation against the detected attack using all resources that lay under its control [13, 14].

SDN can provide security solutions (e.g., routing isolation) while handling the heterogeneity, scalability, and limited resources of WSN for IoT applications [12]. However, such a centralized architecture, brought forth by SDN, comes with new challenges due to not having dedicated channels for the control plane. In software-defined WSNs, there is no secure channel due to limited resources since the control plane uses in-band and multi-hop communication [15]. By contrast, in other scenarios, an SDN controller connects with devices through a secure channel, e.g., Transport Layer Security (TLS)/Secure Sockets Layer (SSL)-based communication [16]. The robustness of software-defined WSNs is heavily dependent on the control plane and the intermediate nodes between the controller and the rest of the network [17]. However, the identification of the security issues in software-defined WSNs has received little attention. Analyzing the attacks and defense aspects in software-defined WSNs is highly essential at this stage. In particular, software-defined WSNs have some unique properties, which differ from those of typical distributed WSNs. These unique properties range from the central controller to the inherited SDN vulnerabilities.

Similar to conventional WSNs, software-defined WSNs are highly vulnerable to traffic analysis attacks [18]. In these passive attacks, the adversary tries to obtain detailed knowledge about the network behavior (topology) [19]. Usually, the goal of these attacks is to identify flow routes and high-profile nodes that play significant roles in network communication, including sink nodes, intermediate nodes, and shared nodes of both control and data traffic. After discovering these nodes, the adversary can compromise (hijack) these nodes later or launch a DDoS attack . In most cases, sensed data and control traffic are transmitted along paths from source nodes to determined destinations such as sink nodes. This produces pronounced traffic patterns that can be revealed by monitoring the network traffic. Thus, the network cannot be guarded by applying only conventional privacy solutions such as encryption [20]. Moreover, some traffic analysis attacks from the conventional WSNs are exacerbated in the software-defined WSNs. Recall that SDN introduces communication between the controller and the nodes for the control plane. Therefore, due to the evident pattern of this additional control traffic, software-defined WSNs are more vulnerable to traffic analysis attacks.

1.3 Research Motivations and Contributions

Due to the open wireless environment and susceptibility to physical capture, software-defined WSNs are vulnerable to many communication threats that target the functionality and robustness of the network. Besides, they are vulnerable to traffic analysis attacks that reveal network topology. Although SDN adds several benefits for security to the software-defined WSN, such as supervised decisions and routing isolation, the network has some disadvantages in terms of network availability and single point of failure issues. SDN comes with new challenges due to the logically centralized architecture without having a secure dedicated channel for the control plane. The network depends mainly on the control plane, so any loss of connectivity will have an aggravated effect. As a result, threats that occur in the conventional WSNs will have more impact on the software-defined WSNs. Thus, we investigate reputation-based (reactive) and network obfuscation (proactive) security solutions to protect the network. In addition, we consider the resource limitations and energy consumption challenges of the network.

1.3.1 Securing Software-defined WSNs Communication via Trust Management

Motivations

As mentioned in section 1.2.2, having insecure in-band and multi-hop communication between the controller and sensor nodes increases the security attack surface, thereby making the network vulnerable to several communication threats that affect the network availability [21]. Message dropping and flooding behaviors are examples of communication threats as illustrated in Fig. 1.2. Crypto-based authorization and authentication techniques on their own are insufficient in dealing with such threats [22]. Therefore, it is imperative to develop trust management among the participating nodes to provide a secure reflection of the network. Trust management aims to secure WSNs against attacks launched by insider malicious nodes. It establishes trust relationships among network nodes based on their experiences with others [23]. Besides defending against communication threats, trust management schemes need to be robust against good/bad mouthing and ON-OFF attacks. In this research, we consider that all nodes (including malicious ones) are authorized and authenticated, and we focus on trust management.

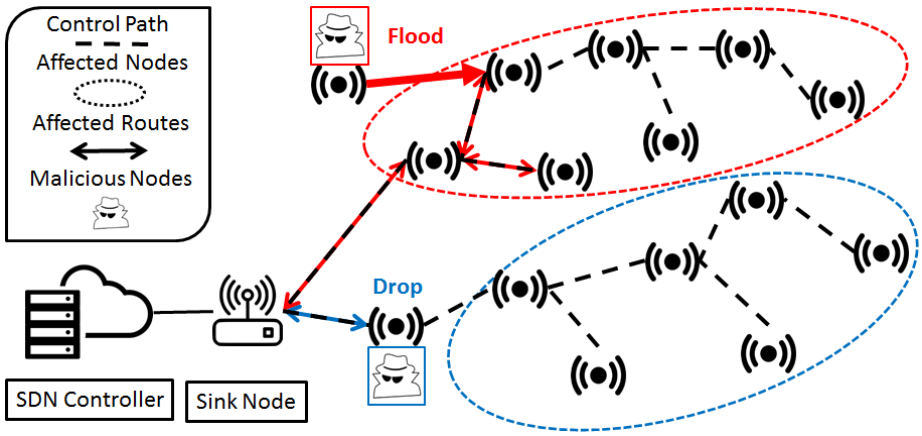


Figure 1.2: Examples of software-defined WSNs communication threats.

Extensive research efforts have been directed towards designing trust management schemes for conventional WSNs [24, 25, 26, 27, 28, 29]. Recall that SDN enables network programmability via the controller node and provides a decoupled separation between control and data planes. Therefore, trust management schemes for conventional WSNs cannot be directly deployed for software-defined WSNs, and they do not harness of

the introduced centralized and programmable capabilities. Moreover, due to the decoupled nature of software-defined WSNs (i.e., decoupled control and data planes), and due to the existence of a single point of failure, the effect of some threats can become aggravated, which calls for more careful consideration. Only few works address trust-based routing and trust management for software-defined WSNs [30, 31]. An energy-efficient trust management and routing mechanism (ETMRM) was proposed in [31]. ETMRM utilizes the node's trust score and residual energy to detect malicious behavior and ensure secure yet energy-efficient routing. The authors provide a flow-table extension to achieve lightweight trust monitoring at the node level. However, the inherent *decoupled* and *hierarchical* architecture of software-defined WSNs is not fully considered. That is, the aforementioned trust management scheme does not consider a separation between the control and data planes which is very crucial in such networks. Moreover, a trust management scheme that addresses the several hierarchical levels of software-defined WSNs is not yet studied. Moreover, we provide a thorough investigation and new careful design to trust evaluation models. See section 2.2.1 for a more detailed discussion.

Contributions

The goal of this research is to secure software-defined WSNs from nodes that behave maliciously to perform several attacks, e.g., black-hole, selective forwarding, and DoS (new-flow [31]). While considering the particular characteristics of software-defined WSNs, we propose a trust management scheme for software-defined WSNs (namely, TSW scheme). We address key design issues and provide analysis of trust management for software-defined WSNs, including trust recording (i.e., how and what information to record about the nodes), trust evaluation (i.e., how to evaluate the trust score of each node), and trust propagation (i.e., how trust scores are aggregated at each architectural level). Specifically, the main contributions of this research are summarized as follows:

- We propose a hierarchical trust management scheme in which trustworthiness is evaluated at each level of the software-defined WSN architecture. This enables the architecture to have finer detection granularity and early response to malicious behavior. Furthermore, to avoid biased trust scores, trust metrics are computed from reliable scores based on a weighted averaging approach with a defined reliability trust metric.

- To address the decoupled nature of software-defined WSNs, we separate the computation of trust scores to control plane and data plane traffic, respectively. The

separation can provide more critical treatment to the control plane. Also, we utilize specific SDN-based control message types to design an acknowledgment-based trust recording mechanism, resulting in an enhanced detection rate of control dropping attacks.

- We design trust evaluation models that have more sensitivity to the change of abnormal behavior for an enhanced detection time and accuracy. Moreover, we propose a new Bayesian method with reward and penalty factors to increase sensitivity to bad behavior. Furthermore, we design a trust updating mechanism that can dynamically learn from past trust evaluation by giving past bad behavior more weight in the next time-window evaluation.

1.3.2 Secure and Energy-efficient Network Topology Obfuscation for Software-Defined WSNs

Motivations

Network Topology Obfuscation (NTO) is an effective technique for hiding the entire network and securing it against the traffic analysis adversary as a proactive defense. The primary goal of the NTO solution is to minimize the damage level and maximize the cost to the adversary to launch efficient attacks. The damage level determines how successful the attack is, e.g., how many flows may drop down when flooding a link. Network reliability is classed as an essential design concern alongside the security issue in WSN-enabled IIoT networks. WSNs are supposed to be operable for an extended lifetime and should overcome routing challenges. By applying NTO, each node decides to hide the routes of messages inside the network; meanwhile, the communication of the whole network is not harmed. Energy constraint and resource utilization are critical issue in the WSNs [32, 33]. Therefore, any NTO mechanism must consider both the energy and the security issues. In other words, energy consumption and traffic overhead should be highly optimized when applying defense mechanisms. Notably, it is challenging to entirely hide specific nodes in the network, such as the sink node, without forcing excessive overhead. Thus, it is reasonable to consider maximizing the attack cost to the adversary instead of proposing unpractical solutions. Conventional NTO solutions [34, 35, 36, 37, 38] still need improvements to be suitable for the WSNs' application. Different from those proposals, a solution is proposed that jointly considers energy consumption and obfuscation level (defense performance). See section 2.2.3 for a more detailed discussion.

Contributions

In this research, an SDN-based NTO solution is proposed to hide the network from the traffic analysis adversary. SDN allows for the setup of adaptive flow rules in the network. In the proposed solution, all decisions are executed by the controller rather than the sensors themselves. There are no exchanges among sensors, which is the main feature of existing proposals for conventional WSNs. Moreover, the proposed NTO is an energy-aware solution that is especially suited to the characteristics of WSNs and exhibits several advantages contrary to some existing solutions. This research aims to design obfuscation techniques to increase the cost for an adversary to discover the network topology. The proposed techniques aim to secure the network against network traffic attacks such as sniffer attacks, link-flooding attacks, CrossPath attacks [39], and heuristic attacks [38, 40]. Specifically, the main contributions of this research can be summarized below:

- An SDN-based NTO solution is proposed to secure the network against network traffic attacks. The proposed solution provides load balancing between security requirements and resource and QoS restrictions.
- A ranking-based route mutation mechanism is proposed where paths are selected for flows' routes to obfuscate the high-profile nodes in the network. These paths are selected according to several criteria that ensure a high obfuscation level of the network based on route overlapping and consider the compatibility of WSN requirements such as energy consumption and link cost. Moreover, multiple mutated routes can be generated to deceive the adversary and provide additional obfuscation level.
- A sink obfuscation technique is proposed for fully centric WSNs. Multiple fake sink nodes are selected to deceive the adversary who aims to locate the sink node. The selection of fake sink nodes approach jointly considers the residual energy of nodes and the gained obfuscation level. The approach further uses a fitting parameter that considers the residual energy of the selected fake sink nodes' neighbors due to the expected additional communication overhead in their zone.

1.3.3 Security Networking Functions Placement for Software-defined WSNs

Motivations

WSNs support vast ranges of modern applications and are considered one of their primary underlying technologies. WSNs employ a set of typical data functions, such as dissemination or collection, and typical networking functions, such as routing or security [41, 42]. Security networking functions play significant roles in protecting the network from adversaries. Intrusion Detection System (IDS) [43] and reputation-based systems (trust management [44]) are examples of security functions that can be deployed in WSNs. However, networking functions management is a complex process. SDN provides flexibility and programmablity while simplifying the networking functions management [6]. These advantages have boosted the deployment of SDN in WSNs to support solutions for complex processes [5]. Fig. 1.1 presents a software-defined WSN model in which several applications and security services are deployed in the upper layers.

The security network functions can be positioned in access points (switches), dedicated nodes, or sensor nodes [45]. The advantage of applying the security network functions in the access points is to detect malicious behavior against the sensor nodes from outside the network [46]. However, this usually causes query traffic overhead between the border nodes and the network. Having dedicated nodes is not cost-effective and dependent on the deployment area. While applying the security network functions in the sensor nodes might reduce the traffic overhead that arises due to behavioral monitoring. However, the sensor nodes will consume extra energy and use additional computational and memory resources, which introduce further challenges due to the resource limitations of WSNs [47]. To cope with limited resources, less traffic monitoring and less processing capacity must be utilized by placing security network functions across a subset of sensor nodes.

Trust management is a reputation-based solution to detect malicious behavior of sensor nodes using past experiences [48]. Trust management improves the reliability and security of the network by avoiding unreliable nodes and securing multi-hop routing against communication attacks (e.g., selective forwarding and Denial-of-Service (DoS) attacks) [31]. In trust management, sensors are used as watchdog nodes. Watchdog nodes are deployed to provide behavior evaluation for nodes in WSNs [49]. A watchdog node must be able to overhear and communicate with neighbors. These watchdog nodes produce trust reports about their targets. Watchdog function consumes sensors' resources and introduces extra energy consumption. Furthermore, in the active watchdog task, the dedicated node floods target nodes with messages to assess their behavior [50]. This introduces additional energy

consumption from data transmission and overhearing. Most trust management schemes in the literature have overlooked this issue for such limited resource networks and usually consider all nodes watchdog nodes [51]. The solution is to determine a group of sensor nodes to be watchdog nodes while utilizing the network resources for the monitoring tasks. In addition, with the existence of a central controller in software-defined WSNs, the security network function placement is more effective in saving the network energy.

Many challenges arise when selecting the subset of watchdog nodes besides the limitations of energy, computational, and storage resources. First, the degree of watchdog task (i.e., the number of assigned watchdog nodes for each target) must be sufficient for precise behavioral evaluation of the target nodes. Second, an overlapping challenge happens when a target node is assigned to multiple watchdog nodes and vice versa due to the propagation characteristics of wireless signals. Last, unreliable watchdog nodes must not be selected in the subsequent round of watchdog selection.

Contributions

In this research, a secure and energy-aware watchdog placement solution is proposed for software-defined WSNs. The solution balances the required energy consumption and computational resource, and security in terms of the honesty of watchdog nodes. The objective is to minimize the energy consumption yielded due to the watchdog functions while maintaining the activation of sufficient trustable watchdog nodes. To achieve this objective, the problem is formulated as a placement problem where the outcome is the activated nodes that run the watchdog function during the control period. The activated watchdog nodes are selected by considering the energy cost of monitoring the assigned target nodes along with their available computational resources. Moreover, the watchdog performance of these nodes in the previous control periods is considered a security factor (honesty) for placement. Accordingly, a multi-population genetic algorithm is proposed for the optimal placement of the watchdog function in the network, given the comprehensive aspects of resources and security. Our main contributions can be summarized as follows:

- A placement problem formulation is presented to minimize the energy consumption and maximize the defensive gain of the security (watchdog) function for software-defined WSNs. The formulation expresses the degree of watchdog function as the number of activated watchdogs for targets. It determines the coverage and the overlapping challenges. Additionally, the number of target nodes assigned to each activated watchdog node is represented in resource constraints.

- Honesty, a new security metric, is introduced to evaluate the security when selecting nodes for watchdog functions. The metric extends the placement solution to consider the reliability of nodes as watchdog nodes and mouthing attacks where malicious nodes can deceive the trust model by falsifying the trust report. This can happen by praising misbehaving nodes or falsely accusing good nodes. The honesty score is computed based on the deviation of the aggregated trust evaluations. This security metric discourages the controller from selecting malicious nodes for watchdog functions.
- A modified multi-population genetic algorithm is proposed to find the optimal (semi) placement of the watchdog functions that minimizes the energy cost and maximizes the security gain.

1.4 Scholarly Publications

- **M. Bin-Yahya** and X. Shen, "HTM: Hierarchical Trust Management for Software-Defined WSNs," in 2019 IEEE Globecom Workshops (GC Wkshps). IEEE, 2019, pp.1–6.
- **M. Bin-Yahya**, O. Alhussein, and X. Shen, "Securing Software-defined WSNs Communication via Trust Management," IEEE Internet of Things Journal, pp. 1–16, 2021.
- **M. Bin-Yahya** and X. Shen, "SRRM: Ranking-based Route Mutation Scheme for Software-Defined WSNs," in 2021 IEEE Global Communications Conference (GLOBECOM). IEEE, 2021, pp. 1–6.
- **M. Bin-Yahya** and X. Shen, "Secure and Energy-efficient Network Topology Obfuscation for Software-Defined WSNs," IEEE Internet of Things Journal, pp. 1–15, 2022.
- **M. Bin-Yahya**, A. S. Matar, and X. Shen, "Security Networking Functions Placement for Software-defined WSNs," *To be Submitted*

1.5 Thesis Outlines

The remainder of the thesis is organized as follows: In Chapter 2, we provide a comprehensive review of the state-of-the-art security of software-defined WSNs. In Chapter 3, we

design a trust management scheme to detect communication threats in software-defined WSNs. In Chapter 4, we develop two mechanisms to obfuscate the network from traffic analysis attacks, namely route mutation and sink obfuscation. The proposed mechanisms jointly balance the defense gain and energy consumption in software-defined WSNs. In Chapter 5, we design a security function placement solution using the genetic algorithm to cope with the limited resources of software-defined WSNs. Finally, we conclude the thesis and discuss future research in Chapter 6.

Chapter 2

Background and Literature Survey

This chapter presents the background of software-defined WSNs and survey state-of-the-art secure software-defined WSNs in two main aspects: trust management and network topology obfuscation.

2.1 Introduction to Software-defined WSNs

SDN is a promising solution that adds programming capability to WSNs [52]. The SDN is a logically centralized architecture that separates the control plane from the data plane [53]. WSNs have attracted attention for their use of SDN to achieve different benefits, such as flexible and centralized network management [54], efficient routing [55, 56], mobility and localization management [57], and security [58].

2.1.1 Software-defined Networks

SDN is an emerging network technology that attempts to simplify the management task and boost the network performance by enabling network configuration programmatically [59]. The main SDN concept is to decouple the data and control plane while concentrating the network intelligence in one centralized node, which is the SDN controller. Rather than destination-based routing rules in conventional networks, SDN follows flow-based rules. This leads to several benefits, such as vendor reliability, independence, and security [7]. SDN is usually implemented in data centers, cloud computing, campus networks, and wireless networks. In traditional networks, routers and switches have their own routing mecha-

nisms and security. In contrast to traditional networks, SDN offers beneficial features, such as traffic isolation, network monitoring, and dynamic flow control [60]. There are different standards for SDN technology, such as OpenFlow [61] and ForCes (Forwarding and Control element separation) [62]. OpenFlow, developed by Open Network Foundation, is the most widely used model. In OpenFlow, there are three layers of SDN architecture: application layer, control layer, and data layer [61]. Fig. 2.1 shows the general SDN architecture where the data layer consists of SDN-enabled devices that are controlled by the SDN controller in the control layer through the southbound interface. On the other hand, the upper layer contains the applications that interact with the controller through the southbound interface. The data layer or the infrastructure layer is where the forwarding nodes exist. These SDN-enabled nodes have a flow table which is built on the configuration rules from the control layer. Also, the forwarding process takes place according to these flow rules. Thus, the flow table consists of matching rules, actions, and statistical information. In the control layer, the centralized control nodes manage the network through the southbound interface. The controller oversees the whole network and applies the configuration rules. In this layer, different management and processing services are implemented to improve the system's efficiency. The application layer consists of a set of network applications that communicate with the control layer through the northbound interface. Basically, the controller determines the configuration rules for the data layer based on the requirements of the applications. Moreover, the controller may deliver some information from the network nodes to these applications.

2.1.2 Software-defined WSN Architecture

Several software-defined WSN architectures are proposed to integrate SDN with the WSN [63]. As shown in Fig. 1.1 , the infrastructure of the general system model for the software-defined WSN consists of a number of wireless sensors that perform sensing functions and forward messages while the SDN controller controls the whole WSN. Above the controller, there are different network service applications of WSNs. Many studies have proven the feasibility of utilizing SDN technology in WSNs. A complete software-defined WSN system named SDN-WISE [5] is developed and built for multi-hop WSNs. Based on this system, an extended SDN framework for WSN systems is designed using an open and distributed SDN operating system [64]. Even if software-defined WSN reduces the power consumption of the network by utilizing the power and distributing the roles and tasks among sensor nodes, the limited battery life of these nodes is a significant challenge in software-defined WSNs [8]. Also, control messages between the controller and the nodes deplete communication and power resources. As mentioned before, the SDN controller has multi-hop communication

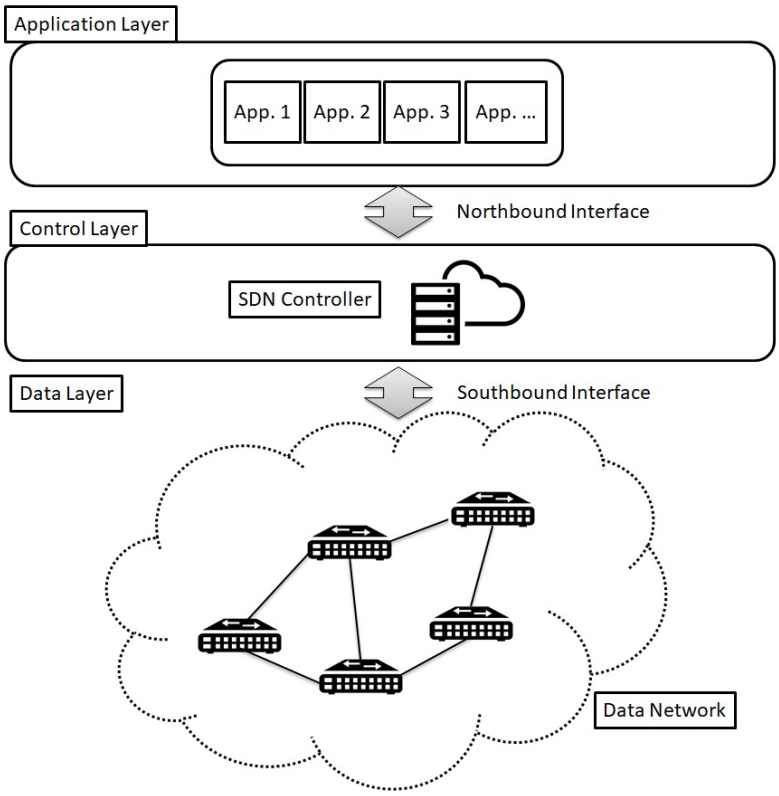


Figure 2.1: General SDN architecture.

with nodes in software-defined WSNs. Therefore, information about the new flow rules must be forwarded and passed through a number of nodes to reach each node. Also, the same applies to the statistical update and other control messages initiated by the sensor node to the controller. This is a challenge because the SDN is designed to have a dedicated logical channel for the control plane [65].

2.2 Security of Software-defined WSNs

WSNs endures several communication threats that attack the functionality of WSNs, such as black-hole, selective forwarding, DoS, and new-flow attacks. Additionally, WSNs are vulnerable to traffic analysis attacks that reveal network topology.

2.2.1 Trust Management

Trust management is a strategy to defend against attacks initiated by authenticated or authorized actors in the network, such as compromised nodes that behave maliciously [66]. Trust management depends on reputation scores which are calculated for target nodes by their neighbor nodes based on their past interactions [67]. SDN platform provides several management services, such as network management. Trust management can be introduced as one of the provided services. In software-defined WSNs, trust management is accomplished in a centralized manner based on the SDN controller. Therefore, the controller determines the flow tables based on the trust scores of nodes by avoiding the lower trust score nodes [68]. Furthermore, the controller performs isolation and mitigation operations against malicious behaviors in the network. In the literature, the sensor nodes are able to overhear all communication exchanges in their range and calculate trust scores accordingly [22]. However, this is not usually the case. Therefore, we need another approach to collect such information toward trust evaluation, such as recommendations. Moreover, the central controller of SDN can help due to its supervision of the overall network [3].

WSN and Trust Management

Securing WSNs through reputation-based schemes is widely proposed in the literature and has proven to be an effective approach for guarding the network against several attacks [69, 22, 23]. Many research studies are proposed to build trust-based services and applications include sensing [70], routing [71], clustering [72], IDS [73], IoT [74], mobile computing [75], transportation [76], and marketing [77]. We discuss some of the recently proposed schemes as follows. Jiang et al. proposed a distributed trust model scheme [24]. In this scheme, communication trust is determined through direct (subjective logic framework) and indirect (trust chain) approaches. Data trust is calculated based on the deviation between the received sensing data from a certain node from the average of other nodes' data in the same area. Energy trust is calculated based on the residual energy of the node under the assumption that a node knows its neighbors' initial energy. Trust reliability and familiarity are used to ensure the precision of recommendation trust. Meng et al. proposed a trust-based scheme with traffic sampling implemented with IDS for IoT-driven WSN [25]. A fixed or random sampling approach is used while considering the computational and energy capability of a limited resources network with a high data traffic rate. To detect malicious behaviors inside the network, a Bayesian-based intrusion detection method is used in which all data flows are expected to be independent. Zhang et al. proposed a trust evaluation approach for clustered WSNs based on cloud model. The model takes into consideration

multiple factors [26]. The approach can be used to meet security requirements according to WSN applications. However, this approach is constrained by the limited power resources of WSNs with lower computational power and incompatible recording approaches. Chen et al. proposed a trust management scheme for IoT networks [27]. The scheme provides adaptive management that is capable of evaluating relationships among IoT node owners to better assess social trust for those devices. Three social trust metrics are presented, namely, a cooperativeness metric based on social ties, an honesty metric based on node reliability, and a community-interest metric to assess co-relationships. Li et al. propose a lightweight and dependable trust management system (LDTS) [28]. In this hierarchical scheme, there are different levels of trust evaluation. The trust scores are calculated based on self-evaluation and Cluster Head (CH) recommendation at the cluster member level. Moreover, each CH evaluates other CHs while introducing the base station as a source of indirect trust evaluation. Sahoo et al. proposed a bio-inspired trust management scheme to provide a secure clustering approach for WSNs [29]. The trust scheme uses a honey bee mating algorithm for trust-based clustering. This approach avoids selecting a malicious node as a CH by creating a list of the nominee nodes associated with trust scores. The algorithm selects the appropriate CH according to this list. Distributed non-SDN-based trust management solutions do not conflict with Software-defined WSNs. However, such approaches do not naturally consider the decoupled nature of the network, nor can they harness the introduced centralized and programmable capabilities. For example, we develop an acknowledgment-based trust recording mechanism that is compatible with SDN-based control message types.

Software-defined WSNs and Trust Management

Research on trust management schemes for Software-defined WSNs is still in a nascent stage [30, 31]. Vishnu et al. proposed a trust-based and QoS-aware routing mechanism for Software-defined WSNs (namely SeC-SDWSN) [30]. The authors define a three-tier architecture of clusters, switches, and SDN controllers. Sensors are clustered using a secure hash tree-based clustering algorithm. Then, an encryption approach is proposed for data security. After that, a fuzzy weighted technique is used to transmit data to the appropriate switch. A more relevant work was proposed by Wang et al. where energy-efficient trust management and routing mechanism (ETMRM). ETMRM considers both the node's trust score and residual energy to detect malicious behavior and ensure secure routing [31]. The authors provide a flow table extension to achieve lightweight trust monitoring at the node level. A Bayesian model is used as a trust computational model. In addition, trust scores are collected from sensor nodes at the controller level to detect and isolate malicious nodes

in a centralized manner. ETMRM's trust scheme does not consider separating data and control traffic evaluation which is an important inherent property when considering SDN-based networks. Moreover, a hierarchical scheme that addresses the inherent architectural levels in Software-defined WSNs is not considered. To this end, we consider and address several research gaps and design factors pertinent to the architecture and properties of Software-defined WSNs which have not been addressed yet. We design a hierarchical trust management scheme on the node, CH, and controller levels. Moreover, we design a trust management scheme that treats the control and data planes separately, which by design can provide higher sensitivity and protection to the crucial traffic of the control channels. Furthermore, we design trust scores that account and preserve historical behavior in the trust updating process. Also, we improve on the existing trust computational models by introducing a new Bayesian factor model with penalty and reward factors to increase the sensitivity to bad behavior, thereby improving the detection rate. We also make sure our trust management scheme is robust to bad/good mouthing and ON-OFF behavior.

2.2.2 Watchdog placement in WSNs

Zhou et al. [50] proposed a stochastic solution to reduce the energy consumption of the watchdog task. They assume that collaborative attacks are launched by closely positioned nodes because they are more likely to be compromised together. Also, these closely positioned nodes consume less energy. Therefore, watchdog nodes are located based on the probability of attacks while considering energy consumption. The energy consumption is a function of the distance between the watchdog and the target nodes. As a result, the objective function is distance-dependent in both energy minimization and security (trust accuracy) maximization. In a realistic scenario, the probability of attacks and the attack type factor are unknown parameters. Moreover, the solution is resource-intensive since there are no limits to the number of activated watchdogs. Finally, the degree, overlapping, and collision challenges are not resolved. Hasan et al. [78] proposed two models to consider overlapping and coverage. In the first model, they limit the overlapping factor for each target node by a certain threshold, while in the second model, the watchdog placement algorithm heuristically aims the complete coverage with minimal overlapping. However, no security factor is used when selecting the watchdog nodes. Moreover, the resource constraints of the WSNs are not considered. Finally, the number of activated watchdog nodes might be too low for both models, affecting the evaluation precision. Zeng et al. [4] proposed generic task scheduling for software-defined WSNs, particularly, energy-aware task scheduling and management strategy for multi-task sensors. The network manager applies flexible task assignments. However, the solution is not specific for watchdog task schedul-

ing, as one activated watchdog node satisfies the solution requirements. Furthermore, no security factor is provided.

2.2.3 Network Topology Obfuscation

Several obfuscation mechanisms are proposed in the literature. These moving target defense solutions aim to secure different types of networks [79, 80]. Obfuscation mechanisms adjust the network's attack surface to maximize the attack cost [81]. However, general SDN-based solutions [82] are not applicable for resource-constrained networks such as WSNs. Solutions for general wireless networks such as multi-hop mobile networks have a high energy consumption [83, 84]. Additionally, most of these solutions require high computational and storage resources. Several research studies are proposed to achieve location privacy in conventional WSNs using cryptographic approaches [85]. However, these approaches fail against adversaries that launch traffic analysis attacks. There are several proposed defense techniques against traffic analysis attacks in WSNs [20]. There are two types of solutions: non-centric and centric techniques. The non-centric solutions (standalone or cooperative) have a higher rate of energy consumption and a higher cost (in terms of path length, end to end delay, etc.). For example, the probability-based routing protocols in [40] rely on broadcasting fake packets from fake sources concurrently with the transmission of real packets from the real source nodes to deceive the adversary. In addition, the discovery of alternative routes requires more broadcasting messages as deceptive traffic or to collect the relevant information from neighbor nodes; this results in additional energy consumption and higher overhead [86]. Random walk protocols [36] deliver the messages through a random route every time. Ring routing, an improved version of random walk protocol, is proposed in [87]. Liu et al. [87] proposed a Multi-Representative Re-Fusion (MRRF) data collection technique. MRRF is designed to ensure acceptable energy consumption and end-to-end delay of the random walk ring routing. However, the technique has demonstrated some performance limitations. The technique considers only energy factors, and no security constraints are introduced.

The existing centric solutions such as SDN-based solutions have their limitations. Duan et al. [35] proposed a proactive random route mutation mechanism against sniffer and DoS attacks. The selected routes are dynamically and randomly changed while preserving QoS end-to-end connectivity. However, multiple uncrossed routes for each flow are required. The mechanism ensures that a previously selected route consisting of certain links must not be selected for the current route. It is challenging to satisfy this requirement in WSNs topology. Zhou et al. [34] proposed Scalable Node-Centric Route Mutation (SNcRM) technique that formulates the problem into a signature matching problem and solves it by

a binary branch and bound method. The technique obfuscate the network topology by decreasing the variety of historically accumulative traffic volume among the SDN-enabled nodes. The SDN controller recognizes previously highly loaded nodes with high exposure risks and finds alternative routes for their traffic flows via other lowly loaded nodes. This is determined based on the accumulative traffic of the node. Rauf et al. [37] proposed Secure Route Obfuscation (SRO) scheme which is an SDN-based solution. To obfuscate the network, the controller randomly generates random routes for each communicating pair of nodes. Only a reliability score proposed in [88] is considered toward the route selection. No energy or security constraints are designed to select routes. Thus, SRO shows weak performance in terms of energy and security when applied to WSNs due to the uncontrolled randomness. Chai et al. [89] proposed a sink obfuscation technique against a global adversary based on the k-anonymity model. At least k nodes are selected in the network to mimic the sink node. Thus, the traffic loads around these nodes make the sink area indistinguishable. However, due to the significant overhead of the fake sink nodes' deceptive traffic, this solution has comparatively costly energy consumption. The network's lifetime is decreased due to every node sending messages to these fake sink nodes when sending the real one. Moreover, the authors do not consider acknowledgment (ACK) messages. This limits the solution to not be applied to software-defined WSNs, which have many built-in ACK-based messages. In addition, the data aggregation nodes are the same selected fake sink nodes, thereby leading to a traffic collision and low message delivery and reliability. Baroutis and Younis [38] proposed Preserve Location Anonymity through Uniform Distribution of Traffic volume (PLAUDIT) technique to obfuscate WSNs. This technique achieves uniform distribution of traffic volume through injecting deceptive messages. Several dedicated nodes are selected to generate the deceptive flows that challenge the adversary's mission to reveal the network topology. In addition, the deceptive traffic rate is determined to balance the traffic density across the network and avoid network overhead. The authors try to achieve robust anonymity with load balancing and energy consumption. However, PLAUDIT fails to hide the centralized architecture of software-defined WSNs, especially from the heuristic traffic analysis attacks. Many research studies [90] define mutated identification methods. However, for WSNs, changing IP addresses too frequently may cause serious ramifications, including service interruptions, routing inflation, delays, and security violations [91]. Moreover, these techniques do not secure the network from a traffic analysis attack, for example, when the adversary uses physical equipment to collect the network information. All solutions described above attempt to obfuscate network topology or enhance the anonymity of certain high-profile nodes. Nonetheless, different from those countermeasures, in this thesis, a balance between energy consumption and obfuscation level (defense performance) is achieved. The proposed mechanisms minimize energy consumption, prolong the network lifetime, and suffer lower attack success rates.

Chapter 3

Securing Software-defined WSNs Communication via Trust Management

In this chapter, a trust management scheme for software-defined WSNs is proposed (namely, TSW scheme). The goal is to secure software-defined WSNs from nodes that behave maliciously to perform several attacks, e.g., black-hole, selective forwarding, and DoS (new-flow). We address key design issues and provide analysis of trust management for software-defined WSNs, including trust recording, trust evaluation, and trust propagation.

3.1 Problem Formulation

In this section, we present the software-defined WSN system model as well as network and security assumptions. Then, we describe several potential threats in software-defined WSNs. Finally, we discuss the design goals of the TSW scheme.

3.1.1 System Model

The system model consists of several components, namely sensor nodes, CHs, sink nodes, and the SDN controller, as shown in Fig. 3.1. Three hierarchical levels exist in this model, namely the node level, the CH level, and the controller level. We assume that network nodes are deployed randomly and are homogeneous (i.e., every sensor node has the same

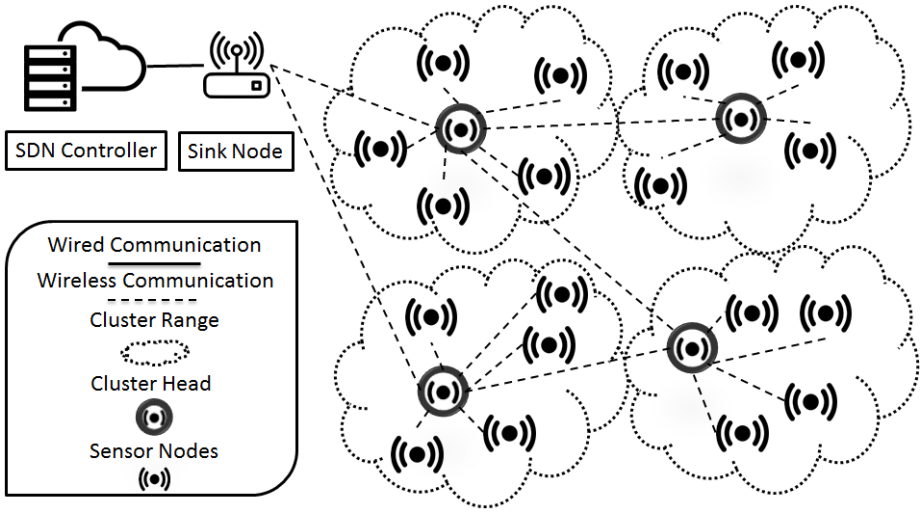


Figure 3.1: Software-defined WSN system model.

communication range). Also, each node in the network has a fixed position. Each sensor node must have at least two one-hop neighbors in its communication range. Thus, the network can be very dense. Nodes in the same communication range can detect, communicate, and overhear each other. For each cluster, the CH aggregates and relays messages inside the network. The controller has a global view of the network. It has a high computational capacity and unlimited communication resources compared to other components in the system. The controller receives statistical updates from the underlying network components to build comprehensive network maps. These statistical reports are collected through a number of SDN management messages. The controller can provide several services efficiently, such as routing, network management, and security based on these maps. Each node has a flow table, and the SDN controller is responsible for updating the flow rules. The flow table consists of matching rules, actions, and statistics fields.

3.1.2 Security Assumptions

We assume that each node can be identified by a unique legal ID. The network applies defense mechanisms to deal with replay, Sybil, and identity-based attacks [92]. Every node (including malicious nodes) is authorized and authenticated. The communication inside the software-defined WSN is encrypted using a shared key. Thus, unauthorized nodes cannot eavesdrop nor be an active part of the network. Also, every network node has another shared key with the SDN controller. Thus, no one is able to modify the message

content between a component and the controller, such as to forge statistical updates, flow rule control messages, or trust update messages. Moreover, we assume that sink nodes and the controller are fully trusted. Sensor nodes and CHs can be malicious. We assume that there is no attack at the initial network setup.

3.1.3 Threat Models

Software-defined WSNs face several security challenges and vulnerabilities due to their open environment and limited resources [65]. Thus, sensor nodes can be compromised physically or remotely and become malicious by an outsider attacker. Also, some nodes, such as selfish nodes, can act maliciously to preserve their limited resources, such as energy.

Black-hole Attack

A malicious node drops all received messages received from its neighbors.

Selective Forwarding Attack

A malicious node drops received messages partially either at random or deliberately. A malicious node can launch this attack to block certain types of messages or certain node's messages, or degrade the network's message delivery ratio. On the other hand, if a selfish node launches the attack, dropped messages can be random.

Fig. 1.2 demonstrates an aggravated effect of dropping messages in software-defined WSNs. The bottom (blue) attacker drops the received control messages from/to the bottom part of the network, isolating it from connecting with the controller. In addition, both black-hole and selective forwarding attacks can be launched on the data plane.

DoS and New-flow Attacks

The new-flow attack is a flooding DoS attack that targets the control plane of SDN-based networks. However, in software-defined WSNs, this attack can target both the control and data plane due to the large number of packet-in messages, which can degrade the network's availability. For example, in Fig. 1.2, the upper (red) malicious node floods a node in the control plane with packet-in messages. The control path starting from this node will be degraded in terms of bandwidth, overhead, and nodes' energy. As a result, the availability

of the upper (red) part of the network will be affected. This attack can also be directed at the availability of the controller.

ON-OFF Attack

A malicious node launches the communication attacks irregularly. In so doing, a malicious node attempts to deceive trustworthiness metrics by behaving normally following a bad behavior.

Good/Bad Mouthing Attack

A malicious node attempts to falsify the aggregated trust for a particular node by providing biased scores. A malicious node sends a low trust score for a non-malicious node in a bad mouthing attack, which would affect the non-malicious node's trustworthiness at higher levels in the software-defined WSN. On the other hand, in a good mouthing attack, malicious nodes try to raise the trust of other malicious nodes by sending good feedback. Potentially, a number of malicious nodes can also launch a collaborative mouthing attack.

3.1.4 Design Goals

In TSW, we aim to provide a security mechanism to defend against the aforementioned attacks. The main design goals of this work are listed below:

Simplicity and efficiency: The trust scheme must have lightweight operations due to the limited resources of sensor devices. At the same time, the trust scheme needs to accurately detect and isolate malicious nodes from the network services. It can be challenging to develop lightweight models that are sufficiently sensitive to malicious behavior. Trust computational models are to ensure that the trustworthiness of a node must fall quickly following bad behavior yet rise slowly following good behavior.

Compatibility with software-defined WSNs: There are three hierarchical levels in software-defined WSN, thereby the need for a hierarchical trust management scheme. And, there should be separate data and control trust scores. Besides, it is instructive to utilize built-in SDN messages for the recording phase, while statistical information can be collected from existing flow-table constructs.

Dynamicity and timeliness: The computed trust score for a particular node at each level must be dynamic. Moreover, node reliability must be reflected correctly in the change of

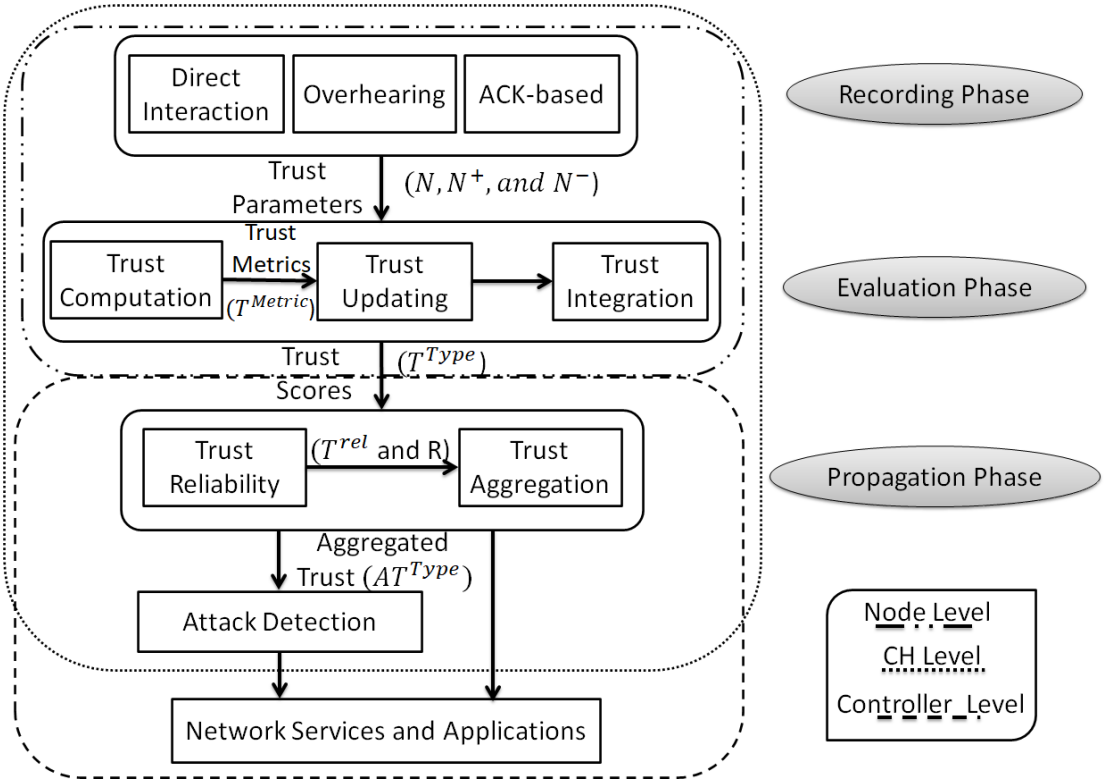


Figure 3.2: TSW scheme overview.

its trust score over time. Furthermore, the past behavior of a node needs to be considered when continuously updating its respective trust score.

Resilience and honesty: Unreliable scores need to be avoided from the trust evaluation process in higher levels to reduce the effect of biased scores.

3.2 TSW Scheme Overview

Fig. 3.2 presents an overview of the TSW scheme which consists of three main phases, namely recording, evaluation, and propagation. Trust parameters (in subsection 3.3.1) are collected and recorded in the recording phase through direct interaction, overhearing, and ACK-based approaches. Next, based on these parameters, trust metrics are computed through trust computational models (in subsection 3.3.2). The trust scores are updated with the consideration of past scores (in subsection 3.3.2), and integrated together through-

out every time window (Δt) (in subsection 3.3.2). After that, the computed trust scores are aggregated at the upper layers of the SDN architecture (in subsection 3.3.3), where the trust reliability is provided, and biased scores are excluded in the computation of the aggregated trust score (in subsection 3.3.3).

Table 3.1: Table of Notations for Trust Management Problem

Notations	Descriptions
$T_{x,y}$	Trust score computed by node x for node y.
T^{Metric}	Trust metric defined in section 3.2.1.
T^{Type}	Trust score computed through integration (3.3.2).
N_y	Number of messages initiated by node y.
N_y^+	Number of successful messages forwarded by node y.
N_y^-	Number of unsuccessful messages forwarded by node y
th_H	Trust score threshold of being trusted ($T \geq th_H$).
th_L	Trust score threshold of being untrusted ($T \leq th_L$).
η_{max}	Maximum number of messages that a node can initiate.
η_{min}	Minimum number of messages that a node can initiate.
AT_{avg}	Average of received trust scores.
AT	Aggregated trust score.
R	Score reliability.
T^{rel}	Node reliability.

3.2.1 Trust Metrics

In TSW scheme, we define six trust metrics to evaluate the trustworthiness of nodes. These metrics are computed based on the sending and forwarding parameters from the recording phase 3.3.1. The trustworthiness evaluation is conducted to determine the participation and cooperation of every node in the forwarding process. These metrics can differ based on the application of trust management [22]. Our scheme aims to secure network communications. We assume that node x is the evaluating node and node y is the evaluated node.

(1) Forwarding trust: Denote the forwarding trust metric for data traffic and control traffic by $T_{x,y}^{DF}$ and $T_{x,y}^{CF}$, respectively. These trust metrics evaluate how a node cooperates in the forwarding process of data and control messages, respectively. Such metrics are needed

to detect message dropping attacks. They are computed by considering successful and unsuccessful experiences of participation through forwarding counters (subsection 3.3.1).

(2) Sending-rate trust: Denote the sending-rate trust metric for data traffic and control traffic by $T^{DS}_{x,y}$ and $T^{CS}_{x,y}$, respectively. These trust metrics evaluate whether or not the sending rate of data and control messages of a node is within the limit, respectively. Such metrics are needed to detect flooding attacks. If the number of initiated messages exceeds the maximum limit, the trust score must decrease to penalize node y. However, $T^{CS}_{x,y}$ also has a lower limit as the number of expected control messages can be determined (section 3.3.2). $T^{DS}_{x,y}$ and $T^{CS}_{x,y}$ are computed based on sending rate parameters (section 3.3.1).

(3) New-flow trust: This trust metric, $T^{NF}_{x,y}$, evaluates if the sending rate of packet-in control messages from a node is under the maximum limit. The packet-in messages are crucial because they can be used to launch a DoS attack in the SDN-based architecture. If the number of initiated messages exceeds the limit, the trust score must decrease. Similar to $T^{CS}_{x,y}$ and $T^{DS}_{x,y}$, the new-flow trust metric can computed through direct interaction and overhearing approaches.

(4) Node reliability: We need to determine a node trustworthiness when it sends or propagates trust scores to the upper layer. That is, we need a metric to defend against mouthing attacks which can be computed by CHs and the controller. The node reliability metric, T^{rel}_{y}, is determined based on the aggregated trust scores from node y about its neighbors. A node becomes trusted when it provides reliable trust scores (subsection 3.3.3).

3.2.2 TSW Architectural Levels

Next, we describe the trust management process at every software-defined WSN architectural level, namely node, CH, and controller levels.

Node Level

Algorithm 1 illustrates the trust management process at the node level. Each node builds a record for its neighbors based on the recording approaches. The recorded parameters (counters) are used in the computation of trust metrics. These trust metrics are combined to build specific direct trust scores. Thus, for each time window Δt, nodes store the calculated trust scores for future use in the trust updating process. As well, these scores are sent to the upper level. According to [24], Δt should not be very small because this would mean frequent trust computation and updating, which would lead to power

Algorithm 1: Trust Evaluation Algorithm at Node Level

while t ***during*** Δt **do**
  if *E (event) triggered for node y by node x* **then**
    $N^{E}_{x,y} = N^{E}_{x,y} + 1$;
    if *E is positive* **then**
      $N^{E+}_{x,y} = N^{E+}_{x,y} + 1$;
    else
      $N^{E-}_{x,y} = N^{E-}_{x,y} + 1$;
    end
  end
end
for *$y \in Y$ (neighbor nodes)* **do**
  $T^{m}_{x,y} =$ **metric_model**$(N^{E}_{x,y}[, N^{E+}_{x,y}, N^{E-}_{x,y}])$, $\forall m \in Metrics$;
  $T^{m}_{x,y} =$ **update**$(T^{m}_{x,y}(t), T^{m}_{x,y}(t - \Delta t))$, Eq. 3.3;
  $T^{ty}_{x,y} = \sum_{m} w_m T^{m}_{x,y}$, $\forall ty \in Types$;
end
send$(T^{ty}_{x,Y}, CH)$, CH is the CH of nodes' cluster;

consumption. Furthermore, the trust evaluation will be affected by non-malicious causes such as congestion and delay. On the other hand, the trust score should not be computed over a very long period. As a result, the trust evaluation would not reflect the most current state of the nodes' trustworthiness.

CH Level

Algorithm 2 illustrates the trust management process at the CH level. At this level, CHs evaluate the trust of nodes in two directions. The first direction is the trust evaluation of their neighbors in the same communication range (node level). Additionally, every CH has to record and evaluate the trustworthiness of other CHs and report it to the controller. The second direction is that CH aggregates the trust update messages from cluster nodes and computes cluster-based trust aggregation and trust reliability of cluster nodes. These computed trust scores are sent to the controller. CHs can provide a quick response when malicious nodes are detected via aggregated trust score $AT^{ty}_{C,Y}$. Here, the cluster head can stop forwarding packets from/to the malicious node and report such events to the controller.

Algorithm 2: Trust Evaluation Algorithm at CH Level

node_level_function();
for $x \in C$ *(Cluster)* **do**
  $T^{ty}_{x,Y}$ = **receive**(x), $\forall ty \in Types$;
end
$[R_{X,Y}, T^{rel}_{X}]$ = **reliability**$(T^{ty}_{X,Y})$, Eq.s 3.5 and 3.6;
$AT^{ty}_{C,Y}$ = **aggregate**$(T^{ty}_{X,Y}, R_{X,Y})$, Eq. 3.8;
forward$(T^{ty}_{X,Y}, controller)$;
if $AT^{ty}_{C,y} \leq th_L$ **then**
  report$(AT^{ty}_{C,y}, controller)$;
end

Controller Level

Algorithm 3 illustrates the trust management process at the SDN controller level. The controller, which has a supervisory view of the network, computes each sensor's overall trust scores. The trust scores, which are computed at the node level, are received by the controller. Based on these trust scores, the controller analyzes and combines these scores for each node. To this end, the controller computes the global trust score for each node from the collected trust scores. Again, an outlier detection mechanism is used to ensure that the global aggregated trust is not affected by the mouthing attacks. The trust matrices at the controller have a broad sight because it includes the trust scores evaluated for nodes that are not from the same cluster. As a result, the controller determines the trust level of every node and decides which nodes can be trusted to provide the different network services. At the same time, the untrusted nodes are prevented from participating in the network operations. Therefore, the flow table updates depend on the collected trust scores.

3.3 TSW Scheme Phases

In this section, we provide details about the three phases of the TSW scheme. First, we describe the trust recording approaches and the recorded trust parameters. Next, we define the trust computational models for trust evaluation in addition to the trust updating and integration mechanisms. Finally, we give details about the propagation phase where

Algorithm 3: Trust Evaluation Algorithm at Controller Level

for $x \in$ *All nodes of the network* **do**
  $T^{ty}_{x,Y}$ = **receive**(x), $\forall ty \in Types$;
end
for $c \in C$ *(Clusters)* **do**
  $T^{ty}_{c,Y}$ = **receive**(c);
end
$[R_{X,Y}, T^{rel}_{X}]$ = **reliability**$(T^{ty}_{X,Y})$, Eq.s 3.5 and 3.6;
$AT^{ty}_{All,Y}$ = **aggregate**$(T^{ty}_{X,Y}, R_{X,Y})$, Eq. 3.9;
network_services$(AT^{ty}_{All,Y})$;

the trust aggregation takes place at the CH and controller levels while considering the reliability of these aggregated trust scores.

3.3.1 Trust Recording Phase

Each node monitors and records other nodes' behavior in the same communication range in the recording phase. Thus, the recording process of trust information is accomplished by overhearing neighboring nodes during the message interactions and transmissions. The recorded information is learned from the interactions between the evaluated node and the evaluating node and the interactions between the evaluated node and other nodes in the same communication range. Moreover, we use direct interaction counters in the Statistics fields of the flow table as part of the recording process. As well, some software-defined WSN control messages can be used to evaluate intermediate nodes' behavior. For example, receiving a packet-out message means that the intermediate nodes have delivered the packet-in message successfully.

Forwarding Parameters (N^+_y, N^-_y)

Using these parameters, the number of successful and unsuccessful forwarding messages by a particular node y is recorded. Three cases of recording these parameters are defined in the TSW scheme.

Overhearing Approach: In the first overhearing case, as shown in Fig. 3.3(a), when node x sends a message to node y that is not the destination, node x overhears the transmissions

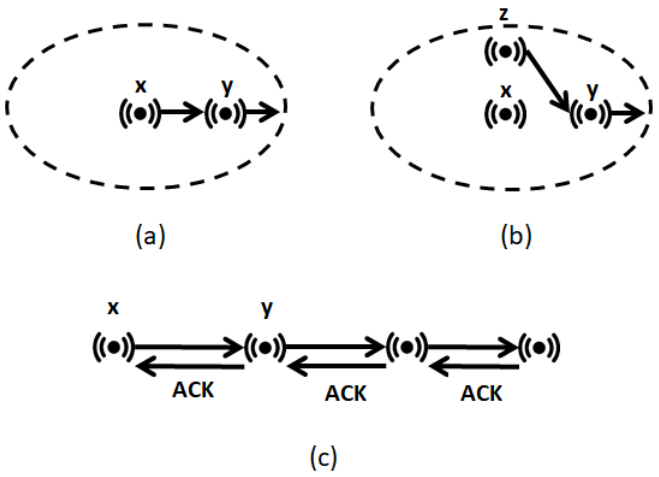


Figure 3.3: Recording approaches of forwarding parameters; (a), (b) the two overhearing cases and (c) the ACK-based case.

of node y to check if node y forwarded the message. If the node successfully forwarded the message, then the counter N_y^+ is updated by adding 1. If not, then the other parameter counter N_y^- is updated by adding 1. The second overhearing case is shown in Fig. 3.3(b). Hence, when node z sends a message to node y (node y is not the destination) and node x is a watchdog for both nodes, then node x overhears node y's transmissions to check whether or not node y forwarded the message successfully. Accordingly, N_y^+ and N_y^- parameters are updated as in the previous case.

ACK-based Approach: In SDN-based networks, nodes send updates between the controller and sensor nodes in the control plane. In the TSW scheme, we utilize these types of messages to enrich the trust recording for the control traffic. For example, node x sends a message to the controller through node y, as shown in Fig. 3.3(c). If node x received an ACK or any response confirming that the message is forwarded successfully by node y, the counter N_y^+ is updated by adding 1. Otherwise, if no ACK or response is received, the counter N_y^- is updated by adding 1. The packet-out message is an instance of a response for successfully forwarding a packet-in message. Thus, if node x sends a packet-in message to the controller through node y, and the packet-out message is received by node x, node y performed a successful forwarding, and the success counter is increased by 1.

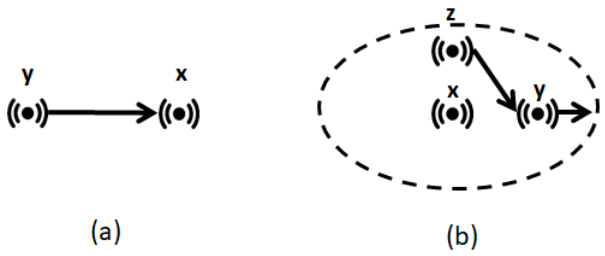


Figure 3.4: Recording approaches of sending parameters; (a) direct interaction and (b) overhearing cases.

Sending Parameter (N_y)

This parameter counts the number of initiated messages by a particular node y (N_y). Two approaches are used to record these counters.

Direct Interaction Approach: The first case is the direct interaction approach, where a node receives messages from its neighbors directly. From Fig. 3.4(a), when node x receives a message from node y, the counter N_y is increased by 1. If node y is not the source of the message and the source is node z, and it has not already been recorded through overhearing, then the counter N_z is increased by 1. In software-defined WSNs, the statistics of direct interaction can be retrieved directly from the statistical part of the flow table.

Overhearing Approach: In this case, as shown in Fig. 3.4(b), when node x overhears node y's transmissions, if node y sends a message, the counter N_y is increased by 1; If node z is the source of the message, and the message has not been recorded, the counter N_z is increased by 1. However, if node x is a watchdog for nodes y and z, and it records that node z sent that message to node y, then the counter N_y is decreased by 1. Moreover, we define a specific counter of this type to count the number of forwarded packet-in messages.

3.3.2 Trust Evaluation Phase

In the trust evaluation phase, each node executes trust computation models based on the parameters resulting from the recording phase. Each type of trust metric has its features, therefore evaluation model. As a result, each of these metrics must be computed using different trust computational models. We divide the trust score $[1, 0]$ into three zones, namely trusted $[1, th_H]$, uncertain $[th_H, th_L]$, and untrusted $[th_L, 0]$. These trust score

boundaries can be adaptive based on the two sets of trusted and untrusted nodes and the total number of nodes that contain trusted, trusted, and uncertain nodes [93].

Success/Fail Model

Trust evaluation of data forwarding ($T^{DF}_{x,y}$) and control forwarding ($T^{CF}_{x,y}$) metrics depends on successful and unsuccessful hits of the measured property. The Bayesian method is used to compute these types of trust metrics [94]. Additionally, we consider rewarding the successful hits and penalizing the unsuccessful hits. Therefore, rewarding and penalty factors are defined as shown in equation 3.1 which is inspired by [29]. The rewarding term is used to ensure a slow rising of the trust scores and reward credit for the number of successful hits. The penalty factor is used to ensure a fast falling of the trust scores when there are unsuccessful hits and punish the failure. As a result, the falling and rising behavior are less sensitive to the uncertain area. Equation 3.1 combines the Bayesian method with the rewarding and penalty factors in the trust computation as follows:

$$T^{Metric}_{x,y} = \frac{N^{+}_{x,y} + 1}{N^{total}_{x,y} + 2} \frac{N^{+}_{x,y}}{N^{+}_{x,y} + 1} \frac{1}{\sqrt{N^{-}_{x,y} + 1}} \tag{3.1}$$

where x is the evaluating node which calculates the trust scores and counts the number of success and failure hits, and y is the evaluated node by node x. $N^{+}_{x,y}$ is the successful count that node x has recorded about node y. $N^{-}_{x,y}$ is the number of failure hits that node x experiences with node y. $T^{Metric}_{x,y}$ is the trust metric computed by x for y. In equation 3.1, the first term is the Bayesian factor and then the rewarding and the penalty terms, respectively.

Threshold-limit Model

The trust evaluation of data sending-rate ($T^{DS}_{x,y}$), control sending-rate ($T^{CS}_{x,y}$), and new flow ($T^{NF}_{x,y}$) trust metrics is done based on a threshold limit. For example, sending a large number of packet-in messages attacks the control line between nodes and the SDN controller. In this case, each node counts the number of received parameters, e.g., the number of initiated packet-in messages from a particular node. Thus, a defined threshold η for this parameter should not exceed a specific limit. Exceeding the threshold limit must degrade the trust score. Threshold η can be defined as the maximum number of expected messages from the sending node or the maximum capacity of received messages by the receiving node. We set this to be the initiated number of data and packet-in messages.

In the proposed threshold-limit model, the trust score starts degrading at η_1 when the N_y approaches the threshold value where $\eta > \eta_1$. The trust score degrades linearly from full trust value at η_1 to (1/2) when N_y is equal to η. Another drop starts from η to η_0, at which point the trust score reaches the zero value. Thus, the proposed threshold equation is as follow:

$$T_{x,y}^{Metric} = \begin{cases} 1 & , N_y \leq \eta_1 \\ 1 - \frac{\eta_1 - N_y}{2(\eta_1 - \eta)} & , \eta_1 < N_y \leq \eta \\ \frac{N_y - \eta_0}{2(\eta - \eta_0)} & , \eta < N_y \leq \eta_0 \\ 0 & , \text{otherwise} \end{cases} \tag{3.2}$$

Basically, the distance between η and η_1 must be greater than the distance between η and η_0 because the trust score must drop faster and penalize more when N_y exceeds the threshold value. Hence, the threshold points η_1 and η_0 are calculated as follows: of $\eta_1 = \frac{\eta}{2}$ and $\eta_0 = \eta + \frac{\eta}{4}$. Note that η is set by the network manager based on the network statistics or prior knowledge of the expected maximum limit of the number of initiated messages inside the network [95]. This threshold must be lower for a network with a limited bandwidth than for a higher bandwidth network.

Moreover, the trust evaluation of control sending-rate trust metric ($T_{x,y}^{CS}$) depends on lower and higher limit thresholds as it should not exceed a specific limit η_{min} or fall below another specific limit η_{max}. The trust score of this node must be degraded when going above or below these two threshold limits. The lower limit threshold η_{min} is defined for control messages due to the properties of software-defined WSNs where nodes in this network have to send symmetric messages (e.g., statistical information) periodically. Thus, the node is expected to send a certain number of this type of message. A selfish node, which tries to save its residual energy, may refrain from sending these messages. Similarly, thresholds η_{min} and η_{max} are set by the network manager based on the network statistics. The same threshold-limit model is used to determine the trust score based on the lower threshold with few modifications of inequalities and threshold points $\eta_1 = \frac{3\eta}{2}$ and $\eta_0 = \eta - \frac{\eta}{4}$.

Trust Updating

To consider the historical trust scores, the new trust score at time t is calculated by combining the current trust score during the time window (Δt) and the previously calculated trust score at time ($t - \Delta t$). Therefore, we use an improved trust updating mechanism to

compute the trust as follows:

$$T(t) = \begin{cases} (1-\alpha)T(t) + \alpha T(t-\Delta t) \\ \qquad\qquad\qquad \text{, if } T(t) \leq T(t-\Delta t). \\ (1-(\alpha+\beta))T(t) + (\alpha+\beta)T(t-\Delta t) \\ \qquad \text{, if } T(t) > T(t-\Delta t) \text{ and } \alpha+\beta < 1. \end{cases} \tag{3.3}$$

where $T(t)$ is the new trust score computed for the current window Δt at time t, while $T(t-\Delta t)$ is the previous trust score calculated in the previous window at time $(t-\Delta t)$. T is $T_{x,y}^{Metric}$ for all symbols. α is a decay factor determining the balance between the current and previous calculations of trust. β is the newly defined decay factor which gives more weight to the old trust score when the current trust score is greater than the old score. In other words, the trust scheme relies more on the old trust score when the node has behaved maliciously in the past.

Trust Integration

Each node computes the trust score of other nodes in every Δt. Each separate trust score depends on a different number of trust metrics. For example, control trust is computed based on control sending rate and control forwarding trust metrics. Having a separate control trust score from the data trust score enhances the ability to detect attacks that target the control plane and network availability. The equation below gives the weighted linear approach to compute the trust scores by combining trust metrics:

$$T_{x,Y}^{Type}(t) = \sum_{m \in Metrics} w_m T_{x,Y}^{m}(t) \tag{3.4}$$

where $Type$ represents the separate trust scores for specific applications such as data and control trust scores. $Metrics$ are the trust metrics used to calculate the specific trust score. However, for each trust score type, not all trust metrics are combined or have the same impact. Therefore, a weighting parameter is used for every trust metric to determine its weight. In the equation, $0 < w_m < 1$ and $\sum_{m \in Metrics} w_m = 1$. For example, if a metric m is not related to a specific trust type, its weight w_m will be $zero$; hence, it will not be included in that trust type evaluation. On the other hand, it is possible to have a trust type computed through only one trust metric. Assigning the weighting parameters depends on a specific application to utilize the trust management performance.

3.3.3 Trust Propagation Phase

In this section, we describe our defense mechanism against mouthing attacks through computing trust reliability. Furthermore, we describe how the trust aggregation takes place at CH and controller levels.

Trust Reliability

The CHs and the controller must validate the aggregated trust scores as some nodes can perform the mouthing attacks. In TSW scheme, two reliability scores are defined, namely, score reliability and node reliability. The node reliability (T_x^{rel}) is similar to the recommendation trust metric, which evaluates the trustworthiness of a node's positive or negative recommendation of other nodes. However, the score reliability ($R_{x,y}$) is used to detect an outlier from the aggregated trust scores at CH and controller levels.

To compute these reliability scores, first, the average trust score ($AT_{avg,y}$) of collected trust scores of a certain node y is computed as a base point. Then, the trust reliability of each trust score of node y received by the CH or controller from a node x (score reliability) is computed as follows:

$$R_{x,y}(t) = 1 - |T_{x,y}(t) - AT_{avg,y}(t)| \tag{3.5}$$

where the resulting value of $R_{x,y}$ is used to compute the aggregated trust score in next subsection. Again, score reliability only determines the weight that must be considered for a single trust score. The score reliability of each trust score is determined separately. However, these values do not determine the evaluation trustworthy of node x. Thus, the node reliability can be calculated by using the score reliability as follows:

$$T_x^{rel}(t) = \frac{\sum_{y \in Y} R_{x,y}(t)}{N_Y} \tag{3.6}$$

where N_Y is the number of neighbor nodes. If the calculated node reliability (T_x^{rel}) of node x is in the trusted zone as defined in section 3.3.2, then this node collects scores at that level which can be used toward computing trust aggregation. The node reliability has the exact updating mechanism described in equation 3.3. To conclude, the node reliability score is considered a special trust score type. This score evaluates the node's trustworthiness when evaluating others. Thus, this trust score is used to detect mouthing attacks and avoid biased trust scores. Consequently, the controller excludes trust scores submitted by a node that has a low reliability score. This isolation guarantees that bias nodes are not capable of affecting the detection efficiency of TSW scheme.

Trust Aggregation

Due to the hierarchical architecture, the trust scores are aggregated at CH and controller levels. At the CH level, the aggregated trust score value is calculated for each node in the cluster. Then, the controller also collects trust scores aggregated by CHs to compute the node's trust evaluation. Usually, to combine the collected trust scores, the average is computed to represent the final trust as follows:

$$AT_{avg,y}(t) = \frac{\sum_{x \in X} T_{x,y}(t)}{N_X} \tag{3.7}$$

where $AT_{avg,y}$ is the average aggregated trust score for the node y, which is calculated by averaging the received trust scores of node y from different nodes in X. X is a one-dimensional array of nodes that are able to evaluate the trust of node y. N_X is the total number of x's in X. However, both CH and controller must validate these collected trust scores from the sensor nodes as some nodes can be biased and perform mouthing attacks. To detect any outliers in the aggregated trust scores, we use a modified Averaged Difference Algorithm from spatial weighted outlier detection presented in [96].

Therefore, CH evaluates the trustworthiness of nodes in its cluster, and each cluster node sends the trust scores it computes about its neighbors to the controller. CH computes the cluster-based aggregated trust of each cluster node by averaging the receiving trust scores with weights, where the weight of each trust score is computed by trust reliability:

$$AT_{C,y}(t) = \frac{\sum_{x \in C} T_{x,y}(t) R_{x,y}(t)}{\sum_{x \in C} R_{x,y}(t)} \tag{3.8}$$

where $R_{x,y}$ (score reliability) is a value between $[0, 1]$ which is defined in equation 3.5. Using this approach in equation 3.8, the aggregated trust score is calculated mainly from the reliable nodes by giving the outlier evaluation less weight to be included in the calculation. In equation 3.7, the exact weight is given for all collected trust scores. This leads to the trust score being easily affected by the mouthing attacks. This makes our approach a more realistic approach than the averaging approach in equation 3.7.

However, at the CH level, the trust scores are collected from the cluster nodes only. By comparison, the controller collects trust scores from every node in the network. This gives the controller the advantage to evaluate the trustworthiness of any node from a broad sight and increases its ability to detect any malicious behavior. Therefore, the global aggregated trust score is computed as follows at the controller level:

$$AT_{All,y}(t) = \frac{\sum_{x \in All} T_{x,y}(t) R_{x,y}(t)}{\sum_{x \in All} R_{x,y}(t)} \tag{3.9}$$

where $AT_{All,y}$ is the global trust score for node y and is computed by the SDN controller. This score is used by the controller to detect any malicious behavior of node y. Also, it determines the credibility of this node's participation in different network services and applications.

3.4 Trust Model Analysis

In this section, we provide an analysis of the trust evaluation models presented in the previous section 3.3. The analysis is conducted to study the behavior of the trust score under these models. Note that all evaluation is implemented using Matlab. We study the behavior of the success/fail model, the threshold-limit model, the updating mechanism, and the aggregation approach with some existing models.

3.4.1 Analysis of Success/Fail Model

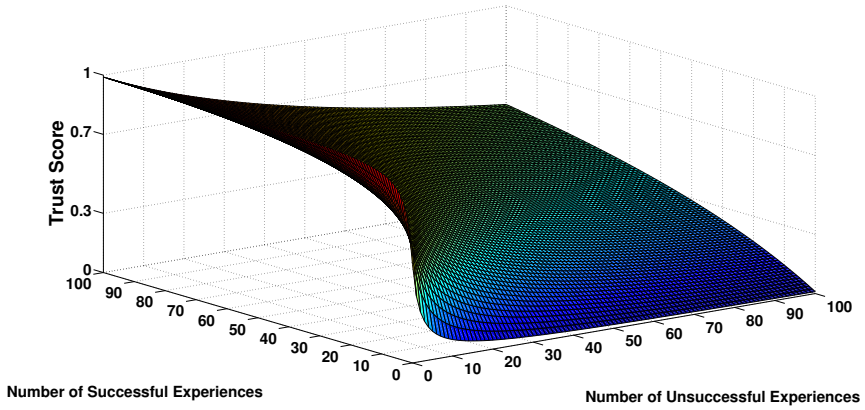
The trust score does not only represent the probability of having a successful experience. However, it is defined as the probability of having a future successful experience given the probability of successful events of past experiences. The Bayesian method is the best approach to determine the posterior probability. Thus, the Bayesian method is used to compute the trust score of success/fail type [31, 25]:

$$T_{x,y}^{Metric} = \frac{N_{x,y}^{+} + 1}{(N_{x,y}^{+} + 1) + (N_{x,y}^{-} + 1)} \tag{3.10}$$

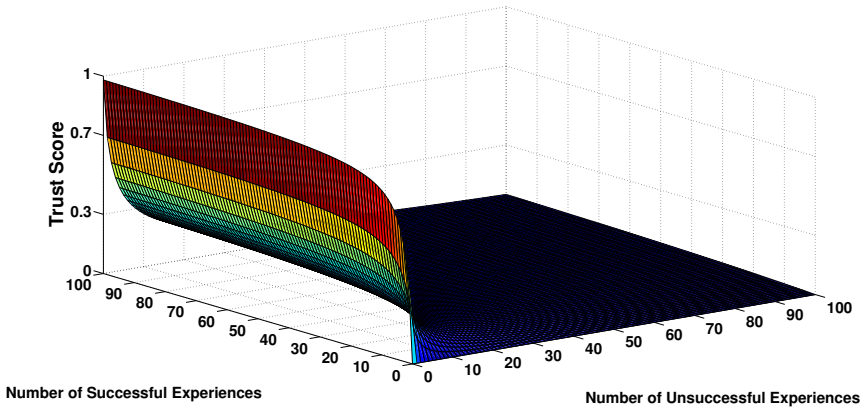
In TSW scheme, we use an improved Bayesian method (section 3.3.2). We consider a rewarding factor and a penalty factor to make our model more sensitive to the bad behavior in the uncertain area of the trust score. The rewarding factor approaches *one* gradually as the number of successful hits increases, while the penalty factor approaches *zero* gradually as the number of unsuccessful hits increases. Hence, we consider the density of communication experiences and the density of successful and unsuccessful experiences.

Definition 1. In TSW scheme, node x considers node y malicious; i.e., $T_{x,y} \leq th_L$ only if $N_{x,y} > 0$ and $N_{x,y}^{-} > N_{x,y}^{+}$, where $N_{x,y}^{-}$ and $N_{x,y}^{+}$ are positive integers.

Lemma 1. *TSW is able to detect malicious behaviour at node level if the number of successful experiences ($N_{x,y}^{+}$) is less than the number of unsuccessful experiences ($N_{x,y}^{-}$). Also, $T_{x,y}$ is sensitive to the bad behaviour of y.*



(a) Bayesian model.



(b) The proposed model.

Figure 3.5: Success/fail trust score behavior when using three different models with the variation of successful and unsuccessful experiences from 0 to 100.

Proof: If node y is considered bad, then the computed trust score by node x is $T_{x,y} < th_L$.

$$T_{x,y} = \frac{N^+_{x,y}+1}{N^{total}_{x,y}+2}\frac{N^+_{x,y}}{N^+_{x,y}+1}\frac{1}{\sqrt{N^-_{x,y}+1}} \leq th_L \tag{3.11}$$

Case 1: When the penalty factor (PF) $> th_L$, where $PF = \frac{1}{\sqrt{N^-_{x,y}+1}}$, $N^-_{x,y} < \frac{1}{th_L^2}$. So, whatever $N^+_{x,y}$ is, if $N^-_{x,y} > \frac{1}{th_L^2}$, node y is considered a malicious node.

Case 2: When PF $< th_L$, $N^-_{x,y} < \frac{1}{th_L^2} - 1$. To prove this, we need to prove that $\frac{N^+_{x,y}+1}{N^{total}_{x,y}+2}\frac{N^+_{x,y}}{N^+_{x,y}+1} < th_L$ when $N^-_{x,y} < N^+_{x,y}$. Then, $N^+_{x,y} < th_L N^+_{x,y} + th_L N^-_{x,y} + 2th_L$. Finally, $N^+_{x,y} < (\frac{th_L}{1-th_L})N^-_{x,y} + \frac{2th_L}{1-th_L}$. As $th_L > 0$, $N^+_{x,y} < N^-_{x,y}$. ■

Lemma 2. When there are no successful experiences, $T_{x,y} = 0$.

Proof: When $N^+_{x,y} = 0$, the rewarding factor is equal to zero ($\frac{N^+_{x,y}}{N^+_{x,y}+1} = 0$), which leads to $T_{x,y} = 0$. ■

Lemma 3. When there are no unsuccessful experiences, then the node must not be considered a malicious node as $N^+_{x,y} \geq 1$. Also, when $N^+_{x,y} \to \infty$, $T_{x,y} \to 1$.

Proof: When $N^-_{x,y} = 0$, then the penalty factor is equal to 1. This leads to $T_{x,y} = \frac{N^+_{x,y}}{N^+_{x,y}+2} > th_L$, then, $N^+_{x,y} > \frac{2th_L}{1-th_L}$ where $th_L \in [0,1]$ and $N^+_{x,y} \geq 1$. ■

Fig.s 3.5a and 3.5b show the effect of successful and unsuccessful interactions on the trust score. We vary the number of successful and unsuccessful interactions from 0 to 100 to cover all possible values. Fig. 3.5a is obtained from the Bayesian model in equation 3.10. It shows that the Bayesian model behaves similarly to a ratio-based model ($\frac{N^+}{N_{total}}$) when N^{total} is large. However, when N^{total} is small, the Bayesian model tends to not rely very much on the few successful experience counts. Fig. 3.5b shows the behavior of the proposed model. With the rewarding and penalty factors, the trust score is more sensitive to the unsuccessful experience and drops to low scores exponentially.

3.4.2 Analysis of Threshold-limit Model

Our threshold-limit model in equation 3.2 is used to detect flooding behavior. A threshold-limit model is presented in [31] as follows:

$$T_{x,y}^{Metric} = \frac{1}{\left\lceil \frac{N_y}{\eta} \right\rceil} \tag{3.12}$$

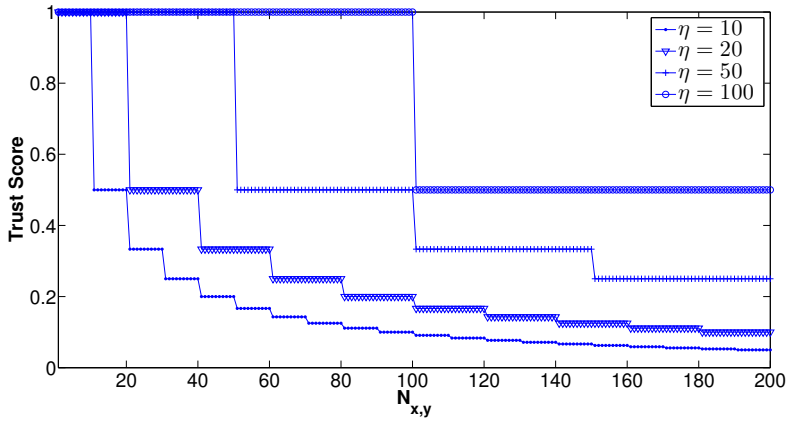
However, there are some issues/limitations in using this model to compute the trust score. As shown in Fig. 3.6a, the model in equation 3.12 sets the trust score to be 1 when N_y is less than the threshold η, even when N_y approaches the η value. Furthermore, the trust score jumps to (1/2) value after the recorded parameter count N_y goes beyond the threshold η. However, N_y does not degrade for some values of $N_y < 2 * \eta$ and the trust score remains the same when N_y is between $[\eta, 2 * \eta]$. This is because the trust score changes with $N_y = k * \eta$, where $k = 1, 2, 3, \ldots$. Finally, we can observe that the trust score does not reach the *zero* value even if $N_y >> \eta$. In the proposed model, we consider the mentioned issues of equation 3.12 when computing this type of trust metrics. Also, we take into consideration the fact that the sensor nodes are resource-limited devices. Thus, the proposed approach must be simple and have lightweight computational operations. Fig. 3.6b shows the outcomes of the proposed model (section 3.3.2) with the same threshold values used in Fig. 3.6a. The proposed model is sensitive when the number of interactions approaches the threshold value. Moreover, the proposed model becomes more sensitive when the number of interactions surpasses the threshold value by rapidly approaching the *zero* trust score.

3.4.3 Analysis of Trust Updating Mechanism

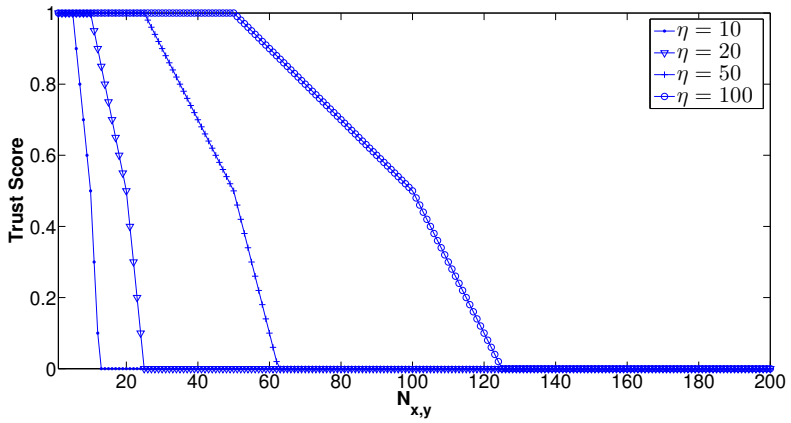
The trust updating mechanism in [97] is widely used [24, 27, 98] in the trust updating process as follows:

$$T_{x,y}^{Metric}(t) = (1 - \alpha) T_{x,y}^{Metric}(t) + \alpha T_{x,y}^{Metric}(t - \Delta t) \tag{3.13}$$

In the trust updating process, there are two cases, one is falling when $T(t) \leq T(t - \Delta t)$, and the other one is rising when $T(t) > T(t - \Delta t)$. Equation 3.13 provides similar behavior for rising and falling cases. However, we need to have a slow rising of the trust score to ensure that the pre-misbehaving nodes are no longer malicious and the system can trust

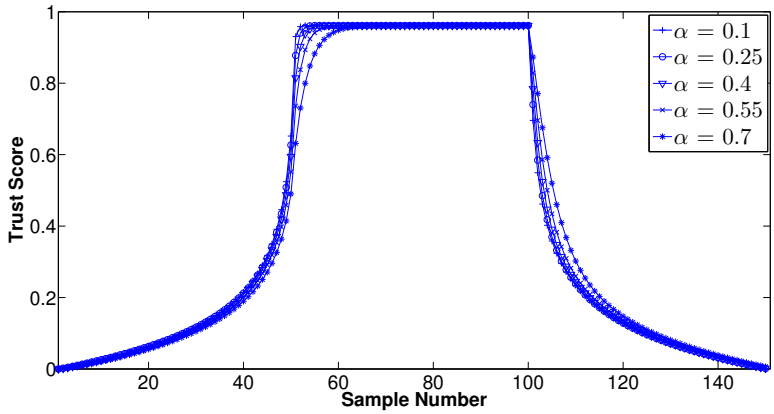


(a) ETMRM threshold-limit model.

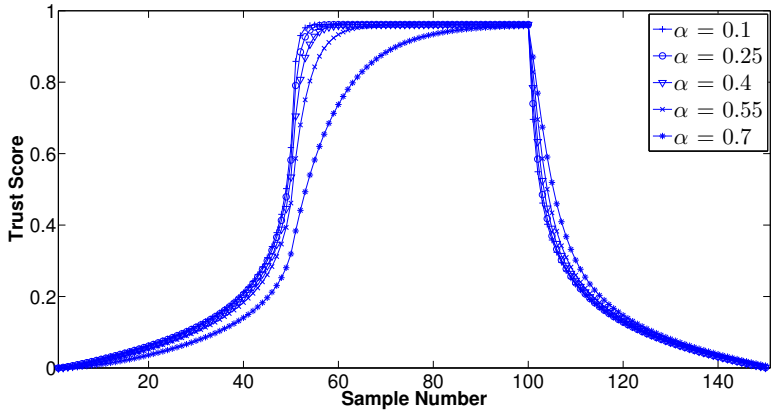


(b) Proposed threshold-limit model.

Figure 3.6: The influence of η in the threshold-limit models.



(a) without β factor.



(b) with β factor ($\beta = 0.2$).

Figure 3.7: The influence of α factor in the trust updating mechanisms for success/fail model.

them again. Our proposed mechanism in equation 3.3 behaves the same as equation 3.13 when $\beta = 0$. However, the trust score rises at a slower rate when $\beta > 0$.

Fig.s 3.7a and 3.7a show the trust score behavior from the trust updating in equation 3.13 and 3.3, respectively. In these figures, the total number of counts N^{total} is 50. The success counter N^{+} goes from *zero* (sample number 1) to N^{total} (sample number 51), and it remains with *zero* unsuccessful count until (sample number 100). Then, the success count goes to *zero* again at sample number 150. Similarly, N^{-} is equal to (N^{total} - N^{+}). In Fig. 3.7b, the trust score rises at a slower rate compared to Fig. 3.7a due to the effect of β factor. That is, it takes longer for a node to be trusted after behaving maliciously. In this figure, the value of β is 0.2.

Definition 2. $|\Delta t|_R$ is the required number of time windows (Δt) of regularly acting by the malicious node to raise its trust score T_y to the trusted zone. The number of Δt, in which malicious behavior is presented, is $|\Delta t|_M$.

Definition 3. The malicious node regularly acts for $|\Delta t|_R > |\Delta t|_M$ to restore its reputation and deceive the scheme.

Lemma 4. TSW scheme is resilient against the deception of the malicious node. Also, it ensures a slower rising and faster falling of the trust score for each time window based on past behavior.

Proof: As we mentioned, there are falling and rising cases.

Case 1: Assume that there is a continuous falling, i.e., $T_i > T_j$, where $i < j$, $\forall i, j \in [1, n]$.

$$
\begin{aligned}
T &= (1-\alpha)T_n + \alpha T_{n-1} \\
&= (1-\alpha)T_n + \alpha[(1-\alpha)T_{n-1} + \alpha T_{n-2}] \\
&= (1-\alpha)T_n + \alpha[(1-\alpha)T_{n-1} + \alpha[(1-\alpha)T_{n-2} + \alpha T_{n-3}]] \\
&= (1-\alpha)T_n + \alpha(1-\alpha)T_{n-1} + \alpha^2(1-\alpha)T_{n-2} + \\
&\ldots + \alpha^n(1-\alpha)T_0 \\
&= (1-\alpha)\sum_{k=0}^{n} \alpha^k T_{n-k} \\
&= (1-\alpha)T_n + (1-\alpha)\sum_{k=1}^{n} \alpha^k T_{n-k}
\end{aligned}
$$

The first part is the most recent evaluated trust, while the second part determines the effect of old trust scores. As $k \to n$, T_k becomes less effective on the new trust score. Thus, if $n \to \infty$, T_k will be neglected when $k \to 0$.

Case 2: Assume that there is a continuous rising, i.e., $T_i < T_j$, where $i < j$, $\forall i, j \in [1, n]$, and $\alpha + \beta < 1$, ($\gamma = \alpha + \beta$ where $\gamma < 1$),

$$\begin{aligned} T &= (1-\gamma)T_n + \gamma T_{n-1} \\ &= (1-\gamma)\sum_{k=0}^{n} \gamma^k T_{n-k} \\ &= (1-\gamma)T_n + (1-\gamma)\sum_{k=1}^{n} \gamma^k T_{n-k} \end{aligned}$$

■

Comparing cases 1 and 2, the model ensures that older trust scores have less weight in the falling case than the rising case as $\gamma > \alpha$, while the newer and larger trust scores have less weight when rising.

Using decay factors, the proposed updating mechanism effectively defends against irregular behavior. The malicious node needs to act non-maliciously for a long period of time to avoid detection. In other words, to raise its trust score, the node must regularly act for a considerable number of time windows (Δt). β factor is only used when the recorded trust level of a node is at the low level and the current level is higher, which indicates that this node might be trying to launch an ON-OFF attack. To summarize, a higher α value means that the new computed trust score relies less on the trust evaluation of the current window than the old trust score that is computed in the previous Δt. A higher β value means that the new computed trust score relies less on the current trust evaluation than the old trust score, which is computed in the previous Δt when the current computed score is higher than the old one. The values of α and β decay factors depend on the environmental and operational conditions of each trust metric [97].

3.4.4 Analysis of Trust Aggregation Model

In TSW scheme, the trust scores are aggregated at CH and controller levels. Moreover, two reliability scores are computed, which are score and node reliability, as discussed in section 3.3.3.

Definition 4. Reliable evaluator gives $T > th_H$ if a node is considered a good node and $T < th_L$ for bad nodes.

Definition 5. Unreliable evaluator gives $T < th_H$ for good nodes and $T > th_L$ for bad nodes.

Lemma 5. TSW scheme is robust against up to 54% *of unreliable evaluators.*

Proof: Assume that K is the total number of evaluators, I is the number of reliable evaluators, and J is the number of unreliable evaluators, where $K = I + J$. Then, equation 3.9 can be written as:

$$AT_y = \frac{\sum_{k=1}^{K} T_{k,y} R_{k,y}}{\sum_{k=1}^{K} R_{k,y}} = \frac{\sum_{i=1}^{I} T_{i,y} R_{i,y} + \sum_{j=1}^{J} T_{j,y} R_{j,y}}{\sum_{i=1}^{I} R_{i,y} + \sum_{j=1}^{J} R_{j,y}} \tag{3.14}$$

Case 1: Assume that a group of unreliable evaluators (J) tries to deceive the system about a non-malicious node y by providing zero trust, i.e., $T_{i,y} = 1$, $\forall i \in I$, and $T_{j,y} = 0$, $\forall j \in J$. Then,

$$AT_y = \frac{\sum_{i=1}^{I} R_{i,y}}{\sum_{i=1}^{I} R_{i,y} + \sum_{j=1}^{J} R_{j,y}} \tag{3.15}$$

when $I = J = K/2$, $AT_{avg,y} = 0.5$. Using equation 3.5, $R_{k,y} = 0.5$, $\forall k \in K$. Thus, from equation 3.14, $AT_y = 0.5$, where $R_{i,y}$ and $R_{j,y}$ are equal to 0.5. This is also applied when a group of unreliable evaluators (J) tries to deceive the system about a malicious node y by providing $T_{j,y} = 1$, $\forall j \in J$, while $T_{i,y} = 0$, $\forall i \in I$.

Case 2: In this case, we consider the worse case of case 1, in which the reliable evaluators give the non-malicious node (y) $T_{i,y} = th_H$, $\forall i \in I$, while the group of unreliable evaluators tries their best to deceive the system by giving a zero trust score for node y, i.e., $T_{j,y} = 0$, $\forall j \in J$.

Assume that $th_H = 0.7$ and $th_L = 0.3$. We have,

$$AT_{avg,y} = \frac{\sum_{i \in I} T_{i,y} + \sum_{j \in J} T_{j,y}}{I + J} = \frac{7I}{10N} \tag{3.16}$$

Thus, $R_{i,y} = \frac{3}{10} - \frac{7I}{10N}$, $\forall i \in I$, and $R_{j,y} = 1 - \frac{7I}{10N}$, $\forall j \in J$.

$$AT_y = \frac{\sum_{i=1}^{I} 0.7 R_{i,y}}{\sum_{i=1}^{I} R_{i,y} + \sum_{j=1}^{J} R_{j,y}} < 0.3 \tag{3.17}$$

Solving this inequality will lead to $I < 0.45335K$ and $J > 0.54664K$. ■

3.5 Performance Evaluation

In this section, we first discuss the simulation setup. Then, we numerically analyze the performance of the TSW scheme in terms of the trust score and the detection rate against several attack scenarios, namely black-hole, selective forwarding, DoS, good mouthing, and ON-OFF attacks. Finally, we provide an analysis of communication and storage overhead.

3.5.1 Simulation Setup

We have conducted simulations with Matlab to evaluate the performance of the TSW scheme. We consider a network with 40 randomly deployed sensors over an area of 400 x 400 with one sink node as an SDN controller. We further divide the network into equal-sized fixed clusters. The cluster size is set to be ten nodes, resulting in a total of four clusters. All nodes have a communication range of 150, while the sink node range is 300. The location of the sink is at the extremity of the network (100,100). The network area with the clusters and sensor placement used in the simulations is shown in Fig. 3.8. Moreover, we consider two types of nodes: normal nodes, which have good cooperation in forwarding messages and providing trust scores for others, and malicious nodes that perform one of the malicious behaviors defined in section 3.1.3. The simulation proceeds in rounds, where various aspects related to sensor communication and trust evaluation are updated, and each round is equal to a time window (Δt). The simulation ends if it reaches round 1000. The default parameter values for the trust scheme are defined as follows: $th_H = 0.7$, $th_L = 0.3$, w_m is equally distributed for all $m \in Metrics$, $\alpha = 0.5$, and $\beta = 0.2$.

3.5.2 Performance Analysis

First, we study the detection performance of TSW whereby a malicious node initiates several attack scenarios. Fig. 3.9 demonstrates the trust score as a function of the simulation rounds. In each scenario, the malicious node launches an attack at the $100th$ round, and terminates its malicious behavior at the $200th$ round.

Black-Hole (BH) Attack

Fig. 3.9a analyzes the detection performance against the malicious node performing a blackhole attack. The figure presents the trust score computed the SDN controller, the cluster head of the cluster containing the malicious node, and another cluster member

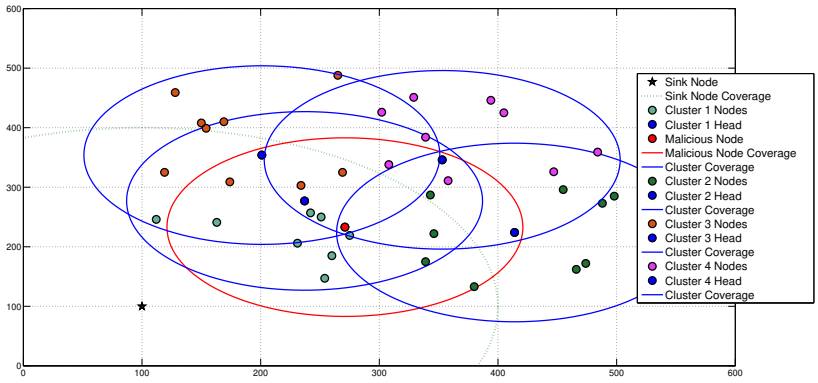


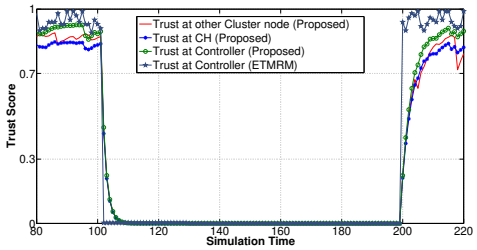
Figure 3.8: An example of simulation model.

that is a neighbour of the malicious node. Once the attack is initiated, the trust score of the malicious node drops significantly at all software-defined WSN levels and reaches zero. Note that if the malicious node stops dropping messages, its trust score does not immediately jump from the untrusted to the trusted zone, where the node requires more rounds to gain trust thanks to the proposed updating mechanism.

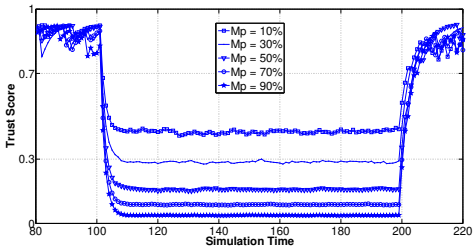
Selective Forwarding (SF) Attack

Here, the malicious node randomly drops received messages by a specific drop percentage (M_p). This scenario repeats with different values of M_p. Fig. 3.9b presents the aggregated trust scores computed for the malicious node by the CH. When M_p equals 50%, 70%, and 90%, TSW scheme is able to classify the malicious node as an untrusted node. However, when M_p is 10%, the CH is uncertain about determining the trustworthiness of the malicious node. When M_p is 30%, the trust score of the malicious node is very close to the uncertain zone.

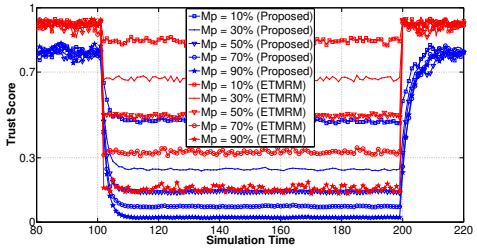
Fig. 3.9c presents the global trust scores computed by the SDN controller. When M_p is larger than 30%, the controller deems the malicious node as an untrusted. Only when M_p is 10% or less that the controller is uncertain about determining the trustworthiness of the malicious node. As a benchmark, we also compare with the detection performance of ETMRM. TSW detects the malicious node when M_p is larger than 30%, whereas ETMRM detects the malicious node when M_p is larger than 90%. Due to the incorporation of the rewarding and penalty factors when computing the forwarding trust metrics, TSW outperforms ETMRM.



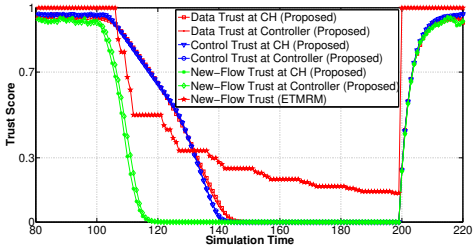
(a) BH attack detection.



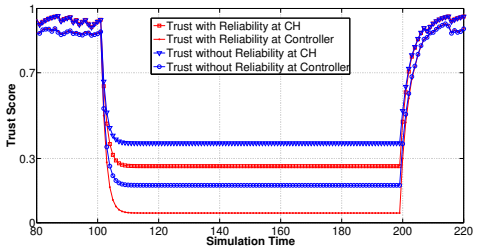
(b) SF Attack detection at CH level.



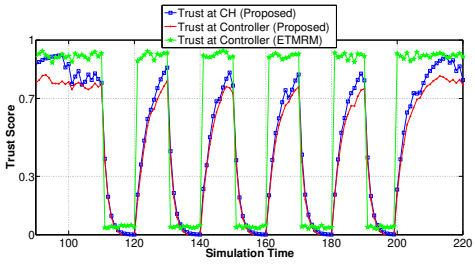
(c) SF attack detection at controller level.



(d) DoS attack detection.



(e) BH attack detection with GM attack.



(f) Defending against ON-OFF attack.

Figure 3.9: Detection performance for black-hole, selective forwarding, DoS, good mouthing, and ON-OFF attacks.

DoS Attack

Here, to launch DoS attack on several fronts, the sending rate of the malicious node increases gradually for data, control, and packet-in messages. Therefore, we need to observe the data trust metric, the control trust metric, and the new-flow trust metric. Fig. 3.9d shows that all trust metrics starts dropping once the malicious node initiates the attack. Although the malicious node behavior returns to normal at round 200, the trustworthiness of the malicious node does not enter the trusted zone instantly due to the use of the trust updating mechanism. In contrast, in ETMRM, the trust score jumps to the trusted zone immediately after the attack stops as it does not consider past scores in the evaluation updating mechanism. In addition, ETMRM scheme does not provide detection functionality at other levels besides the controller level. This prevents the scheme from responding quickly to malicious behaviors.

Good Mouthing (GM) Attack

We need to determine the effect of biased trust scores due to GM attack. In Fig. 3.9e, a malicious node launches a BH attack by dropping received messages. At the same time, other malicious nodes try to cover for this node by submitting a full trust score to the network. Malicious nodes make up for 30% of the total number of nodes. Fig. 3.9e shows the results of the attack detection when the bias score detection is applied and not applied, respectively. Fig. 3.9e shows the effectiveness of trust reliability in avoiding bias trust scores and computing the aggregated trust score from reliable nodes. Therefore, TSW can successfully detect the malicious node that drops messages even when 30% of nodes perform a GM attack.

ON-OFF Attack

We evaluate the TSW scheme against the ON-OFF attack scenario, whereby the malicious node performs a BH attack periodically. That is, the malicious node launches the attack for a certain period of time, and then it behaves regularly for another period of time. The period of being ON (behaving maliciously) or OFF (behaving normally) is called the ON-OFF period (ε). For example, if $\varepsilon = 5$, the malicious node launches a BH attack for five time rounds, and it behaves normally for the next five time rounds, and so on.

Fig. 3.10 shows the detection rate of a malicious node during the attack period with different ON-OFF periods ($\varepsilon = [1, 20]$). We modify the ETMRM to protect against ON-OFF

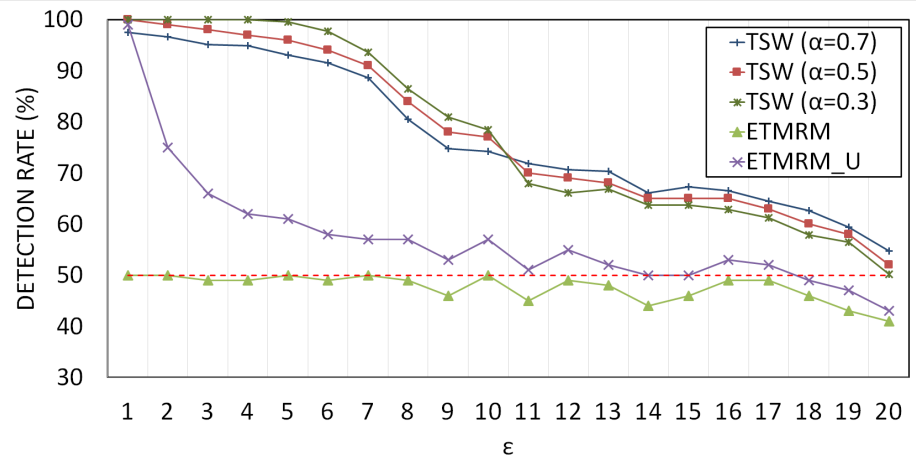


Figure 3.10: BH attack detection with ON-OFF attack.

attacks by adding an updating mechanism shown in equation (3.13) (named ETMRM_U). In Fig. 3.10, TSW scheme ($\alpha = 50\%$) detects the malicious node more than 70% of the attack period. However, the modified ETMRM method recognizes the malicious node as untrusted during the ON period only as it does not consider the historical trust level in the next time window. In addition, we observe that the detection rate for smaller ε is better when α is low (0.3), while a lower detection rate is observed for longer ε. As shown in Fig. 3.7, the trust convergence speed is lower when old scores are smaller (slow rising case), while the trust convergence speed is larger when old scores are larger (fast falling case). Therefore, as α increases, it allows the trust scheme to keep the malicious node in the untrusted zone.

In Fig. 3.9f, a malicious node launches a BH attack with $\varepsilon = 10$. Due to the use of the trust updating mechanism, the malicious node does not reach the trusted zone immediately. Longer time is needed to reach the trusted zone again compared to the ETMRM scheme. Note that this figure shows only the aggregated trust score computed based on the trust scores collected from the network about the malicious node. However, the controller can have a more restrictive policy for the ON-OFF attacker. For example, the controller can suspend the attacker for a sufficient time period to provide network services such as the forwarding process. Thus, the controller makes the final decision based on its policies on the previously recorded attackers.

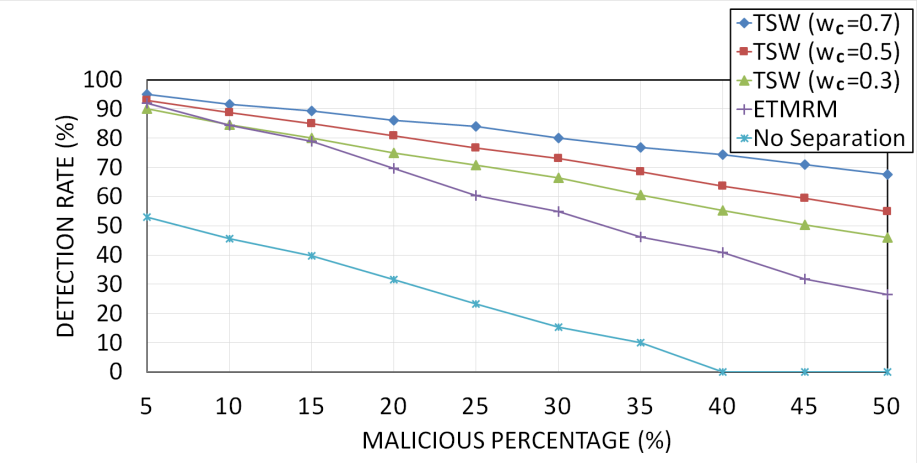


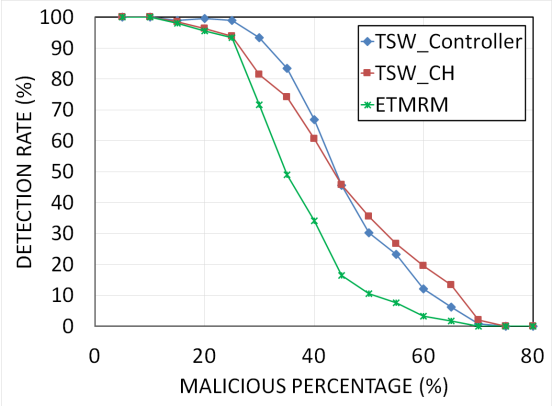
Figure 3.11: SF attack detection on the control plane.

Attacking the Control Plane

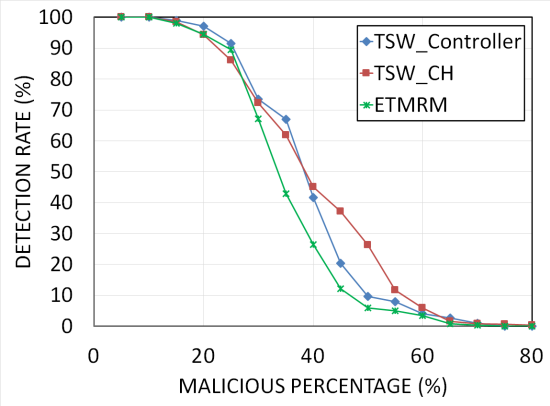
To demonstrate the importance of separating the control and data plane, we simulate a selective forwarding attack that deliberately drops control traffic. The number of control messages is less than that of the data traffic. Recall that w_c determines the weight of the trust metrics of the control traffic. Fig. 3.11 shows the detection rate against the percentage of malicious nodes in the network. Here, we vary w_c from 0.3 to 0.7 and show that our trust management scheme outperforms the no-separation baseline as well as ETMRM. This demonstrates the advantage of a decoupled treatment in our trust management scheme.

Collaborative Attack Detection

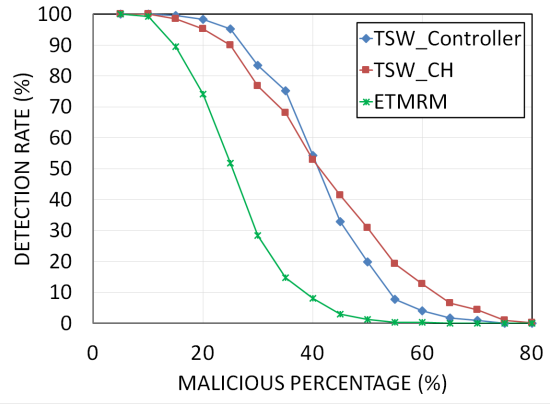
In this scenario, we evaluate the effect of biased aggregated trust scores on the detection rate of the BH, SF (Mp = 50%), and new-flow attacks. We design a scenario where a group of malicious nodes cooperates to perform one of these attacks while these nodes try to cover each other by performing GM attacks. Specifically, each malicious node sends full trust scores about other malicious nodes to a higher level to avoid their behavior detection. Fig. 3.12 shows the detection rate of malicious nodes with the increase in the number of malicious nodes in the attacking group (malicious percentage). We observe that the detection rate of BH and DoS attacks is mostly high (above 70%) when the malicious percentage is lower than 40%. A similar trend is shown for SF attack but with a lower detection rate. This is because only half of the messages are dropped. We also observe that



(a) BH attack detection.



(b) SF attack detection (Mp = 50%).



(c) New-flow attack detection.

Figure 3.12: Collaborative attack detection with GM attack.

with the increase of malicious nodes, the detection rate at the CH level becomes slightly higher than at the controller level. This is because of the increase in the number of collected biased scores at the controller than at the CH level. TSW shows better performance than ETMRM in all cases. ETMRM shows poor performance in detecting the new-flow attack when combined with a good mouthing attack.

3.5.3 Overhead Analysis

In TSW, we use topology discovery protocol for trust reporting [31]. Trust scores are attached to the report messages to minimize the communication overhead. First, we consider the total communication overhead of schemes considering the worse case (when every node wants to connect with every other node). TSW has a communication overhead of $\Pi + \pi(h-1)$, where Π is the total number of nodes, π is the number of clusters (or the number of aggregation points in ETMRM), and h is the average hop count to the controller. Due to the lack of an SDN controller, the communication overhead is larger in distributed protocols since each evaluating node needs to request trust recommendations from other nodes. We calculate the communication overhead for a clustered scheme in [28] to be $2\pi(\Pi^2 + \Pi) + 2\pi^2 + 2\pi$.

The TSW scheme uses lightweight computational models to calculate the trust scores that fit the limited-resource devices in terms of computational overhead. Moreover, the energy consumption of these non-complex models is negligible [49]. We further investigate the computational and energy overhead of the general trust evaluation process in chapter 5. Next, we analyze the memory requirement. Each node must store trust counters and scores for each of its neighbors. The counter is reset every Δt; thus, one byte counter can be sufficient. Assume that τ and p are the total bytes needed for trust scores and counters, respectively, the storage required for a sensor node in TSW is $(\tau + p)\phi$, where ϕ is the average number of neighbors for a node. For the CH node, additional storage of $\tau(\xi^2)$ is required for the cluster trust matrix where ξ is the average number of cluster members. Table 3.2 shows a comparison of the storage overhead.

Table 3.2: Storage and Communication Overhead

Overhead	Schemes	Value
Communication	TSW	$\Pi + \pi(h-1)$
	ETMRM	$\Pi + \pi(h-1)$
	non-SDN	$2\pi(\Pi^2 + \Pi) + 2\pi^2 + 2\pi$
Memory	TSW	$(\tau + p)\phi$, CH: $(\tau + p)\phi + \tau(\xi^2)$
	ETMRM	$(\tau + p)\phi$
	non-SDN	$(\tau + p)\phi$, CH: $(\tau + p)\phi + \tau(\xi^2 + \pi)$

Chapter 4

Secure and Energy-efficient Network Topology Obfuscation for Software-Defined WSNs

In Chapter 3, we have investigated a reactive defense solution. In this chapter, we propose a proactive SDN-based NTO solution to hide the network from the traffic analysis adversary. The proposed NTO is an energy-aware solution that is especially suited to the characteristics of WSNs and exhibits several advantages contrary to some existing solutions. The goal is to design obfuscation techniques to increase the cost for an adversary to discover the network topology. The proposed techniques aim to secure the network against network traffic attacks such as node-based attacks, link-based attacks, CrossPath attacks, and heuristic attacks.

4.1 Problem Formulation

In this section, first, the system and threat models are presented. Finally, the design goals of the proposed solution are discussed. A list of notations is given in Table 4.1.

4.1.1 System Model

The system model consists of several components, namely the SDN controller, sink nodes, and SDN-enabled sensor nodes. The network is modeled as a graph $G(V, E)$, where V is

Table 4.1: Table of Notations for Network Obfuscation Problem - Part I

Notations	Descriptions
$G(V, E)$	Network graph (WSN) G, where V is the set of vertices (nodes) and E is the set of edges (links).
$\|V\|$	Number of sensor nodes in the network.
e_{uv}	Direct connection (link) between nodes u and v.
CR_v	Communication range of node v.
d_{uv}	Distance between node u and v.
$Energy_v^0$	Initial residual energy of node v.
$E_{Tx}(l, d)$	The energy consumed to transmit l bytes for distance d.
$E_{Rx}(l)$	The energy consumed to receive l bytes.
$Energy_v$	The current (residual) energy level of node v.
f	Data flow that is defined by a source node and a destination node.
F_t	Set of data flows at time t ($F_t = \{f_1, f_2, ..., f_{\|F_t\|}\}$).
X^f	Boolean variable: $X^f = \{x^f_{v_1}, x^f_{v_2}, ..., x^f_{v_{\|V\|}}\}$ $x^f_{v_1}$ determines if node v_1 is assigned to data flow f.
F_t^c	Set of control flows at time t ($F_t^c = \{f_1^c, f_2^c, ..., f^c_{\|F_t\|}\}$).
X^{f^c}	Boolean variable: $X^{f^c} = \{x^{f^c}_{v_1}, x^{f^c}_{v_2}, ..., x^{f^c}_{v_{\|V\|}}\}$ $x^{f^c}_{v_1}$ determines if node v_1 is assigned to control flow f^c.
Δt	Control period (time window).

Table 4.2: Table of Notations for Network Obfuscation Problem - Part II

Notations	Descriptions
Route Obfuscation:	
C_v^{th}	Capacity limit (maximum number of flow rules (entries)) of the flow table for node v.
L^{th}	Route length limit in terms of the numbers of hops.
s_v	Similarity score of node v (equation 4.12).
h_v	History score of node v (equation 4.14).
k_r	Number of multiple mutated routes.
Sink Obfuscation:	
$\mathcal{S}$	Set of real sink nodes.
$\hat{\mathcal{S}}$	Set of nodes that are selected to be fake sink nodes.
k_s	Number of fake sink nodes in $\hat{\mathcal{S}}$.
th_E	Minimum energy level to be selected as a fake sink node.
$Fit_E(v)$	A score that determines how fit node v to be selected as a fake sink node (equation 4.19).
$v.H(k)$	Set of node v's neighbours that have k hops connection.
$\|v.H(k)\|$	Number of nodes that have k hops connection with node v.
$\Psi(\mathcal{S}, \hat{\mathcal{S}})$	Travel cost of the minimum spanning tree of real and fake sink nodes.
Ψ_{min}	Minimum travel cost threshold of minimum spanning tree of real and fake sink nodes.
th_d	Minimum distance threshold between the sink node and any fake sink node.
$v.cell()$	Cell's ID that node v belongs to.

the set of vertices (nodes) and E is the set of edges (links). SDN controller has the supervisory view of the network and is responsible for flow management. The controller receives the statistical updates from the underlying network components to build comprehensive network maps. Based on these maps, the controller can provide several services efficiently, such as routing, network management, and security. The sink node, which connects the network and the controller, has powerful resources. The network is deployed randomly, and network nodes are homogeneous, which means every sensor node has the same communication range (CR) and initial residual energy ($Energy^0$). Node i is connected to node j with edge e_{ij} only if it is within its communication range ($d_{ij} \leq CR_i$). In this research, a well-known transmission energy model is used [99]. The sensor node consumes $E_{Tx}(l, d)$ when it sends l bytes for distance d (equation 4.1). While, it consumes $E_{Rx}(l)$ when it receives l bytes, $E_{Rx}(l) = lE_{elec}$.

$$E_{Tx}(l,d) = \begin{cases} lE_{elec} + l\epsilon_{fs}d^2 & , d < d_0 \\ lE_{elec} + l\epsilon_{amp}d^4 & , d \geq d_0 \end{cases} \tag{4.1}$$

E_{elec} denotes the transmission circuit loss (50 nJ/bit). Depending on the distance between the sender and receiver nodes, the free space (d^2 power loss) or the multi-path fading (d^4 power loss) channel models are applied. The energy needed in both models for power amplification are ϵ_{fs} (10 pJ/bit/m^2) and ϵ_{amp} (0.0013 pJ/bit/m^4). After each transmission, the energy level is updated for the sender i and receiver j nodes ($Energy_i = Energy_i - E_{Tx}(l, d_{i,j})$ and $Energy_j = Energy_j - E_{Rx}(l)$, respectively).

Each node has a flow table. The SDN controller is responsible for updating the flow tables of each node in the network. If there is no particular rule for an incoming message, then a packet-in control message must be sent to the controller to obtain a new routing rule. When the controller receives the packet-in message, it responds with a packet-out message which contains the new flow entry [2]. To maintain network control and keep the flow table updated at all times, SDN provides several services [39]. Packet service manages packets exchanged between the control and data planes. Flow-rule service installs or updates rules in sensors via flow-mod messages. The topology service maintains the topology of sensors and links, discovers new sensors and tracks their locations, and establishes the control channel between sensors and controllers via several handshake messages. The liveness of sensors is periodically checked via echo request and echo reply messages. Therefore, when a sensor node in the network stops working due to energy exhaustion, the controller will be informed. Flow-metrics service is responsible for collecting flow statistics. It periodically queries the flows on network devices via stats requests and replies.

The controller defines F as the flow set of all computed flows in a certain control period (time window) Δt ($F = f_1, f_2, ..., f_i$). A Boolean variable is used to indicate whether or

not a node is selected as a part of flow f as follows:

$$x_v^f = \begin{cases} 1 & \text{, this node is selected as intermediate node in } f \\ 0 & \text{, otherwise.} \end{cases} \tag{4.2}$$

There are no dedicated links between nodes in wireless networks as the wireless node transmits the packets to the medium within its communication range. Thus, the flow is defined as follows:

$$X^f = \{x_1^f, x_2^f, ..., x_{|V|}^f | \forall f \in F\} \tag{4.3}$$

Moreover, the controller keeps the prior calculated flow sets from the previous control periods in a flow set matrix $F^T = [F_t, F_{t-\Delta t}, F_{t-2\Delta t}, ..., F_{t-T\Delta t}]$ where t is the current time. The controller does not determine the routes for the control flow. The routes are determined by the nodes themselves using a discovery approach [5]. However, the controller has knowledge of these paths. The control flow set for all nodes in the network is defined as F^c ($F^c = f_1^c, f_2^c, ..., f_{|V|}^c$). A similar Boolean variable $x_v^{f^c}$ is used to indicate whether or not a node v is selected as a part of control flow f^c. The control flow can be defined as follows:

$$X^{f^c} = \{x_1^{f^c}, x_2^{f^c}, ..., x_{|V|}^{f^c} | \forall f^c \in F^c\} \tag{4.4}$$

The node that is a part of a data flow X^f and a control flow X^{f^c} is considered as a shared node and vulnerable for the CrossPath attack (section 4.1.2).

4.1.2 Threat Model

In this research, we assume an outsider adversary, which is an unauthorized user who does not have permission to control the sensor network. The adversary wants to attack the network availability, however; s/he cannot attack the controller directly. To launch an effective attack, the adversary must learn the network topology and identify the high-profile nodes that play significant roles in network communication, including sink nodes, intermediate nodes, and shared nodes of both control and data traffic. Most messages are transmitted along paths that have high-profile nodes. This produces pronounced traffic patterns that reveal traffic path information, direction, and thus the location of these nodes. The adversary must first launch a traffic analysis attack, a remote software-based attack, or a physical attack on the network. This adversary can hijack (capture) sensor nodes; consequently, s/he is capable of obtaining its flow table, eavesdropping communications within the node's range (passive monitoring), and revealing some statistics about the neighborhood. The adversary can gather information about the network without being detected, as

the sensor node will continue to act normally with no malicious actions. The adversary can compromise only a small number of nodes at any reasonable cost (time). In this research, without loss of generality, we consider that the adversary can only compromise one node during a control period Δt.

In this research, an adversary can launch a sniffer attack, link-flooding attack, a Cross-Path attack, or a heuristic attack. Also, several attack scenarios and the damage level evaluation based on these attacks are provided in section 4.4.2. In **node-based attack**, the adversary captures and analyzes network communication packets such as sniffer attack. A sniffer adversary is capable of eavesdropping on the data traffic of nodes or links, monitoring network status, and stealing sensitive data. The overlapping points of data flows are convenient for the adversary during a sniffer attack. In **link-based attack**, the attacker is concerned in attacking certain links such as in the link-flooding attack. Link-flooding attack is a DoS attack in which the adversary targets a number of links by flooding packets. The adversary can disrupt a limited number of links or nodes without being detected for a specific time. The selection of the target set of links is based on the belief of what is significant within this set for certain flows. Therefore, the adversary uses data reconnaissance to gain knowledge about the high-profile data forwarding nodes. **CrossPath Attack** is a link-flooding DoS attack that targets the shared links between the control and data traffic in in-band control SDN [39]. A probing technique called Adversarial Path Reconnaissance (APR) is used to find the target links. The technique was inspired by the key observation that the delay of a control path is higher if a short-term burst of the data traffic passes through the shared links. Thus, an adversary can use a compromised node to identify the key data paths by generating data traffic and measuring the delay variations of the control paths. To identify a shared link using APR, the target data path must cross with a control path of a sensor belonging to the data path. After the discovery of these target links, the adversary will be able to launch a link-flooding attack. In this research, we assume an ideal case in which the adversary can identify all the possible shared paths using APR. In **heuristic attack**, the adversary goal is to maximize the attack on the high-profile nodes. To achieve the attack goal, the adversary uses a heuristic approach to move from one node to another [40]. A greedy heuristic expands the attack graph by selecting the most profitable node based on an evaluation function [100]. Using the evaluation function, the adversary can reveal the traffic pattern of neighbor nodes based on traffic volume and communication directions gathered through passive monitoring. This can be achieved either physically or by a software-based method. The adversary continues to move until s/he finds the high-profile targets. The adversary may face a deadlock, i.e., the candidate list of possible targets is empty. Then, the adversary chooses the next target randomly.

4.1.3 Design Goals

The proposed solution aims to achieve the following goals:

(1) **High obfuscation level**: The primary goal of the NTO solution is to minimize the damage level and maximize the cost to the adversary to launch efficient attacks.

(2) **Reliable routing**: The routing path of each flow must be provided with high reliability.

(3) **Energy efficiency**: The NTO solution needs to be energy efficient to eliminate the side effects of defense mechanisms and guarantee network performance.

Primarily, the objective function of the problem is formulated as maximizing the obfuscation level achieved by a certain solution Ω and minimizing the cost that the network must pay for this defense level ($\min Cost(\Omega)$ & $\max Obf(\Omega)$).

4.2 Route Obfuscation

The pattern of data and control traffic can be revealed as the network uses shortest path routing. Moreover, measuring the control delays may reveal the shared paths between data and control traffic [39]. Thus, to provide route obfuscation, a ranking-based route mutation mechanism is proposed. In the proposed solution, paths are ranked based on several criteria. Then, based on this ranking, the controller sets the mutated path for each flow under two key considerations. The first consideration is the total path cost which combines several key criteria to achieve reliable and energy-aware routing. The second consideration is the obfuscation level that is gained by the given paths. This determines the defence efficiency. Therefore, the route obfuscation problem is formulated as follows:

$$\min_{\forall f \in F_t} Cost(f) \;\&\; \max_{\forall f \in F_t} Obf(f) \tag{4.5}$$

When the controller sets the mutated path for a flow, it will update the flow rules for each node in the selected paths. Thus, all these rules will be deployed in the network when required. When the flow rule expires, the controller re-determines the mutated path for the active flows and updates the network. As shown in Fig. 4.1, in normal operations, the shortest path (f_1) is set for a flow from node s to node d. However, using route mutation, a different path (f_1') is set for the flow.

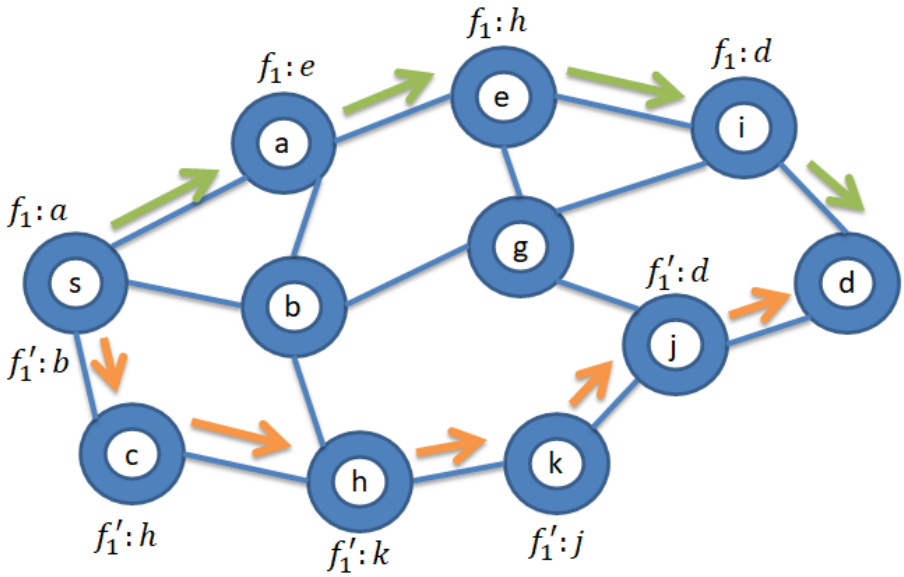


Figure 4.1: An example of shortest route for flow f compared to the muted route (f').

4.2.1 Path Cost Criteria

The cost of a path is the cost of all links in that path.

$$Cost(f) = \sum_{\forall e_{uv} \in f} Cost(e_{uv}) \tag{4.6}$$

Four key criteria are defined for determining link cost: node energy level, edge energy cost, node table flow capacity, and node reliability. The residual energy is a significant factor in selecting a particular node in the path. The energy level's weight of a node is calculated as follows:

$$\varepsilon_v = \frac{Energy_v}{Energy_v^0} \tag{4.7}$$

The energy consumption model of packet transmission is a function of distance. As a result, the distance d_{uv} is considered to determine the edge energy cost for edge e_{uv}. The higher distance results in the higher energy consumption (equation 4.1). Thus, the edge energy cost of e_{uv} is computed as follows:

$$e_{uv}^c = \frac{d_{uv}}{CR_u} \tag{4.8}$$

where the value is $0 < e^c_{uv} \leq 1$ for direct (valid) edges, and when there is no direct connection, it will be excluded by the algorithm. On the other hand, due to the limited capacity of the flow table, the capacity score C^c_v of a node is defined by dividing the number of flow rule entries at the previous control period over the capacity limit of the flow table for the node:

$$C^c_v = \frac{\sum_{\forall f \in F_{t-\Delta t}} x^f_v}{C^{th}_v} \tag{4.9}$$

We assume that C^{th}_v is equal for all the sensor nodes ($\forall v \in G.V$). The flow set $F_{t-\Delta t}$ is used because they are the current deployed flow rules in the network. Finally, the node reliability in packet transmission is considered to avoid nodes that have a history of higher failure rate due to the congestion, for example. A Bayesian method is used to compute the reliability score of a node in transmitting packets based on the total number of successful transmissions $N^{success}_{trans}$ and the total number of transmissions N^{all}_{trans} of this node [101]:

$$Rel_v = \frac{N^{success}_{trans} + 1}{N^{all}_{trans} + 2} \tag{4.10}$$

All of the above scores are normalized, and their values are represented between 0 and 1. Therefore, the cost of a link e_{uv} is calculated based on the above scores as follows:

$$Cost(e_{uv}) = \omega_\varepsilon(1 - \varepsilon_v) + \omega_e e^c_{uv} + \omega_C(1 - C^c_v) + \omega_r(1 - Rel_v) \tag{4.11}$$

where $\omega_\varepsilon + \omega_e + \omega_C + \omega_r = 1$. Assigning the weighting parameters depends on specific application to utilize the network performance.

4.2.2 Route Obfuscation Level

To determine the obfuscation level of the generated mutated paths of flows, two parameters are defined: Similarity s and History h. These parameters determine the overlapping criteria of route selection for a node. The Similarity s score is used to compute the overlapping between paths in the flow set F in the same control period (time window) Δt. Also, s score determines the number of shared intermediate nodes between the data flow F and the control flow F^c.

$$s_v = \sum_{\forall f \in F_t} x^f_v + \sum_{\forall f^c \in F^c_t} x^{f^c}_v \tag{4.12}$$

History h score is used to compare paths to flows in the previous control periods (F^T). h score is used to avoid selecting nodes that have already been selected several times.

The selection effect of previous control periods decays over time; i.e., the effect of flows in control period $t - a\Delta t$ is higher than that in $t - b\Delta t$ when $a < b$. Thus, the h score is defined as follows:

$$h_v = \sum_{\tau=1}^{T} 2^{-\tau} \Big(\sum_{\forall f \in F_{t-\tau\Delta t}} x_v^f + \sum_{\forall f^c \in F^c_{t-\tau\Delta t}} x_v^{f^c} \Big) \tag{4.13}$$

where $2^{-\tau}$ is the decay factor in which τ equals to one for the preceding control period and equals to T for the last stored control period. Using equation 4.12, equation 4.13 can be written as follows:

$$h_v = \sum_{\tau=1}^{T} 2^{-\tau} s_v^{t-\tau\Delta t} \tag{4.14}$$

Therefore, the obfuscation level of selecting a path is computed by combining the similarity and history scores of the intermediate nodes in that path. Hence, to maximize the obfuscation level in equation 4.5, a function of s and h is minimized, as follows:

$$\max_{\forall f \in F_t} Obf(f) = \min_{\forall f \in F_t} \sum_{\forall v \in G.V} x_v^f (\alpha s_v + \beta h_v) \tag{4.15}$$

Then, based on equations 4.6 and 4.15, the objective function of route mutation in equation 4.5 is written as:

$$\begin{aligned} \min_{\forall f \in F_f} & \sum_{\forall u,v \in G.V} x_u^f x_v^f Cost(e_{uv}) + \sum_{\forall v \in G.V} x_v^f (\alpha s_v + \beta h_v) \\ \text{s.t.} & \sum_{\forall f \in F_t} x_v^f \leq C_v^{th} \\ & \sum_{\forall v \in G.V} x_v^f \leq L^{th}, \forall f \in F \end{aligned} \tag{4.16}$$

The first constraint ensures that the selected node cannot be beyond the capacity limit, i.e., $C_u^{th} \geq \sum_{\forall f \in F_t} x_u^f$. The objective function is extended to consider the QoS constraints on routes. We assume that the QoS requirement of a mutated route is defined by the maximum allowed path length (L^{th}) of the route in terms of the numbers of hops (second constraint).

4.2.3 Multiple Mutated Routes

In this section, multiple paths are used for each flow to deceive the adversary. The controller assigns k_r paths with the objective function in equation 4.16. The generated k_r paths can

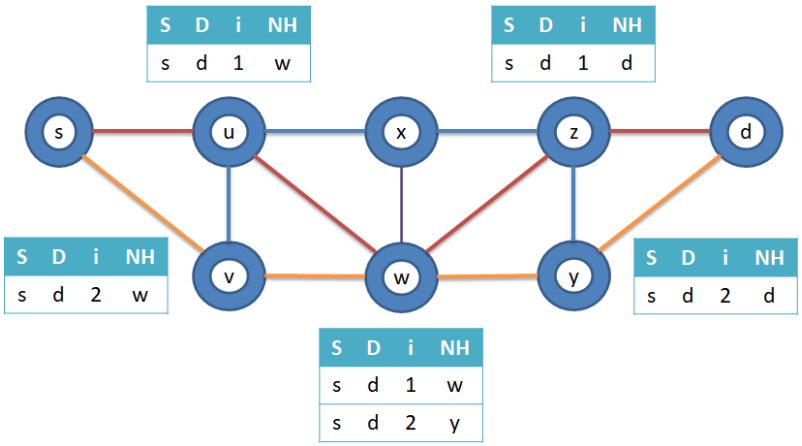


Figure 4.2: An example of multiple mutated routes.

be used in two ways. First, all the k_r paths must be used when sending a message. Thus, the source divides the message into k_r fragments, with each fragment is sent individually through one of the k_r paths. Second, the source will use a round-robin approach to select a path from the assigned k_r paths to send each message in the flow. As shown in Fig. 4.2, two paths ($k_r = 2$) are set for the flow. Moreover, the flow table is extended to have an extra field called path ID (i), ranging from 1 to k_r as it indicates the mutated path ID. This ID will be appended to the message header. Assume that there are three possible paths (orange, red, and green) for a flow (s→ d) in which node s is the source and node d is the destination (as shown in Fig. 4.3). Assume that all these paths are equally ranked, and all nodes have the same h scores. Only two paths ($k_r = 2$) are needed for the flow. Minimizing the number of shared nodes (edges) in the k_r mutated paths is needed. If the similarity between each pair of paths is computed, then $s(red, green) = 2$, $s(red, orange) = 1$, and $s(green, orange) = 1$. Both pairs (red and orange) and (green and orange) have the least similarity score. However, the pair (red and orange) is the best choice in terms of the shortest path. As another example, assume that one path is needed for the flow (s→ d) (Fig. 4.3). If history h score of node indicates how frequent this node was selected in a prior paths, then assume $h(w) = 2$, $h(x) = 1$, $h(y) = 1$, and $h(z) = 5$. This indicates that node z was selected most often for previous flows and/or in the previous time windows. Thus, it is better to avoid selecting this node as it has a higher h score than the other possible nodes. Both red and green paths pass through the node z. Therefore, selecting the orange path is the best choice based on the history of the nodes.

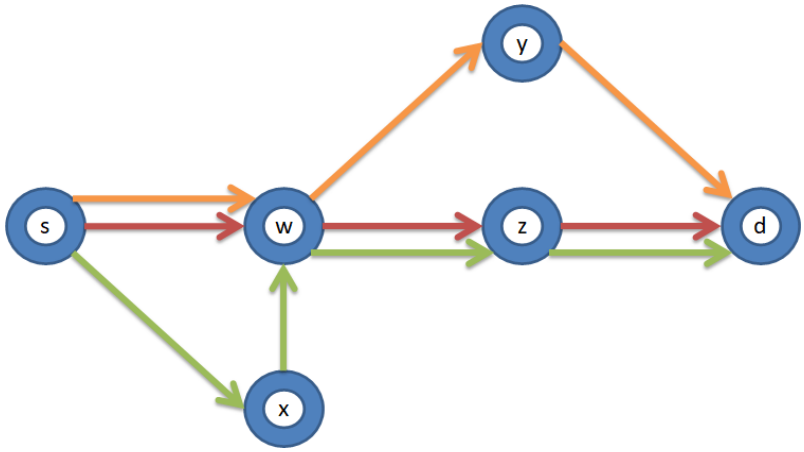


Figure 4.3: Example for paths Similarity and History.

4.2.4 Route Obfuscation Algorithm

Route obfuscation algorithm takes the initial and current energy level, reliability score, and history score of nodes as inputs. All of this information is obtained by the controller due to its supervisory view of the network. First, energy weight E_j, capacity score C_j^c, and reliability score Rel_j are computed for all nodes in the network (equations 4.7, 4.9, and 4.10). Also, the edge cost e_{ij}^c for all edges in the network is computed based on equation 4.8. Thus, the cost of each link in the network is precomputed for all links. In Algorithm 4, the detail of paths assignment of the route mutation is presented. A modified version of the Dijkstra algorithm is used to guarantee that the algorithm will select the optimal minimum cost route for each flow based on the path cost given in equation 4.6. Dijkstra algorithm is a greedy heuristic algorithm that at every step the fittest option possible is chosen at that step without consideration to future consequences. The algorithm excludes nodes to be the next hop when their flow table limit is exceeded ($available\_adjacent()$). The edge is valid as a possible candidate only if the combination of h_v and s_v is minimum. This step is defined as $\alpha s_v + \beta h_v < \Phi_{max}$ where Φ_{max} is initially very low then increases when the number of generated mutated paths are less than k_r, or the path length exceed L^{th}. After every failure, Φ_{max} is increased by ϕ^+. Finally, s_j and h_j scores are updated for all nodes in the selected paths. The time complexity of Algorithm 4 is $O(\varphi k_r|E|log|V|)$, where $|E|log|V|$ is the running time of Dijkstra algorithm. For the multiple mutated routes, it is multiplied by k_r as the algorithm runs the Dijkstra algorithm k_r times. φ is defined as the average number of failures to generate a valid path. φ is inversely proportional with the step size ϕ^+ due to the size of the selection pool.

Algorithm 4: Route Obfuscation Algorithm.

```
Input  : G(V, E), Cost(Link), h, s
Output: f
while (f.paths(k_r)! = valid) do
    for k = 1 ⟶ k_r do
        Q.init(G.V) ;
        while not Q.isEmpty() do
            u ⟵ Q.extractMin() ;
            for each v ∈ u.available_adjacent() do
                if (αs_v + βh_v) < Φ_max then
                    if v.cost() > u.cost() + Cost(Link_{u,v} ) then
                        v.cost ⟵ u.cost() + Cost(Link_{u,v}) ;
                        v.parent ⟵ u ;
                    end
                    Q.modifyKey(v);
                end
            end
        end
    end
    Φ_max = Φ_max + φ^+;
end
for each v ∈ f.paths(k_r) do
    update(v, s_v, h_v) ;
end
```

SNcRM algorithm has a time complexity of $O(|V|^2 * 2^{|V|})$ [34] while SRO algorithm's time complexity is $O(k_r|E|log|V|)$ [37].

4.3 Sink Obfuscation

In fully centric WSNs, most of the data messages are delivered to the sink node to reach the application server. Moreover, in software-defined WSNs, the controller adds another level of centrality, as shown in Fig. 1.1. First, the sensed data is delivered to the application layer above the controller. Second, there is the network configuration exchange between the controller and the SDN-enabled sensors [102]. This produces a pronounced communication

pattern that exposes the sink identification. The unique function of the sink node to connect the network and the controller makes it a single point of failure. Thus, an adversary that attempts to attack the network availability can reveal the sink node by applying traffic analysis techniques. The goal of sink node obfuscation is to minimize the traceability of a sink node by an adversary and increase the search time due to the traffic pattern of the network. To hide the sink node, misleading the adversary and covering the exclusive traffic pattern are needed, which usually exposes the sink node. To achieve this, k_s fake sink nodes are employed in the network that have a similar deceptive traffic pattern as the sink nodes. This solution is inspired by the k-anonymity model [89]. As k_s fake sink nodes increases, sink node discovery becomes more difficult. However, the deceptive traffic will result in extreme and redundant overhead. As a result, the problem is formulated to determine how these fake sink nodes are selected while simultaneously maximizing the obfuscation level and minimizing the cost:

$$\max Obf(\hat{\mathcal{S}}) \ \& \ \min Cost(\hat{\mathcal{S}}) \tag{4.17}$$

where $\hat{\mathcal{S}}$ is the set of nodes that are selected to be fake sink nodes ($\hat{\mathcal{S}} = \{\hat{s}_1, \hat{s}_2, ..., \hat{s}_{k_s} \ | \forall \hat{s} \in G.V\}$). $Cost(\hat{\mathcal{S}})$ is the cost of selecting $\hat{\mathcal{S}}$ which determined by maintaining the energy constraints of the selected nodes and generating deceptive traffic. The goal is creating more local maxima where the fake sink nodes act as traps for the heuristic adversary. The deceptive traffic is created when a real message is generated. In contrast to existing techniques, all messages are delivered to the nearest fake sink node only and vice versa to minimize the overhead. Furthermore, the sink node forwards the broadcast message to the fake sink nodes to reduce the overhead. When the fake sink node receives the broadcast message, it will broadcast the message within its cell.

4.3.1 Selection of Fake Sink Nodes

First, the network is divided into k_s non-overlapping cells to minimize the traffic overhead. Then, one of the k_s fake sink nodes is selected from each cell. The deceptive traffic inside a cell is generated between the cell members and the chosen fake sink node to create local maxima. The formation of a cell can be achieved based on the number of nodes (node density), distances, and/or expected traffic. The controller sorts all nodes in the cell based on an energy-based fit score to select the fake sink node from a cell. Then, the $\hat{\mathcal{S}}$ set is formed of the top nodes in each cell. Initially, $\hat{\mathcal{S}}$ are chosen randomly under the security constraint as all the nodes have the same initial energy. However, after one round, the residual energy will be different. When the residual energy of the cell's fake sink node falls

below a threshold th_E (computed based on the energy level of all nodes in the cell), the controller reselects the cell's fake sink node and updates the flow rules. Thus, equation 4.17 is re-written such that the cost of selecting the fake sink for each cell is minimized while the residual energy of the selected node is greater than th_E, as follows:

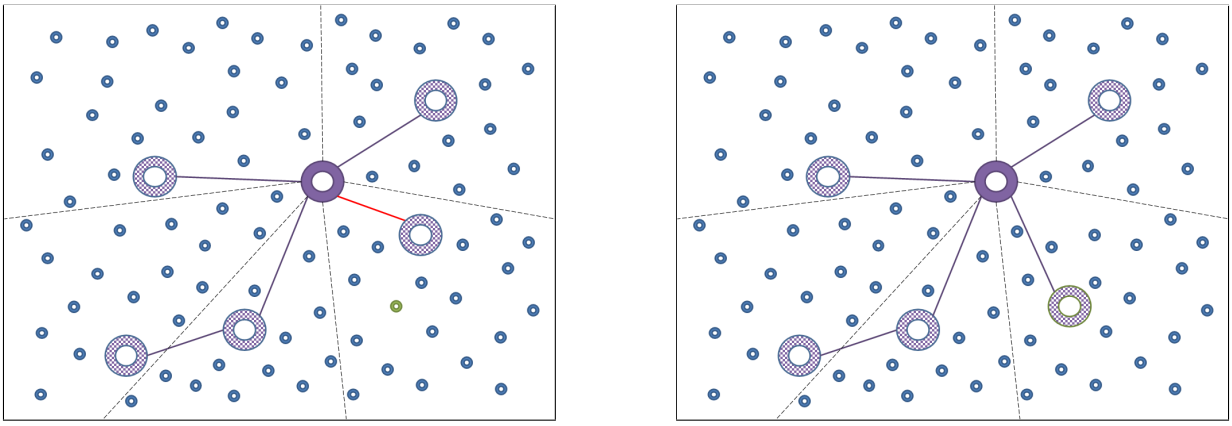
$$\begin{aligned} &\max \sum_{\forall \hat{s} \in \hat{\mathcal{S}}} Obf(\hat{s}) \;\&\; \min \sum_{\forall \hat{s} \in \hat{\mathcal{S}}} Cost(\hat{s}) \\ &\text{s.t.} \quad Energy_{\hat{s}} \geq th_E, \forall \hat{s} \in \hat{\mathcal{S}} \end{aligned} \tag{4.18}$$

When a node is selected in the $\hat{\mathcal{S}}$, both its own energy level and also the energy levels of its neighbors are crucial. All nodes in the cell deliver deceptive messages to this fake sink node. The candidate nodes are sorted based on a fit score (Fit_E) to minimize the effect of energy consumption due to the deceptive traffic inside the cell. The fit score considers the energy level of the node and the energy level of m-levels neighbor nodes in the cell. If the neighbor node has fewer hop connections to the candidate node, then there is a more significant effect on its fit score. The nodes closest to the fake sink node will consume more transmission energy due to deceptive traffic delivery. For example, if node v is a candidate node, node u is a one-hop neighbour to node v while node w connects to v in three hops. The energy level of these neighbor nodes u and w are considered when determining the fit score of node v. However, the influence of node u's energy level must be greater than that of node w due to the closer distance. Hence, 2^{-k} factor is associated with each neighbor nodes when computing Fit_E where k is the number of hops. Therefore, the fit score Fit_E of node v is calculated as follows:

$$Fit_E(v) = Energy_v + \sum_{k=1}^{m} 2^{-k} \left(\frac{\sum_{i=1}^{|v.H(k)|} Energy_{v.H(k)[i]}}{|v.H(k)|}\right) \tag{4.19}$$

where Fit_E is a score in the range of [0, 2]. v is any candidate node, and m is the highest number of hops for node v with the farthest node in the cell ($v.cell()$). $v.H(k)$ returns the set of neighbour nodes that have exactly k hops connection with node v where $v.H(k)[i]$ is the i-th node in the set and $|v.H(k)|$ is the number of nodes in that set. The average energy level of each node in the k-levels is computed. The information needed to determine Fit_E score is easily obtained by the controller due to its supervisory view. Thus, the cost function of selecting a fake sink node in equation 4.18 is determined by Fit_E as follows:

$$\begin{aligned} &\max \sum_{\forall \hat{s} \in \hat{\mathcal{S}}} Obf(\hat{s}) \;\&\; \sum_{\forall \hat{s} \in \hat{\mathcal{S}}} Fit_E(\hat{s}) \\ &\text{s.t.} \quad Energy_{\hat{s}} \geq th_E, \forall \hat{s} \in \hat{\mathcal{S}} \\ &\quad Fit_E(\hat{s}) \geq Fit_E(v), \forall \hat{s} \in \hat{\mathcal{S}} \;\&\; \forall v \in \hat{s}.cell() \end{aligned} \tag{4.20}$$



(a) Initial Set of Fake Sink Nodes ($\hat{\mathcal{S}}$). (b) Final Set of Fake Sink Nodes ($\hat{\mathcal{S}}$).

Figure 4.4: Initial and Final Sets of Fake Sink Nodes ($\hat{\mathcal{S}}$).

The second constraint ensures that the selected fake sink nodes have the highest fit score of their cells. Fig. 4.4 shows an example of $\hat{\mathcal{S}}$ selection procedure. First, the network is divided into five cells; thus, five fake sink nodes are selected ($k_s = 5$). Fig. 4.4a presents the initial set of $\hat{\mathcal{S}}$ where the selected nodes are the highest Fit_E nodes of their cells. Fig. 4.4b presents the final selected set of $\hat{\mathcal{S}}$. Fig. 4.5 shows the m-levels neighbour nodes of two nodes (Diamond $\diamond$ and Star $\star$) of the cell in the upper left corner of Fig. 4.4. For both cases, the red nodes are one-hop neighbors, the green nodes are two-hops neighbors, the purple nodes are three-hops neighbors, and so on. Assume that $\diamond.E$ is equal to $\star.E$ which is 0.8. For the diamond node, the average of energy level of the two one-hop neighbours is 0.6 and the average of five two-hops neighbours is 0.68. For the star node, the average of energy level of the four one-hop neighbours is 0.675 and the average of three two-hops neighbours is 0.666. If $m = 2$, then $Fit_E(\diamond) = 0.8 + 0.6 * 2^{-1} + 0.68 * 2^{-2} = 1.27$ and $Fit_E(\star) = 0.8 + 0.675 * 2^{-1} + 0.666 * 2^{-2} = 1.304$. Thus, node $\star$ is fitter than node $\diamond$ to be a fake sink node even they have equal energy level. The effect of the cell's nodes differs for each case as well as Fit_E.

The goal is to maximize the number of steps for the adversary to locate the sink nodes. Therefore, the higher steps indicate a higher level of obfuscation. To determine the steps that adversary must endure moving from one trap (local maxima) to the next nearest trap, the travel cost (Ψ) of a Minimum Spanning Tree (MST) of the real and fake sink nodes ($\mathcal{S}$ and $\hat{\mathcal{S}}$) is used. The travel cost $\Psi(\mathcal{S}, \hat{\mathcal{S}})$ is determined as the summation of distances

of MST:

$$\Psi(\mathcal{S}, \hat{\mathcal{S}}) = \sum_{\forall \bar{s}, \tilde{s} \in MST\{\mathcal{S}, \hat{\mathcal{S}}\}} d_{\bar{s},\tilde{s}} \tag{4.21}$$

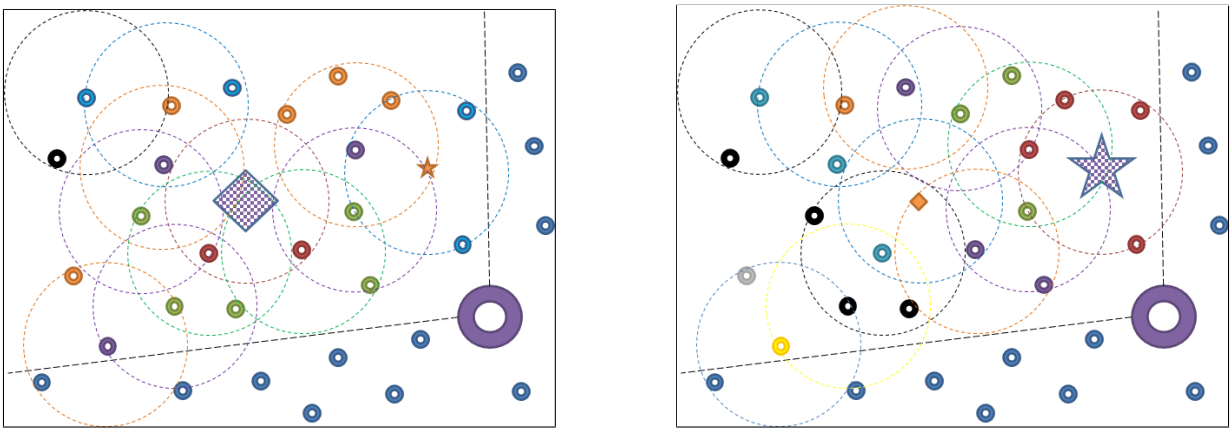
The MST is rooted at the real sink node, where each edge in the MST determines the next nearest maxima from one sink to another (real or fake). Thus, equation 4.20 is re-written by including the MST travel cost as follows:

$$\begin{aligned} \max \quad & \sum_{\forall \hat{s} \in \hat{\mathcal{S}}} Fit_E(\hat{s}) \\ \text{s.t.} \quad & Energy_{\hat{s}} \geq th_E, \forall \hat{s} \in \hat{\mathcal{S}} \\ & Fit_E(\hat{s}) \geq Fit_E(v), \forall \hat{s} \in \hat{\mathcal{S}} \ \& \ \forall v \in \hat{s}.cell() \\ & \Psi(\mathcal{S}, \hat{\mathcal{S}}) \geq \Psi_{min} \\ & d_{s,\hat{s}} \leq th_d, \ \forall \hat{s} \in \hat{\mathcal{S}} \ \& \ \forall s \in \mathcal{S} \end{aligned} \tag{4.22}$$

where Ψ_{min} is the minimum travel cost (third constraint). Furthermore, a minimum distance threshold th_d is defined such that the distance between the sink node and a fake sink node is greater than this threshold (fourth constraint). Fig. 4.4a shows the constructed MST of the initial set of $\hat{\mathcal{S}}$ with the real sink node. However, the initial set of $\hat{\mathcal{S}}$ does not meet the requirements of travel cost (Ψ_{min}). Therefore, the one with minimum distance is removed and replaced with the following top node from the same cell. Then, the MST of the final set of $\hat{\mathcal{S}}$ is constructed as shown in Fig. 4.4b.

4.3.2 Sink Obfuscation Algorithm

First, the network is divided into k_s cells. Then, the controller selects the top (Fit_E score) node in each cell as $\hat{\mathcal{S}}$ excluding nodes that have a distance to the real sink lesser than the minimum threshold th_d. If $\hat{\mathcal{S}}$ fails to meet the travel cost constraint, then the node in $\hat{\mathcal{S}}$ with minimum distance will be replaced with the following top candidate nodes in the same cell, and so on. In Algorithm 5, the overview of the selection of fake sink nodes algorithm is presented. To select the $\hat{\mathcal{S}}$, the controller sorts the candidate nodes which meet the minimum distance threshold th_d in each cell by its current energy level. Then, the $\hat{\mathcal{S}}$ are the top nodes in the k_s cells. Next, the controller constructs an MST from the $\hat{\mathcal{S}}$ set to determine the travel cost. After that, the travel cost of MST ($\Psi(\mathcal{S}, \hat{\mathcal{S}})$) is compared with the minimum travel cost (Ψ_{min}). If $\Psi(\mathcal{S}, \hat{\mathcal{S}})$ is larger than Ψ_{min}, then this initial set is accepted as the final set. If $\Psi(\mathcal{S}, \hat{\mathcal{S}})$ is smaller than Ψ_{min}, then the node in MST that has the closer distance to the real sink(s) is replaced with the next candidate, and



(a) Diamond node as a candidate node.

(b) Star node as a candidate node.

Figure 4.5: m-levels of connections when determining the Fit_E score of the star node.

the MST step is repeated. The time complexity of Algorithm 5 is $O(|V|\eta^2 log\eta)$ where η is the number of real and fake sink nodes. The first loop takes $O(|V|)$ as the algorithm sorts the cells' nodes based on Fit_E. Creating MST's time complexity is $O(\eta^2 log\eta)$. The last loop can be repeated $|V|$ times as a worse case. Thus, this part's time complexity is $O(|V|\eta^2 log\eta)$. PLAUDIT algorithm's time complexity is $O(|V||E|^2)$ [38].

4.4 Performance Evaluation

A simulation model is used to evaluate the effectiveness of the proposed proactive defense against traffic analysis attacks, and several evaluation parameters are measured. Also, several routing mechanisms are simulated and compared. First, to obtain an estimate of a lower bound on the routing cost, SP routing is simulated, which selects the shortest path for each flow. The second routing mechanism is a ranking-based SP (rSP) scheme in which routes are selected based on the link cost weight in equation 4.11. The third routing mechanism is Route Mutation (RM) scheme, in which routes are selected based on the shortest path while considering the similarity s and history h scores in section 4.2.2. The fourth routing mechanism is the ranking-based RM (rRM) scheme in which routes are selected based on the objective function in equation 4.16. The fifth routing mechanism is Random ϕ^+ ranking-based RM (RrRM) scheme in which routes are selected based on the objective function in equation 4.16, besides, the increment in Φ_{max} is a random variable with range $(0, \phi^+]$. Variations of multiple mutated routes for rRM and RrRM mechanisms

Algorithm 5: Sink Obfuscation Algorithm.

Input : $G(V, E)$, $\mathcal{S}$, $Energy$, $cells$
Output: $\hat{\mathcal{S}}$
for *each* $c \in cells \subset G$ **do**
  $Fit_E \longleftarrow$ computeFit_E(Energy); // Eq. 4.19
  if $(d_{v,s} > th_d)$ $\forall v \in c$, $\forall s \in \mathcal{S}$ **then**
    $c.candidate\_list.append(v)$;
  end
  sort $\forall v \in c.candidate\_list$ based on $Fit_E(v)$;
  $\hat{\mathcal{S}}.append$(c.candidate_list.$extractMax()$);
end
createMST($\mathcal{S}, \hat{\mathcal{S}}$);
while $(\Psi(\mathcal{S}, \hat{\mathcal{S}}) < \Psi_{min})$ **do**
  for *each* $\hat{s} \in \hat{\mathcal{S}}$ **do**
    $\hat{s}.sumDistance \longleftarrow \sum_{\forall s \in \mathcal{S}} d_{\hat{s},s}$;
  end
  $q \longleftarrow$ minNode(sumDistance);
  $\hat{\mathcal{S}}.remove(q)$;
  $\bar{c} = q.cell()$;
  $\hat{\mathcal{S}}.append(\bar{c}$.candidate_list.$extractMax())$;
  createMST($\mathcal{S}, \hat{\mathcal{S}}$);
end

are also simulated (krRM and kRrRM). The sixth routing mechanism is ksRM, in which k_s of fake sink nodes are used in the network using the objective function in equation 4.22. A combination of krRM and ksRM is also simulated (kskrRM). The last routing mechanism is the Random Walk (RW) scheme, in which the next hop is randomly selected by assigning probability for each eligible link. Since RW routing generates random traffic, it provides an upper bound of the routing cost and the defense level. Moreover, the proposed NTO solution is compared with state of the art solutions such as MRRF [87], SNcRM [34], SRO [37], and PLAUDIT [38].

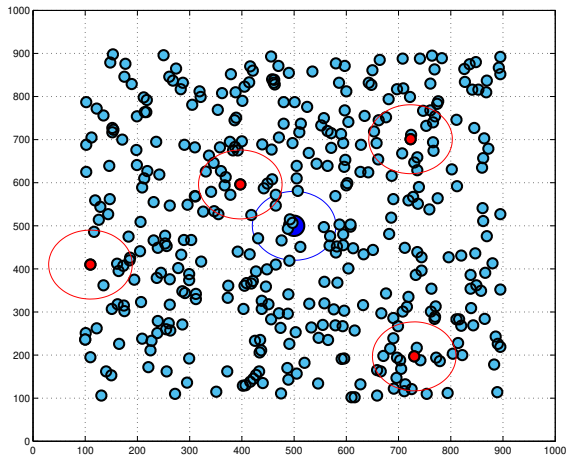


Figure 4.6: Sample of Simulation Model.

4.4.1 Simulation Setup

The simulation is conducted using MATLAB. A network with 400 sensors is considered that are randomly deployed over an area of 800 x 800 with one sink node. The SDN controller is connected with the sink node using a secure wired network. All nodes have a communication range of 80. The location of the sink node is at the center of the network (500, 500). The simulation proceeds in rounds, where various aspects related to flow routes are updated. The positions of nodes in a network would affect the experimental results. Thus, 1,000 experiments are conducted while the positions of sensor nodes are randomly changed in each experiment. Then, the average results of over 1,000 experiments for each topology are used. A sample of the network area with sensor nodes, sink node, and initial fake sink nodes' placement used in the simulations is shown in Fig. 4.6.

4.4.2 Attack Scenarios

In section 4.1.2, several attack types are defined. In this section, the attack scenarios used for evaluation are defined. The attack success rate defines the damage level of each attack scenario. In the first scenario, the adversary compromises nodes to sniff at the traffic that passes these nodes and around them. The success rate of the sniffer attack is determined by the degree of the node. The degree of a node is defined by the ratio of the number

of routes passing the node from all possible routes. A maximum likelihood estimation (MLE) method is used to compute the success rate. In the second scenario, the adversary launches a link-based attack on certain paths to perform link-flooding attack later. The attack success rate is determined by how significant these links are. The weight of a link e_{ij} is the product of the betweenness centrality of node i and j. The betweenness centrality of a node is defined as the average of the total probabilities that routes passing through this node over all possible routes [103]:

$$B_v = \frac{\sum_{\forall r \in routes} x_v^r}{0.5|V|(|V|-1)} \tag{4.23}$$

In the third scenario, the adversary uses a compromised node v to learn about the shared links between data and control paths using the ARP of the CrossPath attack. The attack success rate is determined by the possible discovered shared links over the total number of control links in the network. In the last scenario, the heuristic attack is considered in which the adversary uses the traffic volume for the evaluation function. An attack failure rate is determined by the number of deadlock points the adversary may face when searching for the sink node. Also, the h-steps is defined as the number of steps needed to identify the sink node.

4.4.3 Evaluation Parameters

Several evaluation parameters are used to evaluate the proposed solution:

Energy Consumption

Energy consumption determines the average energy consumption of nodes.

Lifetime

The network's lifetime is determined by the time of the first node dies. In general, a longer lifetime implies that the communication traffic is more balanced among the nodes.

Path Length

Path Length is determined by the average number of hops for the generated routes.

Entropy

Entropy determines the randomness of network traffic (i.e., the distribution of traffic volume).

$$Entropy = -\sum_{\forall v \in G.V} (\frac{N_v}{N_{total}} log_2 \frac{N_v}{N_{total}}) \tag{4.24}$$

In general, a higher value of entropy implies that the communication traffic pattern is more random.

Success Rate

Attack success rate is defined for each attack scenario in section 4.4.2. A lower success rate implies a higher defense level.

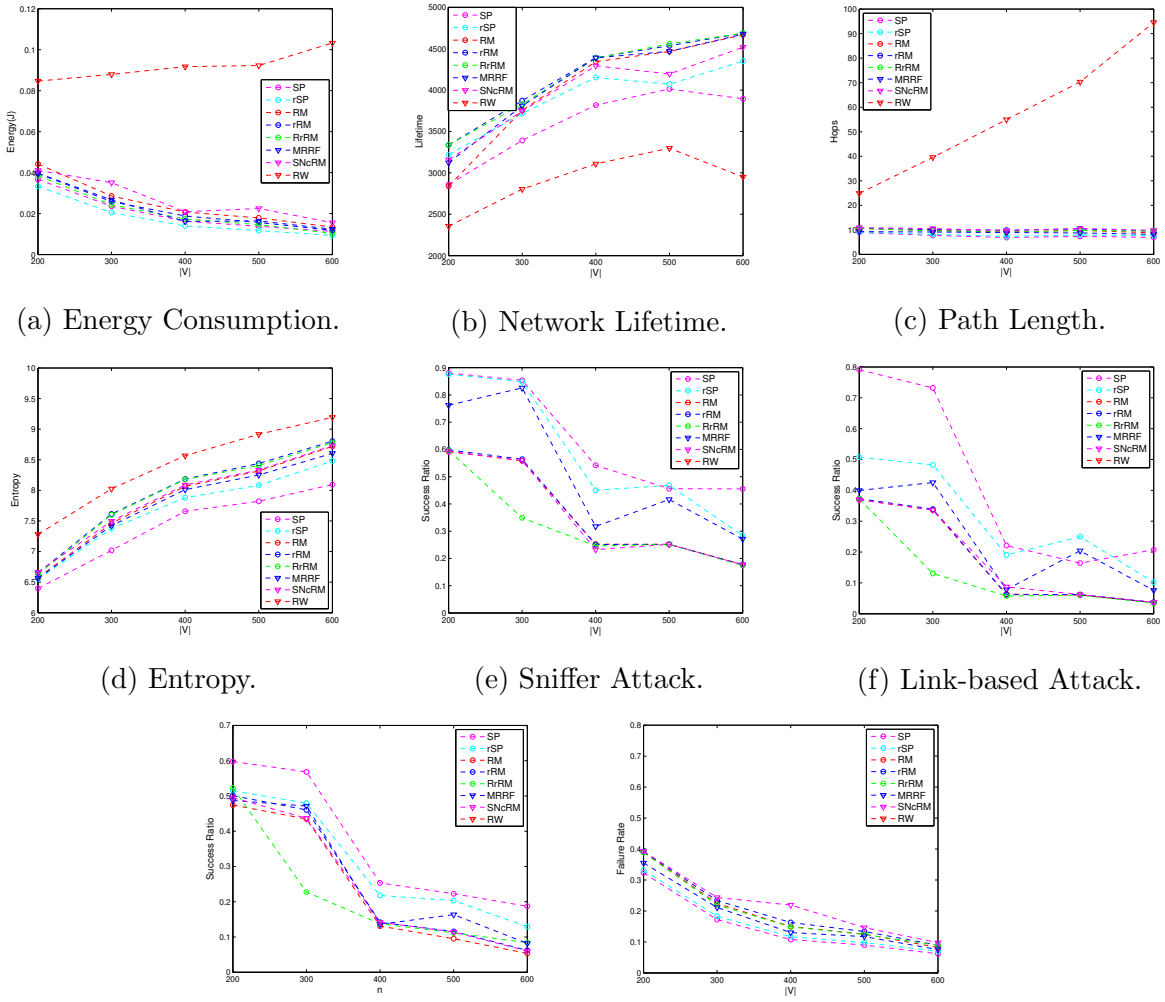
Failure Rate and h-Steps

Heuristic attack failure rate is defined by the deadlock rate described in section 4.4.2. h-steps determine the average number of steps an adversary takes to identify the sink node using the heuristic attack.

4.4.4 Performance Analysis

Route Mutation

Fig. 4.7 shows the network and security performance of SP, rSP, RM, rRM, RrRM, MRRF [87], SNcRM [34], and RW mechanisms. In this figure, five different network sizes are considered where the number of nodes ($|V|$) is 200, 300, 400, 500, or 600. As expected, entropy is lowest for SP and highest for RW (Fig. 4.7d). The entropy is lower with no mutated routes in SP and rSP because flows can pass a particular node repeatedly with no restriction. The entropy is higher in rRM and RrRM than in MRRF and SNcRM due to the security constraints of route selection in terms of similarity s and history h scores. More traffic distribution occurs in the ranking-based mechanisms (rRM and RrRM) than the RM. In Fig. 4.7h, the average deadlock rate for heuristic attack correlates with the entropy values shown in Fig. 4.7d. Higher entropy corresponds to a larger deadlock rate. This implies that entropy is a useful metric to measure the efficiency of the route obfuscation scheme. Without applying route mutation, the deadlock rate drops approximately 25%

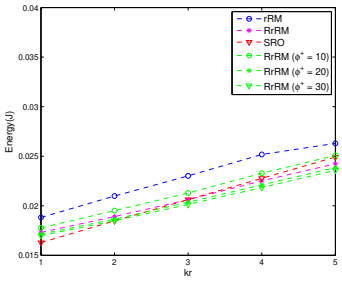


(a) Energy Consumption. (b) Network Lifetime. (c) Path Length.

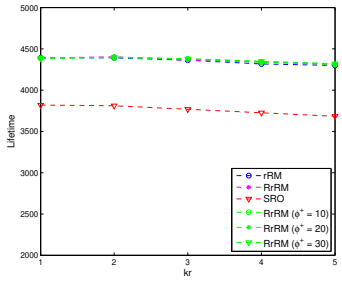
(d) Entropy. (e) Sniffer Attack. (f) Link-based Attack.

(g) CrossPath Attack. (h) Heuristic Attack.

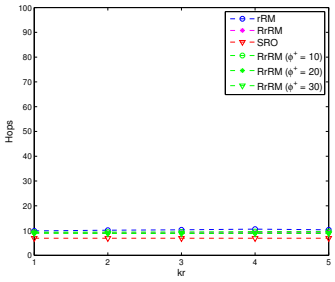
Figure 4.7: Comparison between the proposed route obfuscation solutions and the state of art.



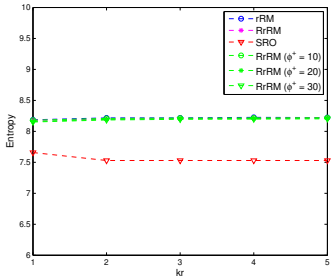
(a) Energy Consumption.



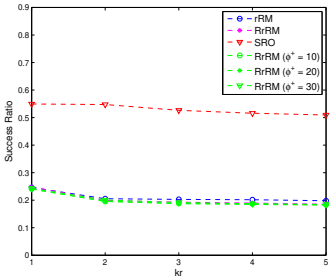
(b) Network Lifetime.



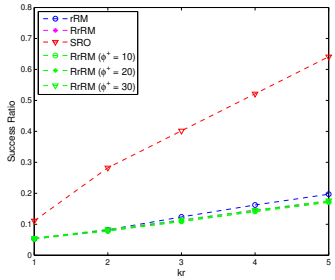
(c) Path Length.



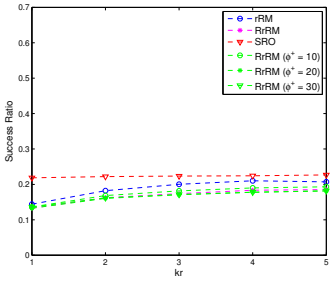
(d) Entropy.



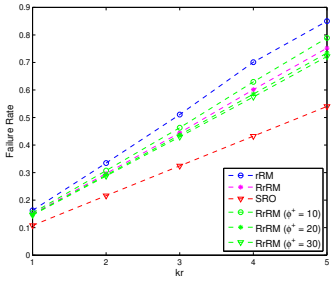
(e) Sniffer Attack.



(f) Link-based Attack.



(g) CrossPath Attack.



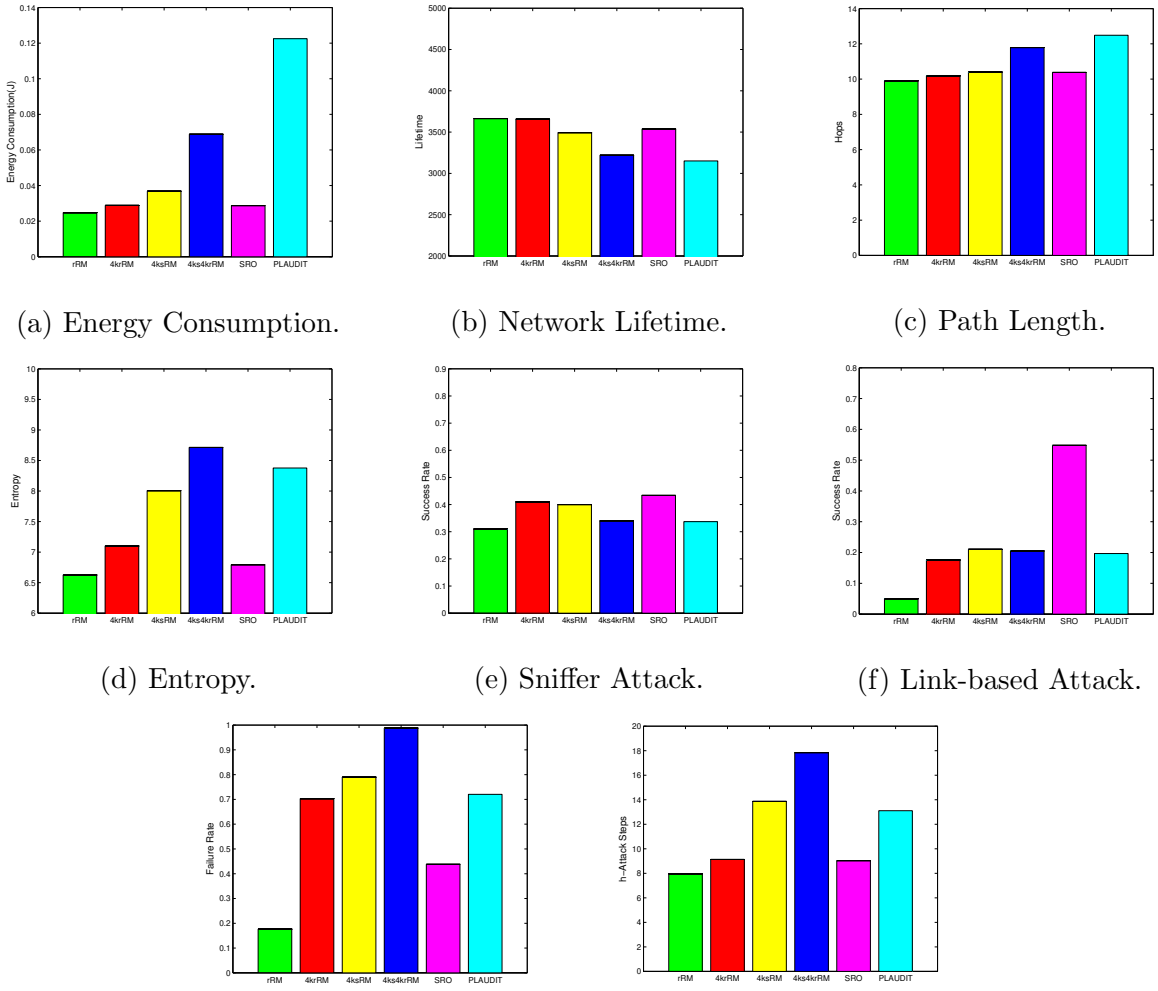
(h) Heuristic Attack.

Figure 4.8: The influence of the number of mutated routes k_r.

as the adversary can easily obtain the shared paths due to the pattern of communication traffic. Using the ranking approach decreases the success rate because the next hop can differ with time for the same flow. Clearly that, SP and rSP show the worse defense performance against the sniffer, link-based, and CrossPath attacks. RM and rRM show a good defensive performance but lower than RrRM due to the additional randomness of choosing the next hop in terms of s and h scores. The success rate of the sniffer attack scenario in MRRF is higher than other routing mutation mechanisms. SP has the lowest obfuscation level because the node can be selected repeatedly as part of the routes. rRM and RrRM have a higher obfuscation level similar to SNcRM (Fig. 4.7e). MRRF has no security guarantee in selecting the mutated routes; hence, it fails to protect the network. In Fig. 4.7a, SP has worse energy consumption results than that of rSP. Likewise, RM and SNcRM have a worse energy consumption result than the other route mutation mechanisms. This is because they do not consider the link cost including the energy constraints. In Fig. 4.7b, the ranking-based RMs mechanisms show a higher network lifetime due to the load balancing of the route assignment among nodes. Their network lifetime even better than the energy-aware shortest path (rSP). SP has a lower network lifetime due to the repeated selection of specific nodes. MRRF shows a good lifetime result as the objective function is formulated to save the energy of the nodes. In Fig. 4.7c, the ranking-based RMs and SNcRM mechanisms have a slightly higher path length. This is acceptable compared to the RW scheme. SP and rSP have the lowest path length as they are the shortest path mechanisms. RW scheme has the highest defense performance in terms of sniffer, link-based, and CrossPath attacks. However, Fig. 4.7 shows that RW has the poorest network performance. RW has a very high energy consumption which results in the shortest network's lifetime. In addition, the average path length of the generated routes is five times (when $|V| = 400$) that of the next highest scheme.

Multiple Mutated Routes (k_r)

Fig. 4.8 shows the influence of the number of mutated routes k_r for rRM, RrRM, and SRO [37]. The proposed route obfuscation techniques have a better network and defense performance than SRO. Fig. 4.8d shows the influence of k_r on the average entropy. As the entropy determines the randomness of network traffic, it slightly increases with the increase of k_r. In Fig. 4.8h, as k_r increases, the number of heuristic attack's deadlocks increases due to the distribution of traffic. The success rate of the sniffer attack scenario is slightly decreased when the number of mutated routes is increased (Fig. 4.8e). However, the success rate of other attack scenarios increases due to the increase in the number of overlapping points in data and/or control paths (Fig.s 4.8f and 4.8g). Fig. 4.8a shows that



(a) Energy Consumption. (b) Network Lifetime. (c) Path Length.

(d) Entropy. (e) Sniffer Attack. (f) Link-based Attack.

(g) Heuristic Attack. (h) Heuristic Attack Steps.

Figure 4.9: Comparison between the proposed mechanism and the state of art in fully centric WSNs.

the energy consumption increases as the number of mutated routes (k_r) increase. This is an acceptable increase as the traffic volume is multiplied by the k_r factor. The network lifetime and path length are also linearly degraded (Fig.s 4.8b and 4.8c). RrRM has better network performance than rRM when $\phi^+ = 15$ (Fig.s 4.8a, 4.8b, and 4.8c). the effect of ϕ^+ values is discussed in section 4.4.4.

The Influence of ϕ^+

Fig. 4.8 also shows the influence of the ϕ^+ parameter. Φ_{max} is a dynamic threshold that determines the optimal similarity s and history h scores. ϕ^+ parameter determines the increased steps of Φ_{max} when constructed paths are invalid due to the s and h scores. ϕ^+ parameter determines how much the selection is strict to find the optimal path. The entropy slightly degrades when the ϕ^+ increases (Fig. 4.8d); hence, there is an increase in the success rate of the studied attacks. Higher ϕ^+ means expanding the candidate list of nodes to be selected for the next hop, hence, a higher consideration to the node/link cost. A lower ϕ^+ leads to fewer candidates due to the strict selection. Increasing the ϕ^+ parameter degrades the defense performance, but it improves the network performance. In the end, this research aims to balance network protection and functionality.

Sink Obfuscation

Fig. 4.9 provides a comparison between the proposed mechanisms (rRM, 4krRM, 4ksRM, and 4ks4krRM) and the state of art (SRO [37] and PLAUDIT [38]) in fully centric WSNs. In this figure, all the network traffic (data and control flows) happens between the sink node and the sensor nodes. In Fig. 4.9d, there is a greater increase of entropy with four fake sink nodes (4ksRM and 4ks4krRM). However, with four mutated routes, the entropy is the highest because the traffic is more evenly distributed around the real and fake sink nodes (4ks4krRM). This shows that the idea of generating multiple routes and multiple fake sink nodes in a controlled manner does aid in making the network traffic pattern more random. Fig. 4.9h shows the average heuristic attack steps to reach the sink node. Having four fake sink nodes dramatically increases the attack steps due to the local maxima. Moreover, integrating route obfuscation and sink obfuscation (4ks4krRM) results in a higher number of attack steps. In Fig. 4.9a, having four fake sink nodes increases the energy consumption as extra traffic is generated. 4krRM and 4ksRM have smaller energy consumption than 4ks4krRM, while PLAUDIT uses a higher energy consumption due to the higher traffic volume. However, 4ks4krRM provides better performance in terms of entropy and heuristic attack defense. Its effect on network parameters can be considered as the cost of better

performance. Fig. 4.9 shows that SRO fails to defend the network that is fully centric WSNs. However, the deceptive traffic of PLAUDIT provides good defensive performance but not better than 4ks4krRM. Moreover, PLAUDIT has poor network performance, such as very high energy consumption and low network lifetime.

4.4.5 Discussion

Algorithm 4 solves a composite routing problem by finding paths for the given flow set using several network performance parameters and security metrics. In other words, a path is selected under similarity and history constraints with the least path cost that is computed by equation 4.6. The path cost is determined based on the residual energy of nodes and the expected energy consumption, as well as the node capacity and reliability. Moreover, the selected paths are restricted to a maximum path length to ensure QoS requirements. Consequently, the results of the proposed mechanism show better network performance, such as lower energy consumption, lower lifetime, and lesser path length associated with higher network protection. The security performance is investigated under several types of traffic analysis attacks, namely, node-based attack (sniffer), link-based attack, and control attack (CrossPath). The proposed mechanism protects the network by hiding the network topology and obfuscating the traffic. The network traffic does not accumulate on the high-profile nodes as the network lessens relying on specific nodes without harming the network functionality. The heuristic attack is more complex; thus, a sink obfuscation algorithm is proposed. Results show that having multiple fake sink nodes degrades the network performance. Thus, Algorithm 5 balances the performance degradation and security gain by carefully selecting the fake sink nodes based on several criteria, including the distance between fake and real sink nodes and the energy level of fake sink nodes and their neighbors.

Chapter 5

Security Networking Functions Placement for Software-defined WSNs

In Chapter 3, we have proposed a trust management scheme that utilizes sensor nodes as watchdog nodes. In this chapter, a secure and energy-aware watchdog placement solution is proposed for software-defined WSNs. The solution balances the required energy consumption and computational resource, and security in terms of the honesty of watchdog nodes. The objective is to minimize the energy consumption yielded due to the watchdog functions while maintaining the activation of sufficient trustable watchdog nodes.

5.1 System and Threat Models

In this section, the system and threat models are presented. Also, a list of notations is given in Table 5.1.

5.1.1 System Model

Consider an SDN-enabled WSN, $G = (V, E)$, where V and E are the set of sensor nodes and wireless links (as shown in Fig. 5.1). The sensor nodes can be forwarding nodes ($\forall v \in V$), target nodes (represented by set Γ), and watchdog nodes (represented by set W). Forwarding nodes are capable of routing traffic based on their flow table that is updated by the

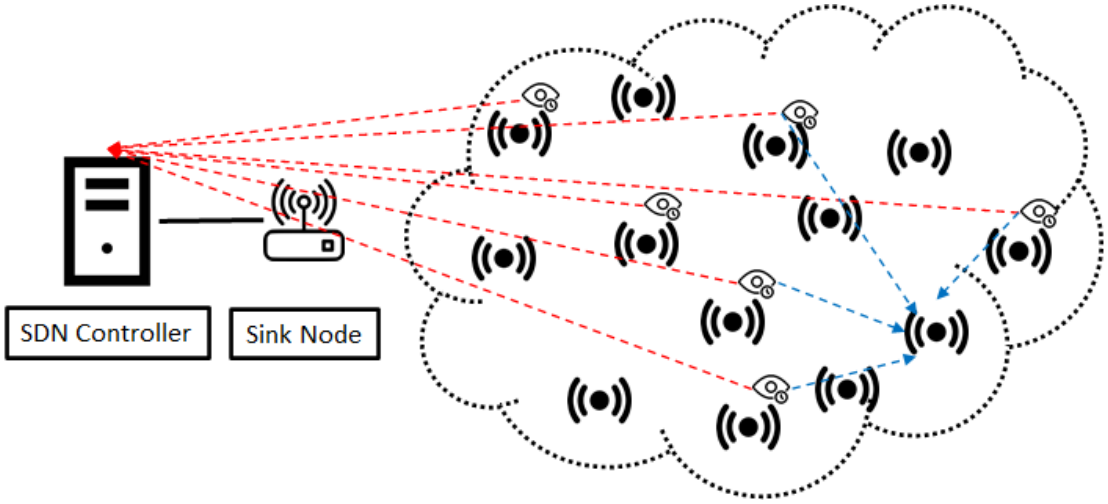


Figure 5.1: System Model.

SDN controller. All nodes can host and operate security network functions (i.e., watchdog). We assume that each watchdog node has a forwarding capability, a capability to overhear of other nodes' communication within its range, and available computational resources. A watchdog node w has available computational capability ϕ_w^p and memory capability ϕ_w^m. Moreover, a watchdog node can overhear a number of watchdog targets simultaneously as long as the available computational, memory, and energy resources satisfy the requirements. $|V|$ is the number of nodes in the WSN. $|\Gamma|$ is the number of targets in the WSN where usually $|V| = |\Gamma|$. $|W|$ is the number of assigned (activated) watchdog nodes where $|W| <= |V|$.

Each sensor node in the network is a target ($\tau \in \Gamma$) of the watchdog task and must be assigned to a set of watchdog nodes. Any node can be a watchdog node ($w \in W$) for one or more target nodes. The watchdog node can perform a watchdog task for a certain target only if the distance between the watchdog and the target nodes is less than the communication range of the watchdog and the target nodes. As a result, all possible watchdog nodes can receive and send messages as well as overhear as part of the watchdog function. The matrix of neighbor nodes N determines if a node can perform the watchdog function for other nodes. Thus, every node within the communication range of the target is a possible watchdog node candidate to be a watchdog node for this target. In this research, without loss of generality, we consider that all nodes have the same communication range (θ). Fig. 5.2 shows an example of three nodes associated with the distances (d) between these nodes.

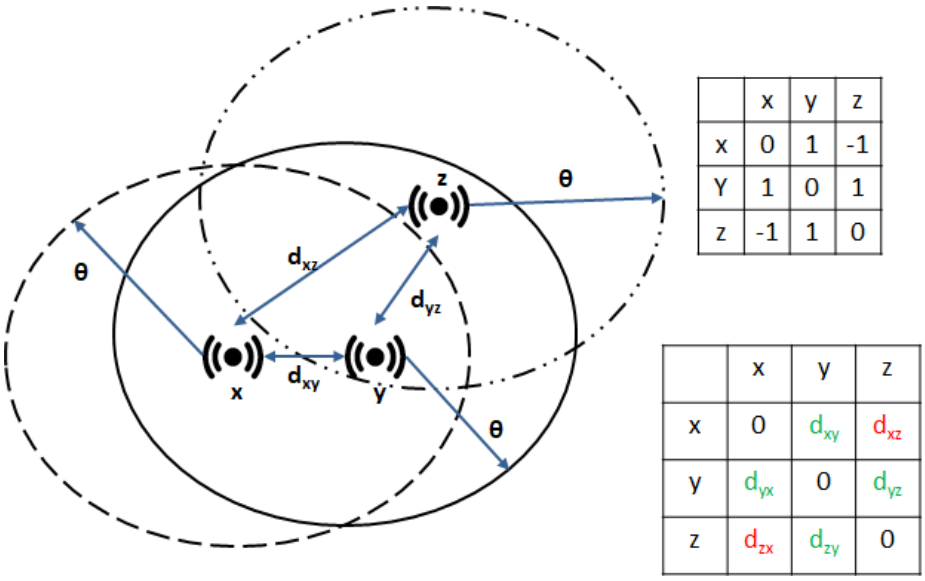


Figure 5.2: An example of three nodes associated with the distances (d) between these nodes.

$$n_w^\tau = \begin{cases} 1 & , d_{w,\tau} \leq \theta \\ 0 & , \text{otherwise.} \end{cases} \tag{5.1}$$

A popular energy model is used to determine the energy consumption of message transmission [104]. $E_{Tx}(l,d)$ is consumed by the transmitter node when l bytes message is sent for distance d, while $E_{Rx}(l)$ is consumed by the reciever node:

$$E_{Tx}(l,d) = \begin{cases} lE_{elec} + l\epsilon_{fs}d^2 & , d < d_0 \\ lE_{elec} + l\epsilon_{amp}d^4 & , d \geq d_0 \end{cases} \tag{5.2}$$

$$E_{Rx}(l) = lE_{elec} \tag{5.3}$$

where E_{elec} is 50 nJ/bit, which is the transmission circuit loss. The power amplification is ϵ_{fs} (10 pJ/bit/m^2) for the free sapce channel model, and ϵ_{amp} (0.0013 pJ/bit/m^4) for the multi-path fading channel model [99].

Table 5.1: Table of Notations for Security Function Placement Problem

Notations	Descriptions
$G(V, E)$	Network graph G, where V is the set of vertices (nodes) and E is the set of edges (links).
N	Neighbor matrix which is $\|V\|$ by $\|V\|$ Boolean matrix.
Γ	Set of target nodes of watchdog task.
W	Set of watchdog nodes.
P^{τ}	Set of possible watchdog nodes for node τ.
$\|set\|$	The number of elements belongs to the set.
$\ddot{W}^{min}$	Minimum number (degree) of watchdog nodes for a target.
θ	Communication range of the sensor node.
d_{ij}	Distance between node i and j.
$E_{Tx}(l, d)$	The energy consumed to transmit l bytes for distance d.
$E_{Rx}(l)$	The energy consumed to receive l bytes.
E_v	The current (residual) energy level of node v.
$\hat{E}_w^{\tau}$	The expected energy consumption of node w to perform watchdog task over node τ.
H_w	The honesty score of watchdog node w that evaluates the trustworthiness of this node to evaluate the target nodes.
T_w^{τ}	Trust score computed by a watchdog w for a target τ.
A	Activation matrix which is $\|V\|$ by $\|V\|$ Boolean matrix.
a_w^{τ}	Element of $\|A\|$ that determines if node w is activated as a watchdog node for target node τ.
ϕ_v^p	The available computational capabilities of the node v.
ϕ_v^m	The available memory capabilities of the node v.
Φ^p, Φ^m	The required computational and memory capabilities to perform the watchdog task over one node.

5.1.2 Trust Model

A trust management model is deployed in the network [101]. At the sensor layer, trust scores are calculated by watchdog nodes to evaluate the target nodes ($T_w^\tau = trust\_model(w \rightarrow \tau), \forall w \in W$). Then, the calculated trust scores are sent to the controller regularly. The aggregated trust scores for the target nodes are calculated at the controller. A trust map is built by the controller that helps to support reliable network functions and services. Furthermore, the trust model secures the network from malicious nodes that perform several communication threats. There are two types of watchdog functions, namely passive and active:

Passive Watchdog Function

In this scenario, the watchdog node only overhears its assigned target nodes. To perform the watchdog function, the watchdog node consumes the amount of energy that is equal to the energy consumption of receiving a message:

$$E_{w,\tau}^{WD} = E_{Rx}(l) + \alpha. \tag{5.4}$$

where α is the energy consumed for processing the received data and extracting the needed information for trust management. The consumed energy for reporting to the controller is neglected as we assume that a watchdog node groups all reports of the assigned target nodes in one message.

Active Watchdog Function

In this scenario, the watchdog node may have to flood its neighbors with a number of messages and then overhear the actions of its assigned target/s after receiving them. The frequency of flooding is determined by a heuristic adjustment algorithm presented in [50]. Both the watchdog and the target nodes will consume energy. In this case, the energy consumption of the watchdog task is a function of the distance between the watchdog w and the target τ nodes:

$$E_{w,\tau}^{WD} = 2[E_{Tx}(l, d_{w\tau}) + E_{Rx}(l)] + \alpha, \tag{5.5}$$

5.1.3 Threat Model

Software-defined WSNs face several communication threats such as message flooding and dropping attacks. DoS attack aims to degrade the network availability of WSNs. For example, a new-flow attack targets both the data and the control plane by multiple packet-in messages in software-defined WSNs. In a black-hole attack, all the received packets are dropped by the malicious node. On the other hand, the malicious node partially drops packets randomly or deliberately in the selective forwarding attack.

In this research, the malicious nodes also perform mouthing attacks to deceive the trust model by falsifying the trust reporting. There are two types of mouthing attacks: good and bad mouthing. In the good-mouthing attack, the malicious node tries to misrepresent the global trust for another malicious node by providing biased high trust scores. In the bad-mouthing attack, the malicious node tries to degrade the trust of a non-malicious node by providing biased low trust scores. Possibly, multiple mouthing nodes can perform a collaborative mouthing attack to raise or degrade the trust score of other nodes.

5.2 Problem Formulation

The goal is to minimize the energy consumption of the watchdog function that occurs due to evaluation operations such as overhearing, computational models, and reporting overhead. The overhearing operation causes a large amount of energy consumption and could negatively affect the lifetime of such a limited resource network. Therefore, there is a need to optimize the watchdog placement in the network.

5.2.1 Objective Function

The objective function is to minimize the total expected power consumption of watchdog functions by all nodes in the network.

$$\min \sum_{\forall \tau \in \Gamma} \sum_{\forall w \in W} a_w^\tau \hat{E}_w^\tau + \max \sum_{\forall \tau \in \Gamma} \sum_{\forall w \in W} a_w^\tau H_w \tag{5.6}$$

where A is the activation matrix which is a two-dimension matrix ($|V|$x$|V|$) that has binary values. Fig. 5.3 shows the activation matrix representation. Fig. 5.4 shows an example of possible raters for certain target and the activated ones after the optimization solution. $\hat{E}_w^\tau$ is the expected energy consumption of watchdog node w to perform watchdog

	w_1	w_2	...	$w_{\|V\|}$
τ_1	a_1^1	a_2^1	...	$a_{\|V\|}^1$
τ_2	a_1^2	a_2^2	...	$a_{\|V\|}^2$
...	...	...	...	...
$\tau_{\|V\|}$	$a_1^{\|V\|}$	$a_2^{\|V\|}$	...	$a_{\|V\|}^{\|V\|}$

Figure 5.3: Activation matrix.

task over target node τ. H_w is the honesty score of watchdog node w that evaluates the trustworthiness of this node to evaluate the target nodes.

$$a_w^\tau = \begin{cases} 1 & \text{, watchdog node } w \text{ is activated for target node } \tau \\ 0 & \text{, otherwise.} \end{cases} \tag{5.7}$$

5.2.2 Connectivity

The connectivity constraint ensures that only the neighbor nodes of a target node are activated as watchdog nodes:

$$n_w^\tau \geq a_w^\tau \ , \forall \tau \in \Gamma, \forall w \in W \tag{5.8}$$

5.2.3 Computational and Memory Resources

To ensure that the required computational and memory capabilities of the assigned watchdog functions do not exceed the available capabilities of the activated watchdog nodes, a computational and memory constraints is used as follows:

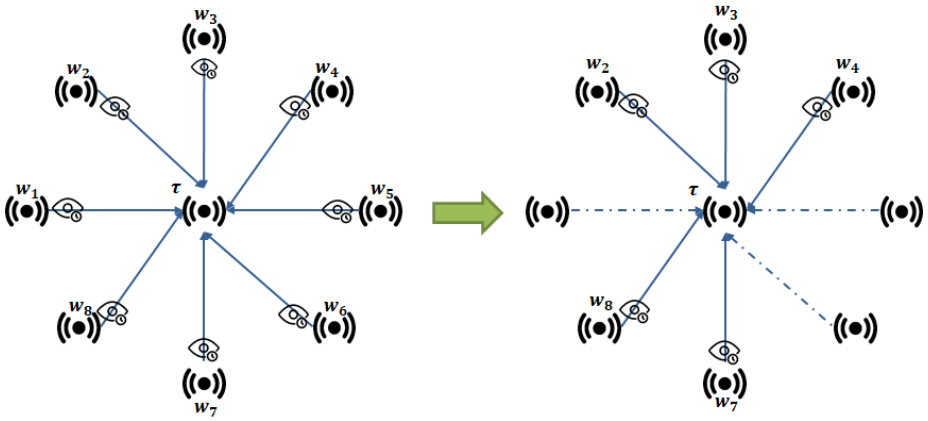


Figure 5.4: An example of possible raters for certain target and the activated ones after the optimization solution.

$$\sum_{\tau \in \Gamma} a_w^\tau \ \Phi^p \leq \phi_w^p \qquad ,\forall w \in W \tag{5.9}$$

$$\sum_{\tau \in \Gamma} a_w^\tau \ \Phi^m \leq \phi_w^m \qquad ,\forall w \in W \tag{5.10}$$

where Φ^p and Φ^m are the computational and memory requirements for one watchdog task, respectively. ϕ_w^p and ϕ_w^m are the available computational and memory capabilities of the watchdog node w. Φ^p and Φ^m are determined based on the number of trust parameters and metrics used in the trust model [101].

5.2.4 Honest Watchdog

The controller validates the honesty of the watchdog nodes for undertaking the watchdog task and not launching the mouthing attacks. The watchdog node's honesty score (H_w) evaluates node w's trust evaluations of the target nodes. To compute this score, first, the average trust score (T_{avg}^τ) of the aggregated trust scores of a target node τ is calculated:

$$T_{avg}^\tau = \frac{\sum_{w \in W} T_w^\tau}{\sum_{w \in W} a_w^\tau} \tag{5.11}$$

Then, for all watchdog nodes, the difference between the calculated trust score by a watchdog node w and the average T^{τ}_{avg} is computed div^{τ}_{w}:

$$div^{\tau}_{w} = 1 - |T^{\tau}_{w} - T^{\tau}_{avg}| \tag{5.12}$$

A watchdog node w can be assigned to multiple targets. Thus, the honesty of a watchdog node can be calculated by using the average of div_w values as follows:

$$H_w = \frac{\sum_{\tau \in \Gamma} a^{\tau}_{w} div^{\tau}_{w}}{\sum_{\tau \in \Gamma} a^{\tau}_{w}} \tag{5.13}$$

The summation of the honesty score of the activated watchdogs must be maximized (equation 5.6). On the other hand, if the calculated honesty score of a watchdog node w is in the trusted zone (i.e., $H_w \geq th_H$ where th_H is the trusted threshold [101]), then this node can be trusted to perform watchdog tasks in the next round. Thus, watchdog nodes that submit biased evaluations or launch mouthing attacks can be avoided and isolated. As a result, every watchdog node is associated with an honesty score based on the aggregated trust score in the previous round of the watchdog task. The watchdog node w is dishonest when it provides biased scores about target nodes.

$$h_w = \begin{cases} 1 & , H_w \geq th_H \\ 0 & , \text{otherwise.} \end{cases} \tag{5.14}$$

As a result, the set of possible watchdog nodes for a target node τ (P^{τ}) is comprised of all neighbor nodes of τ that are honest ($H_w \geq th_H$) and not malicious ($T_w \geq th_H$):

$$p^{\tau}_{w} = n^{\tau}_{w} h_w \ , \forall \tau \in \Gamma, \forall w \in W \tag{5.15}$$

5.2.5 Degree of Watchdog Function

The number of activated raters for any target must be at maximum but with minimum energy consumption. However, this might lead to affecting the precision of the watchdog function. The minimum number of activated watchdog nodes for any target node is defined as the degree of watchdog function $\ddot{W}^{min}$.

$$\ddot{W}^{min}_{\tau} = \begin{cases} \ddot{W}^{min} & , \ddot{W}^{min} \leq \sum_{w \in W} p^{\tau}_{w} \\ \sum_{w \in W} p^{\tau}_{w} & , \text{otherwise.} \end{cases} , \forall \tau \in \Gamma \tag{5.16}$$

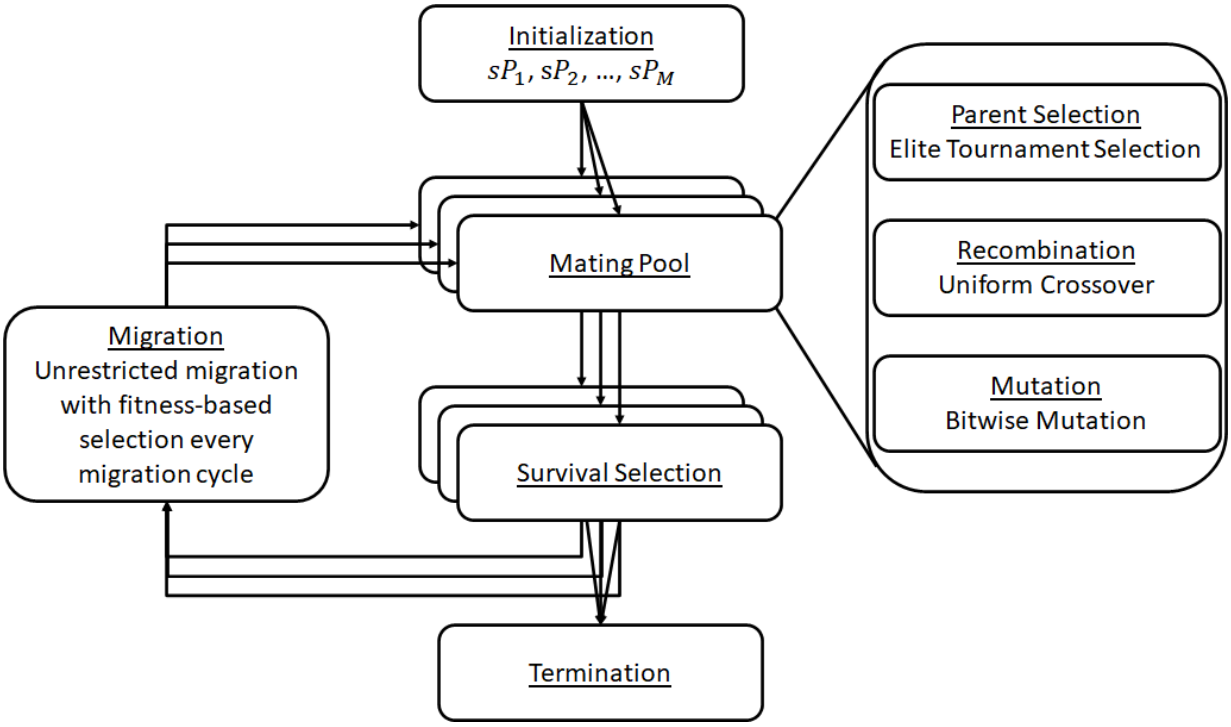


Figure 5.5: Multi-population Genetic Algorithm Overview.

The degree of watchdog function for any target node is $\ddot{W}^{min}$. However, if the number of possible watchdog nodes for a specific target is less than or equal to $\ddot{W}^{min}$, then all these watchdog nodes must be activated. Thus, the coverage constraint is formulated as follows:

$$\sum_{w \in W} p_w^\tau a_w^\tau \geq \ddot{W}_\tau^{min} \ , \forall \tau \in \Gamma \tag{5.17}$$

5.3 Heuristic Algorithm

The watchdog function placement problem shown in equation 5.6 is an NP-hard problem. A meta-heuristic algorithm is designed to find optimal or near-optimal solutions in a practical period. Genetic algorithm [105] is an evolutionary, stochastic, and population-based algorithm that is suitable and has been considerably utilized to solve several problems [106]. A multi-population genetic algorithm is used to solve the placement problem in this research. An overview of the algorithm is presented in Fig. 5.5.

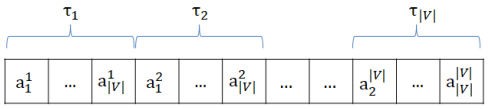


Figure 5.6: Chromosome Representation.

5.3.1 Genetic Representation and Fitness Function

In the proposed algorithm, each chromosome $ch = \{g_0, g_1, ..., g_{|V|*|V|}\}$ is made of $|V|$ by $|V|$ genes (as shown in Fig. 5.6). The chromosome represents the activation matrix (A) of watchdog nodes. The search space is complete and feasible in the proposed heuristic algorithm, and the optimal solution can be found in this space.

The algorithm evaluates the quality of the chromosome based on coverage, expected energy consumption, and honesty metrics. First, to ensure the coverage degree of all target nodes, the algorithm looks for target nodes that have been assigned to watchdog nodes that are less than $\ddot{W}_{\tau}^{min}$ ($\forall \tau \in \Gamma$). The number of missing watchdog nodes for all target nodes is N_{miss}:

$$N_{miss} = \sum_{\forall \tau \in \Gamma} N_{miss}^{\tau} \tag{5.18}$$

The cost of uncovered target and missing watchdog nodes must be very high compared to other costs (energy and honesty):

$$c_{miss} = \frac{\hat{I} N_{miss}}{\sum_{\forall \tau \in \Gamma} \ddot{W}_{\tau}^{min}}. \tag{5.19}$$

where $\hat{I}$ is the maximum value for the numbering data type that is used. As a result, the cost is equal to the largest value when no watchdog nodes are activated ($N_{miss} = \ddot{W}_{\tau}^{min}$). The algorithm needs only a few iterations to converge to solutions that have all target nodes assigned to watchdog nodes with different coverage degrees. On the other hand, the cost of the expected energy consumption from the assignment activation is c_E, which depends on the number of assigned target nodes of each activated watchdog node (passive case), or it is proportional to the distances between the activated watchdog and the assigned target nodes (active case). This cost is calculated as follows:

$$c_E = \begin{cases} \sum_{\forall \tau \in \Gamma} \sum_{\forall w \in W} (a_w^{\tau} \mu) & \text{, passive case} \\ \sum_{\forall \tau \in \Gamma} \sum_{\forall w \in W} (a_w^{\tau} \hat{\mu} d_{w\tau}) & \text{, active case,} \end{cases} \tag{5.20}$$

where μ and $\hat{\mu}$ are scalar factors.

The cost of the solution is also dependent on the honesty of the activated watchdogs. The number of activated watchdog nodes with a higher honesty score must be maximized. Thus, the security cost (c_H) is defined as follows:

$$c_H = \frac{\beta}{\sum_{\forall w \in W}(H_w \sum_{\forall \tau \in \Gamma} a_w^\tau)} \tag{5.21}$$

where β is a scalar factor to normalize the cost value. Therefore, the total cost $c(A)$ of any chromosome is computed as follows:

$$c(A) = c_{miss} + c_E + c_H \tag{5.22}$$

The solution quality (chromosome) is evaluated based on the fitness function that is determined by the cost. Larger cost value indicates less fitness and lower chromosome quality.

5.3.2 Multiple Populations

One of the challenges of genetic algorithms is premature convergence [107]. Multi-population [108] is usually suggested to avoid this issue by splitting the population into M independent subpopulations with equal subpopulation size (where $|X|$ is the number of chromosomes in each subpopulation). Thus, the exchange among subpopulations is implemented after each subpopulation has evolved independently for a number of generations. The migration model determines how the multiple subpopulations communicate. The separated subpopulations are processed independently for specific generations rounds, usually called isolation time. Some of the individuals are exchanged among the subpopulations every migration time. Fig. 5.7 shows a description of the migration model used in the proposed algorithm. This is a fitness-based unrestricted (best-random) migration strategy where chromosomes migrate from one subpopulation to another [109]. Each subpopulation gets one new chromosome from the migration pool of probable immigrants. The migration pool is constructed by taking the fittest chromosome (the highest fitness value, lowest cost) from each subpopulation; thus, the pool size is M. The migrated individual is selected randomly from this pool (except the original subpopulation). In Fig. 5.7, subpopulation sP_M will receive the new chromosome; thus all subpopulations except sP_M contribute to the migration pool with their fittest chromosome. As shown in Fig. 5.7, the lighter the chromosome color, the less fit the chromosome. The chromosome contributed by sP_1 is randomly selected from the migration pool. As a result, this chromosome substitutes the

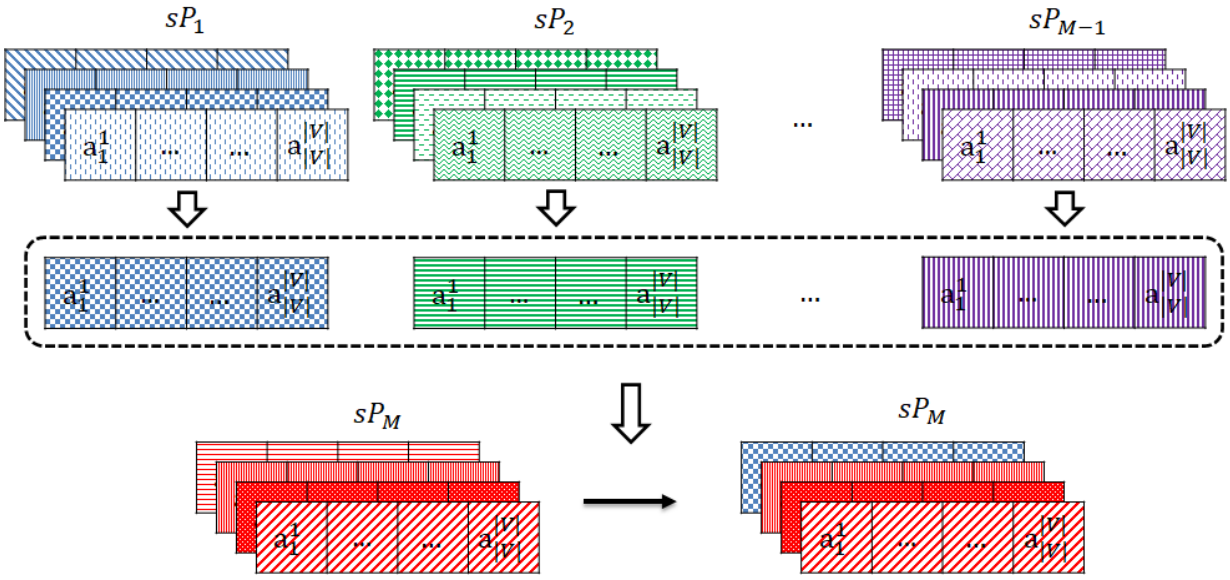


Figure 5.7: Migration strategy of chromosomes between subpopulations.

poorest chromosome (in terms of fitness function) in subpopulation sP_M. Each subpopulation will complete this process when the isolation time ends. Therefore, all subpopulations randomly communicate with each other and exchange individuals.

5.3.3 Selection and Recombination

Parent selection specifies which chromosomes are selected for the recombination process and the generated number of offspring by each selected individual. A rank-based fitness assignment is used in this algorithm. Tournament selection, as one of the most widely used selection strategies, provides more diversity in the selection procedure with smaller tournament size. The tournament selection operator does not need global population knowledge; rather, it depends on an arrangement association to rank any two chromosomes. This algorithm uses the elite tournament selection [110] in which the reproduction of the best solution is more likely.

For recombination, a uniform crossover operator is used. Uniform crossover [105] operates by processing each gene individually and randomly selecting of the parent chromosome from which the offspring is created. Fig. 5.8 provides an example of a uniform crossover operator that is used in the proposed algorithm. Two parents with a size of 10 are selected from the mating pool. A row of 10 random values is generated uniformly in the range of

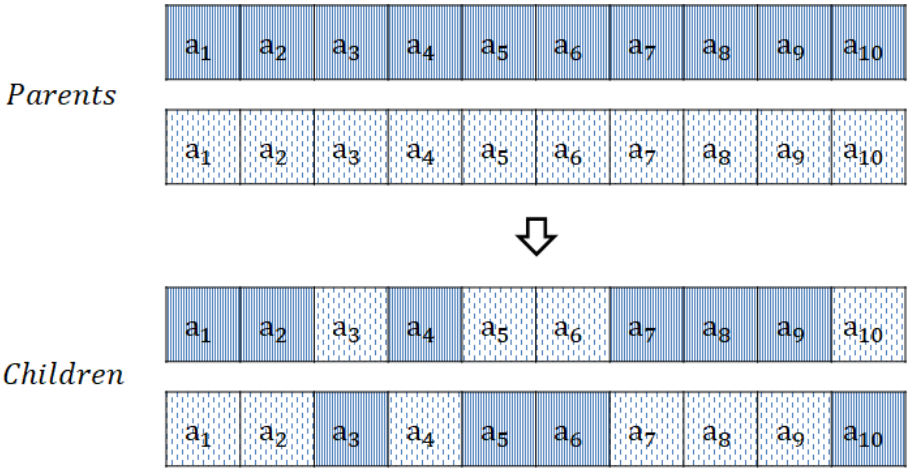


Figure 5.8: uniform crossover operator.

$[0, 1]$. For each gene, the generated random value is compared with the crossover probability ($p_c = 0.5$); the interchange occurs when the random value is greater than p_c. Thus, in Fig. 5.8, at genes 3, 5, 6, and 10, the associated random values are greater than p_c. As a result, for the first child, these genes are inherited from the second parent, while the rest of the genes are inherited from the first parent. Similarly, the second child is generated through the inverse mapping of the first child.

5.3.4 Mutation

In this algorithm, the bitwise mutation operator [105], which is one of the most typical mutation operators, is used. This operator treats each gene independently and lets each gene value be inverted with a small mutation probability ($P_m = 0.2$). Therefore, the number of inverted genes is not fixed; instead, it depends on the row of random values of the chromosome size (on average $|V||V|p_m$). Fig. 5.9 provides an example of a bitwise mutation operator used in the proposed algorithm. The chromosome has a size of 10; thus, a row of 10 random values is generated uniformly in the range of $[0, 1]$. For each gene, the generated random value is compared with the mutation probability ($p_m = 0.2$). In Fig. 5.9, the associated random values are less than p_m at genes 1, 4, 5, and 8. As a result, these genes are inverted; i.e., the new values of these genes are $\bar{a_1}$, $\bar{a_4}$, $\bar{a_5}$, and $\bar{a_8}$, while the rest of the genes keep the same values.

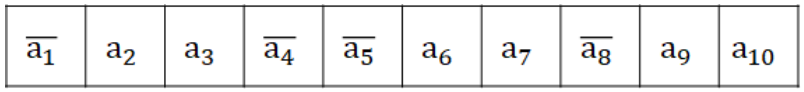


Figure 5.9: Bitwise mutation operator.

5.3.5 Algorithm Overview

Algorithm 6 presents an overview of the multi-population genetic algorithm used to solve the watchdog function placement problem. Firstly, the algorithm generates the initial population (sP). sP is a vector of initial subpopulations with size M, in which each element consists of $|X|$ chromosomes. Three types of initial subpopulations are used in order to have a good initial population as long as subpopulations are independently evolving during the isolation time. First, $M-2$ of the subpopulations are randomly initialized. Second, one subpopulation is initialized with chromosomes that are all 1's (all watchdog nodes are activated). Last, one subpopulation is initialized with chromosomes that are all 0's (no watchdog node is activated). Secondly, each subpopulation is processed separately during the isolation time. In the mating pool, the next generation of chromosomes is created gradually while the two fittest chromosomes from the current iteration are survived. At each iteration of the mating, two parents are selected and recombined. Then, the two generated offsprings are mutated. The fittest chromosome from each subpopulation is added to the migration pool at the end of the isolation time. Then, all subpopulations get a chromosome from the migration pool as described in section 5.3.2. Finally, the fittest chromosome is the output of the algorithm when the termination condition is met. In this algorithm, the fitness score is associated with every chromosome, which is computed based on equation 5.22. It is important to note that the for loop in the algorithm can have a parallel implementation because each subpopulation is processed independently. The parallel version quickens the processing time and utilizes parallel computing.

5.4 Performance Evaluation

In this section, first, the simulation setup is provided. Then, the performance of the proposed watchdog placement solution is numerically analyzed. Last, the proposed solution is compared with state-of-the-art solutions such as EEWO [50] and OWS [78].

Algorithm 6: Watchdog Function Placement using Multi-population Genetic Algorithm

```
Input  : |V|, |X|, M
Output: A
sP =generate_initial_subPopulations(|V|, |X|, M);
while (not termination) do
    for each sP_i ∈ sP do
        while (isolation_time) do
            nextG_sP = ∅;
            while (nextG_sP.size() < |X| − 2) do
                parent1, parent2 = elite_tournament_selection(sP_i);
                x1, x2 = uniform_CX(parent1, parent2);
                offspring1 = bitwise_mutation(x1);
                offspring2 = bitwise_mutation(x2);
                nextG_sP.append(offspring1);
                nextG_sP.append(offspring2);
            end
            nextG_sP.append(get_fittest(sP_i, 2));
            sP_i = nextG_sP;
        end
        migrationPool.append(get_fittest(sP_i, 1));
    end
    sP = migration_function(sP, migrationPool);
end
A = get_fittest(sP, 1);
```

5.4.1 Simulation Setup

We consider different network sizes of 25, 50, 75, and 100 nodes. The sensor nodes are randomly deployed over an area of 600 x 600. The communication range of all nodes is 150. Two types of nodes are considered: non-malicious and malicious nodes. The malicious nodes can launch one of the communication threats described in section 5.1.3. Moreover, they can perform mouthing attacks. The simulation runs in rounds, where several transmission, trust model, and watchdog placement aspects are determined and updated. Usually, The location of sensors affects the experimental outcomes. Consequently, we conducted 1,000 experiments with random positions of nodes; as a result, the average results

are used. The trust management setup in [101] is used. This setup with no optimization of the activated watchdogs is considered the baseline setup. The trust accuracy, coverage degree, and energy saving are used to evaluate the proposed solution. We have considered the watchdog function that uses all available nodes (i.e., all neighbor nodes are activated as watchdog nodes) as a baseline when determining the trust accuracy and energy saving. Thus, trust accuracy T_{acc} is calculated as follows:

$$T_{acc} = \frac{|T_{all} - T_{WD}|}{T_{all}} \tag{5.23}$$

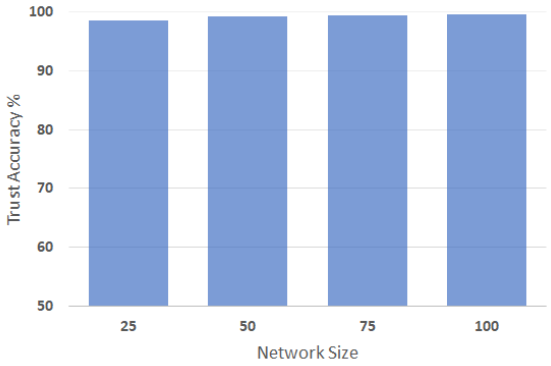
where T_{all} is the average trust score when all nodes are activated, and T_{WD} is the average trust score when placement solution is used. Similarly, energy saving E_{save} is calculated based on the energy consumption as follows:

$$E_{save} = \frac{E_{all}^{cons} - E_{WD}^{cons}}{E_{all}^{cons}} \tag{5.24}$$

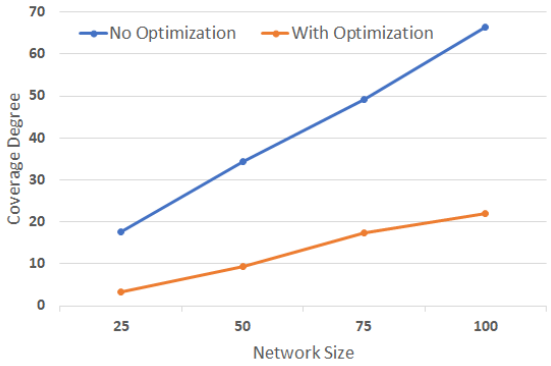
where E_{all}^{cons} is the average energy consumption when all nodes are activated, and E_{WD}^{cons} is the average energy consumption when placement solution is used. On the other hand, the coverage degree is the average number of activated watchdog nodes.

5.4.2 Performance Analysis

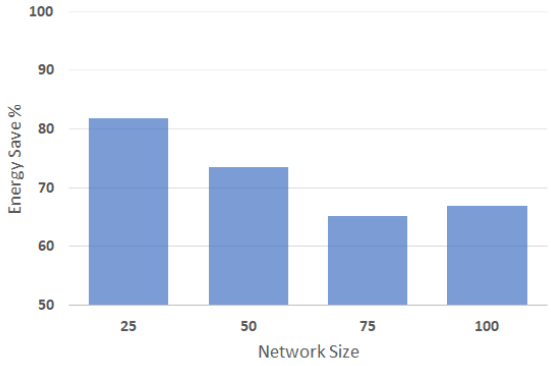
First, the performance of the proposed solution at normal network conditions is evaluated (Fig. 5.10). Fig. 5.10a shows the results of trust accuracy. The accuracy of trust evaluation by the activated watchdog nodes is very high compared to the baseline. This indicates that even with assigning a subset of the available neighbors of nodes, the calculated trust scores are precise. The trust accuracy is higher with higher network density as the number of assigned watchdog nodes to the targets increases. Fig. 5.10b shows the results of coverage degree of the target nodes. The baseline is the average number of neighbor nodes. As shown in the figure, the baseline coverage degree linearly increases with the network size. However, the degree coverage when using the proposed placement solution has a lesser increasing slope. Furthermore, comparing coverage degree of network size 75 and 100, the number of activated watchdog nodes is approximately 20 with no higher increase. This does not affects the trust accuracy. This does impact the energy saving results. Fig. 5.10c shows the results of the energy saving average of the network. The energy consumption of lower density networks is higher due to the wider distances between the nodes (equation 5.2). Therefore, using a subset of the available watchdog nodes saves more energy (network size = 20). Also, dense networks consume more energy due to the increase in the number



(a) Trust accuracy.



(b) Coverage degree.



(c) Energy saving.

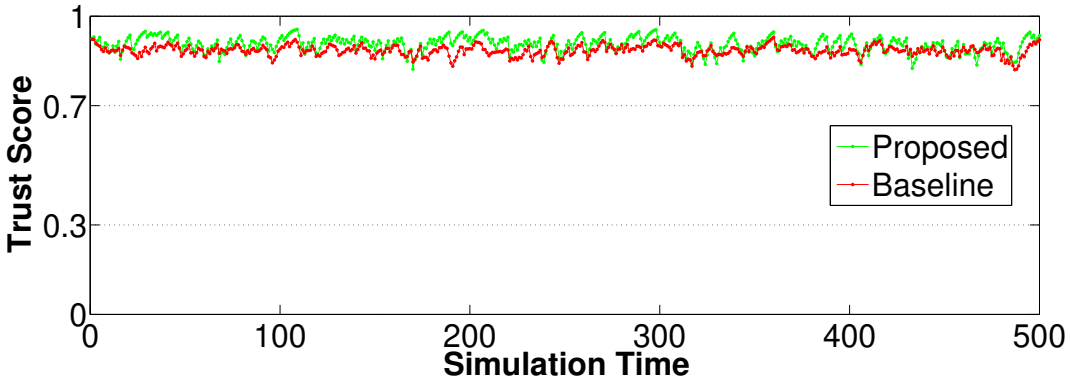
Figure 5.10: Performance of the proposed solution at normal network conditions.

of interactions. However, the energy saving of the proposed solution does not decrease significantly even with the increase of the network size at 75 and 100.

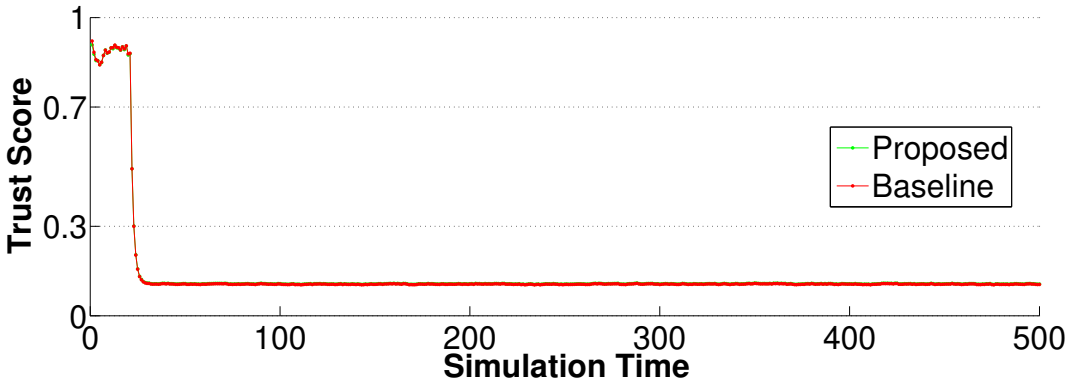
Second, Fig.s 5.11, 5.12, 5.13 and 5.14 present the trust score of a sample setup during the simulation run. The malicious node starts the communication attack at the $20th$ round in these figures scenarios. The presented trust score of the target node (non-malicious or malicious) is calculated by the controller when receiving the aggregated trust scores from the watchdog nodes. Fig. 5.11 shows the trust score for non-malicious and malicious nodes with the proposed placement and the baseline. In Fig. 5.11a, the target node is a non-malicious node that perform its normal tasks faithfully. Fig. 5.11b shows the trust score of a malicious node that performs a selective-forwarding attack (drop rate = 50%). Both figures show that the trust system functions effectively with the proposed solution, which confirms the trust accuracy results in Fig. 5.10a.

In Fig.s 5.12, 5.13 and 5.14, we compare the trust score with and without having set P (equation 5.15). Using set P, non-honest and malicious nodes are eliminated which means the eliminated nodes will not be selected in the next rounds. In these figures, a mouthing attack is launched by a set of watchdog nodes (distributed in the network randomly). In Fig. 5.12, the target node is a non-malicious node. A number of watchdog nodes are launching a bad-mouthing attack to lower its reputation. In the early rounds, the trust score has big jumps below the trust zone because some malicious nodes are selected as watchdog nodes. After that, the trust score has less fluctuation due to the honesty score. With elimination, the trust score becomes more stable, and the bad-mouthing attack has no impact on the trust score, as shown in Fig. 5.12b. In Fig.s 5.13 and 5.14, the target node is a malicious node that launch a black-hole and selective-forwarding attacks, respectively. At the same time, a number of watchdog nodes are launching a good-mouthing attack to promote its reputation. In the early rounds, similar behavior to Fig. 5.12 is shown. Using the honesty score and elimination boost the trust system to detect the malicious node as shown in Fig.s 5.13b and 5.14b. The good mouthing attack has no impact on the trust score, as shown in Fig. 5.12b. In Fig.s 5.12a, 5.13a, and 5.14a, the trust score become stable because the honesty score of all the mouthing nodes get low, thus, they are not selected. However, a few jumps happen due to the energy parameter in the selection process. A mouthing node is selected because the total cost $c(A)$ is computed based on the energy c_E and the security c_H costs.

Last, the proposed solution is tested and compared with the existing solution when collaborative attack is applied. In the scenario of Fig.s 5.15 and 5.16, a group of malicious nodes coordinates to launch a selective-forwarding attack (drop rate = 50%). At the same time, these nodes are falsifying the trust scores by launching a good-mouthing attack. Particularly, to deceive the trust model, higher trust scores are sent by the malicious

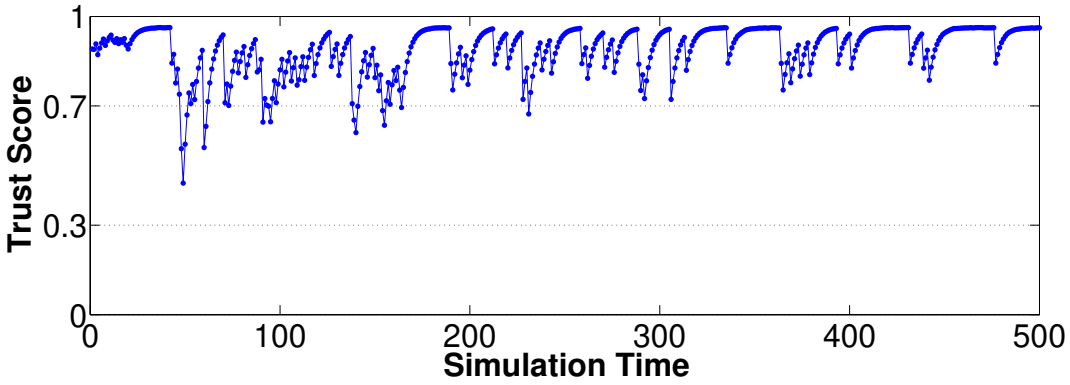


(a) Trust score of a non-malicious node.

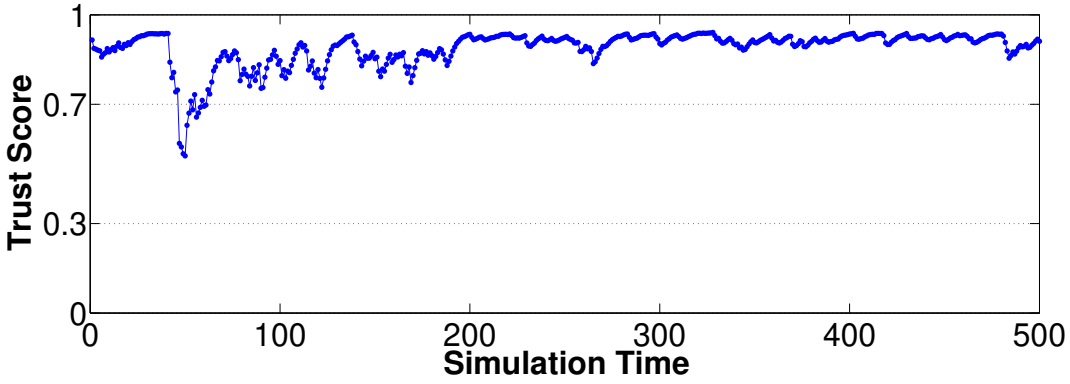


(b) Trust score of a malicious node (drops 50% of messages).

Figure 5.11: Trust score for non-malicious and malicious nodes with the proposed placement and the baseline.

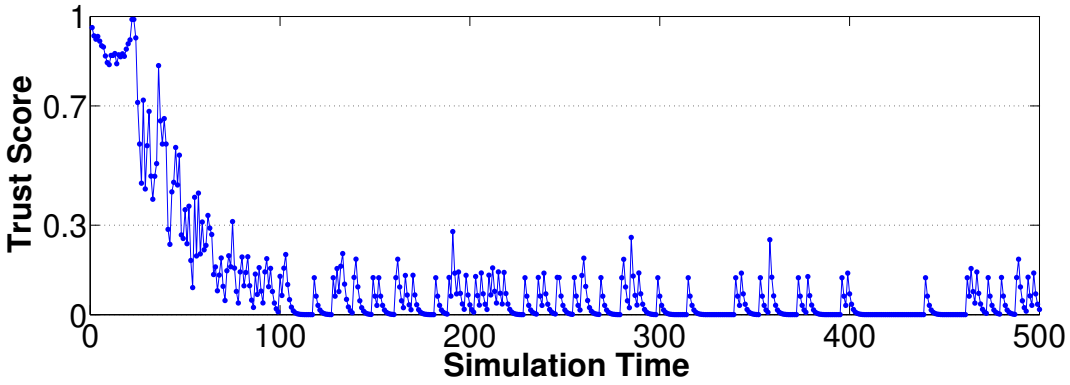


(a) With no elimination of bias nodes.

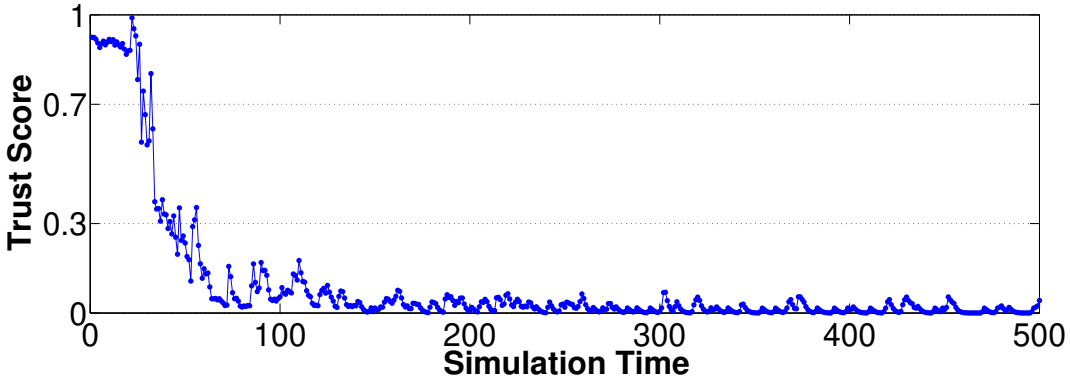


(b) With elimination of bias nodes.

Figure 5.12: Trust score of a non-malicious node with bad-mouthing attack.

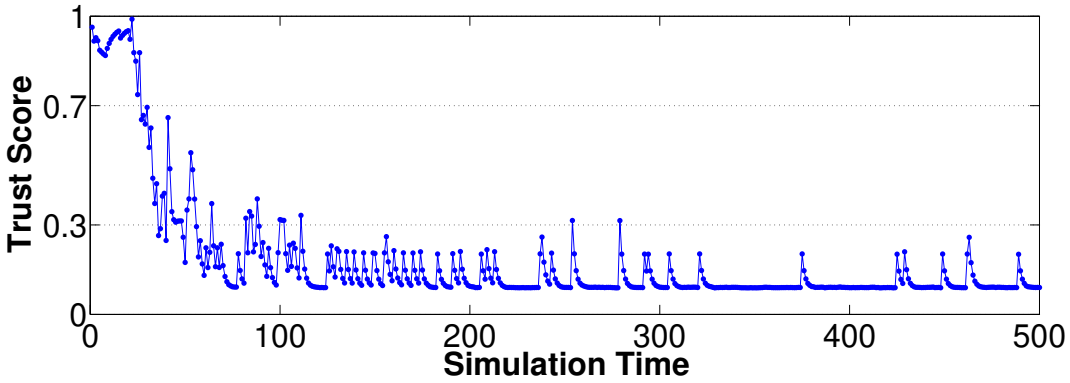


(a) With no elimination of bias nodes.

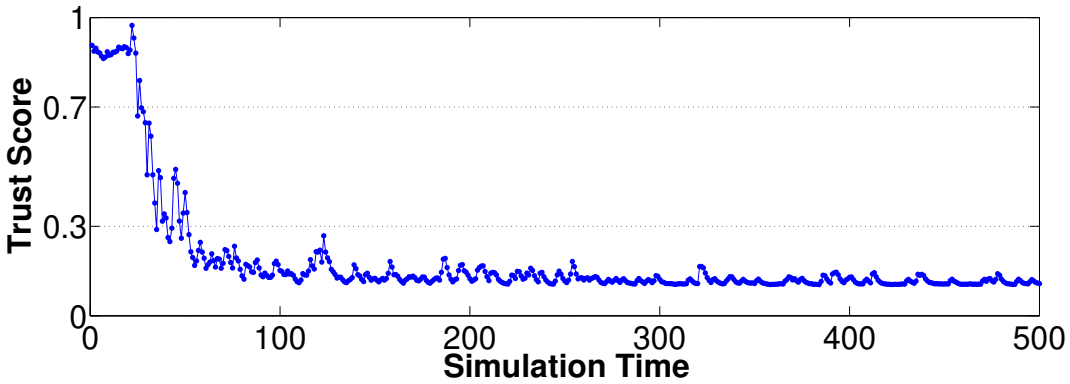


(b) With elimination of bias nodes.

Figure 5.13: Trust score of a malicious node (black-hole attack) with good-mouthing attack.



(a) With no elimination of bias nodes.



(b) With elimination of bias nodes.

Figure 5.14: Trust score of a malicious node (selective-forwarding attack) with good-mouthing attack.

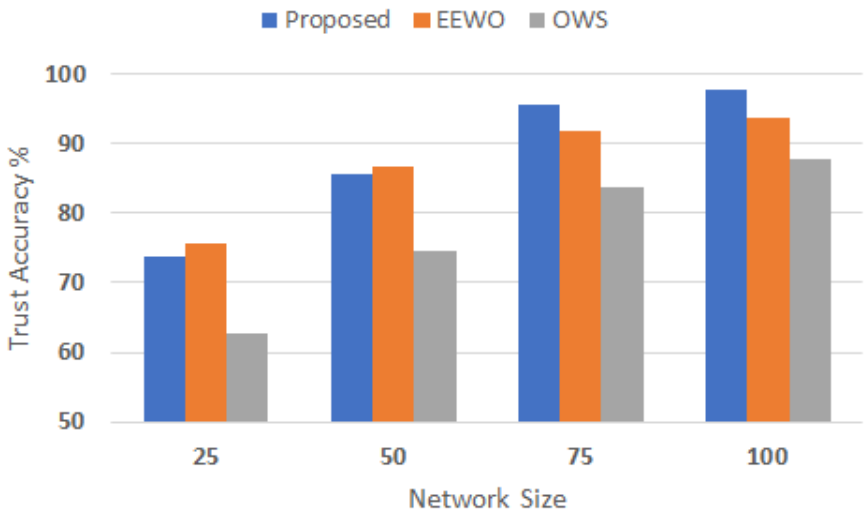


Figure 5.15: Trust accuracy when collaborative attack is applied (selective-forwarding and good-mouthing attacks).

watchdog nodes about the other malicious target nodes. The percentage of malicious nodes is 20% (i.e., the number of malicious nodes is 10 for network size = 50). In Fig. 5.15, the trust accuracy is low for a small network size due to the lack of the coverage degree with the existence of a good-mouthing attack. The proposed solution shows similar trust accuracy performance to EEWO. In a denser network, the proposed solution has better trust accuracy than EEWO due to the increased number of available watchdog nodes. The security selection parameter of EEWO depends on how close the nodes are from the malicious area, while in the proposed solution, the honesty parameter is introduced to lead a more secure selection. OWS shows poor trust accuracy because the selection criteria do not include security or trust-based parameters. On the other hand, the energy saving of OWS is better due to fewer activated watchdog nodes as shown in Fig. 5.16. OWS solution only focuses on the coverage and overlapping parameters when selecting the watchdog nodes. Both the proposed solution and EEWO have similar energy-saving performance.

5.4.3 Discussion

The proposed solution for security function placement shows an excellent performance in terms of trust accuracy and energy saving. However, some challenges still need to be

Figure 5.16: Energy saving when collaborative attack is applied (selective-forwarding and good-mouthing attacks).

addressed and further investigated. First, the networking function deployment in software-defined WSNs is still in a nascent stage. Therefore, the complexity of the watchdog task assignment and replacement requires further investigation. Second, the assignment of the watchdog task lasts for a deterministic number of trust evaluation's time windows. As a result, it is challenging to find the optimal period of watchdog task for the activated nodes. This period should not be very short because this would mean frequent changes in the assigned watchdog nodes for targets, leading to uncertainty regarding the calculated trust for the targets and the honesty score of the watchdog nodes. On the other hand, a more extended period leads to the energy consumption of the assigned watchdog nodes compared to other nodes in the network.

Chapter 6

Conclusions and Future Work

In this chapter, we summarize the main results and contributions of this thesis and present our future research directions.

6.1 Conclusions

In this thesis, we have investigated security countermeasures for software-defined WSNs. Our focus has been on both practical and design considerations for resource constrained networks. We have proposed a hierarchical reputation-based approach to defending the network against communication threats (Problem I), an energy-aware network topology obfuscation to defend against traffic analysis threats (Problem II), and an energy-efficient method for security networking function placement (Problem III). Specifically, in Chapter 3, a hierarchical trust management scheme for software-defined WSNs called TSW was designed to secure SDN-enabled wireless sensor networks. With TSW, trust scores at each level of the software-defined WSN architecture can be computed to allow for swift response against malicious nodes to secure network services. Moreover, TSW considers separate trust scores for the data plane and the control plane, respectively, to detect potential elaborate attacks on either plane. Additionally, using outlier detection and weighted averaging mechanisms, TSW can resist the dishonest behavior. The efficacy of the TSW scheme is demonstrated by simulating and analyzing several communication and trust management threats. In Chapter 4, two mechanisms of network topology obfuscation were proposed to protect WSNs from traffic analysis attacks. In addition, these mechanisms can provide practical and scalable solutions for resource-constrained WSNs. First, a ranking-based route mutation mechanism that considers several route criteria to offer reliable and

energy-efficient routing for route obfuscation is developed. Second, a sink node obfuscation is developed which minimizes the observability of a sink node by an adversary, especially for fully centric WSNs. In Chapter 5, we have studied the watchdog function placement problem. We minimize the total power consumption of watchdog functions by all nodes in the network while considering the security of the selected watchdog nodes. In particular, we introduce the honesty score to determine the node's effectiveness in defending against the mouthing attacks that deceive the trust model by falsifying the trust reporting. The controller avoids selecting malicious nodes for watchdog functions based on this score. We have solved the placement problem using a multi-population genetic algorithm.

6.2 Future Research Directions

For the trust management problem, various trust metrics appeared in the literature have no standard definition. Therefore, standardization of trust metrics is necessary. For the network topology obfuscation problem, advanced adversaries, such as a global adversary or an intelligent heuristic adversary, should be investigated. In addition, a learning-based route mutation approach that utilizes the historical topological data and the current network state needs to be developed. For the security functions placement problem, more design parameters must be investigated, such as collisions that occur due to the overlapping between active watchdog functions. In the following, we provide more details.

6.2.1 Optimized Trust Management Design

More investigations are needed for optimized trust management design. 1) Trust reports may trigger message overhead. Therefore, an optimized message distribution method is needed to decrease the communication overhead. For example, asynchronous trust reports can be used as an alternative approach in which they are triggered whenever there is a change in the trust score and/or if the trust value passes (over/below) a threshold value. 2) In large-scale and dense networks, all the traffic that passes through the network is processed in the trust evaluation phase. As a result, the trust computation becomes a heavy resource-consuming process. Therefore, a lightweight evaluation processing, such as traffic sampling, is needed to be investigated.

6.2.2 AI and Machine Learning

It is timely to integrate modern AI and machine learning techniques to solve problems in traffic engineering and networking. Currently, the direct application of ready-made deep learning and reinforcement learning algorithms is not suitable for many networking domains. As a result, reinforcement learning algorithms need to be adapted to cope with many networking scenarios. For example, networking problems have continuous spaces and involve different data noises. Thus, deep reinforcement learning algorithms with more sophisticated exploration methods should be studied. In addition, trust scores can be used as inputs and/or in reward functions for online learning algorithms. Furthermore, scalability is one of the challenges in the existing learning solutions. As a result, multi-agent and (partially) distributed learning strategies can be introduced.

6.2.3 Other Network Domains

The security measures proposed in this thesis can be investigated for other network domains. First, trust-based solutions can be applied to build a secure network map at the control plane (controllers) for other SDN-based networks, such as wireless access networks. Moreover, social IoT networks are emerging in which social connections play a significant role between the IoT devices and the data owners. Thus, securing social IoT networks using trust-based solutions requires further investigation. On the other hand, moving target defense such as network obfuscation techniques may have a promising impact in improving the security of networks against a wide range of threats. For instance, applying a route mutation scheme in core networks should be studied further under different scenarios and network parameters. Other moving target defense and network obfuscation mechanisms may be applied to software-defined WSNs and other network domains as well.

References

[1] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *Journal of Network and Computer Applications*, vol. 60, pp. 192–219, 2016.

[2] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "Fragmentation-based distributed control system for software-defined wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 901–910, 2018.

[3] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, Q. Zhang, and K.-K. R. Choo, "An energy-efficient SDN controller architecture for IoT networks with blockchain-based security," *IEEE Transactions on Services Computing*, vol. 13, no. 4, pp. 625–638, 2020.

[4] D. Zeng, P. Li, S. Guo, T. Miyazaki, J. Hu, and Y. Xiang, "Energy minimization in multi-task software-defined sensor networks," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3128–3139, 2015.

[5] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 513–521.

[6] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1994–2008, Dec 2017.

[7] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke, "A survey on software-defined wireless sensor networks: Challenges and design requirements." *IEEE Access*, vol. 5, no. 1, pp. 1872–1899, 2017.

[8] K. M. Modieginyane, B. B. Letswamotse, R. Malekian, and A. M. Abu-Mahfouz, "Software defined wireless sensor networks application opportunities for efficient network management: A survey," *Computers & Electrical Engineering*, vol. 66, pp. 274–287, 2018.

[9] F. K. Shaikh and S. Zeadally, "Energy harvesting in wireless sensor networks: A comprehensive review," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 1041 – 1054, 2016.

[10] X. Yang, D. Deng, and M. Liu, "An overview of routing protocols on wireless sensor network," in *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 1. IEEE, 2015, pp. 1000–1003.

[11] L. Chhaya, P. Sharma, G. Bhagwatikar, and A. Kumar, "Wireless sensor network based smart grid communications: cyber attacks, intrusion detection system and topology control," *Electronics*, vol. 6, no. 1, p. 5, 2017.

[12] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 812–837, Firstquarter 2019.

[13] D. He, S. Chan, and M. Guizani, "Securing software defined wireless networks," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 20–25, 2016.

[14] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2015.

[15] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Transactions on Reliability*, vol. 64, no. 3, pp. 1086–1097, 2015.

[16] M. Liyanage, A. B. Abro, M. Ylianttila, and A. Gurtov, "Opportunities and challenges of software-defined mobile networks in network security," *IEEE Security & Privacy*, vol. 14, no. 4, pp. 34–44, 2016.

[17] J. Xie, D. Guo, C. Qian, L. Liu, B. Ren, and H. Chen, "Validation of distributed SDN control plane under uncertain failures," *IEEE/ACM Transactions on Networking*, 2019.

[18] C. Xenofontos, I. Zografopoulos, C. Konstantinou, A. Jolfaei, M. K. Khan, and K.-K. R. Choo, "Consumer, commercial and industrial IoT (in) security: attack taxonomy and case studies," *IEEE Internet of Things Journal*, 2021.

[19] J. R. Ward and M. Younis, "Cross-layer traffic analysis countermeasures against adaptive attackers of wireless sensor networks," *Wireless Networks*, vol. 25, no. 5, pp. 2869–2887, 2019.

[20] J. Jiang, G. Han, H. Wang, and M. Guizani, "A survey on location privacy protection in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 125, pp. 93–114, 2019.

[21] R. C. Alves, D. A. Oliveira, G. C. Pereira, B. C. Albertini, and C. B. Margi, "WS3N: Wireless secure SDN-based communication for sensor networks," *Security and Communication Networks*, vol. 2018, 2018.

[22] F. Ishmanov and Y. Bin Zikria, "Trust mechanisms to secure routing in wireless sensor networks: current state of the research and open research issues," *Journal of Sensors*, vol. 2017, 2017.

[23] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[24] J. Jiang, G. Han, F. Wang, L. Shu, and M. Guizani, "An efficient distributed trust model for wireless sensor networks," *IEEE Transactions on Parallel & Distributed Systems*, no. 1, pp. 1–1, 2015.

[25] W. Meng, "Intrusion detection in the era of IoT: Building trust via traffic filtering and sampling," *Computer*, vol. 51, no. 7, pp. 36–43, 2018.

[26] T. Zhang, L. Yan, and Y. Yang, "Trust evaluation method for clustered wireless sensor networks based on cloud model," *Wireless Networks*, vol. 24, no. 3, pp. 777–797, 2018.

[27] R. Chen, F. Bao, and J. Guo, "Trust-based service management for social internet of things systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, pp. 684–696, 2016.

[28] X. Li, F. Zhou, and J. Du, "LDTS: A lightweight and dependable trust system for clustered wireless sensor networks," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 6, pp. 924–935, 2013.

[29] R. R. Sahoo, A. R. Sardar, M. Singh, S. Ray, and S. K. Sarkar, "A bio inspired and trust based approach for clustering in WSN," *Natural Computing*, vol. 15, no. 3, pp. 423–434, 2016.

[30] V. Vishnu and P. Manjunath, "SeC-SDWSN: Secure cluster-based SDWSN environment for QoS guaranteed routing in three-tier architecture," *International Journal of Communication Systems*, 2019.

[31] R. Wang, Z. Zhang, Z. Zhang, and Z. Jia, "ETMRM: An Energy-efficient Trust Management and Routing Mechanism for SDWSNs," *Computer Networks*, vol. 139, pp. 119–135, 2018.

[32] A. Liu, Z. Zheng, C. Zhang, Z. Chen, and X. Shen, "Secure and energy-efficient disjoint multipath routing for WSNs," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 7, pp. 3255–3265, 2012.

[33] S. Li, Q. Ni, Y. Sun, G. Min, and S. Al-Rubaye, "Energy-efficient resource allocation for industrial cyber-physical IoT systems in 5G era," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2618–2628, 2018.

[34] Y. Zhou, W. Ni, K. Zheng, R. P. Liu, and Y. Yang, "Scalable node-centric route mutation for defense of large-scale software-defined networks," *Security and Communication Networks*, vol. 2017, 2017.

[35] Q. Duan, E. Al-Shaer, and H. Jafarian, "Efficient random route mutation considering flow and network constraints," in *2013 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2013, pp. 260–268.

[36] J. Wang, F. Wang, Z. Cao, F. Lin, and J. Wu, "Sink location privacy protection under direction attack in wireless sensor networks," *Wireless Networks*, vol. 23, no. 2, pp. 579–591, 2017.

[37] A. Rauf, Z. Wang, H. Sajid, and M. Ali Tahir, "Secure route-obfuscation mechanism with information-theoretic security for internet of things," *Sensors*, vol. 20, no. 15, p. 4221, 2020.

[38] N. Baroutis and M. Younis, "Load-conscious maximization of base-station location privacy in wireless sensor networks," *Computer Networks*, vol. 124, pp. 126–139, 2017.

[39] J. Cao, Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang, "The crosspath attack: Disrupting the SDN control channel via shared links," in *28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 19–36.

[40] J. Deng, R. Han, and S. Mishra, "Countermeasures against traffic analysis attacks in wireless sensor networks," in *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*. IEEE, 2005, pp. 113–126.

[41] M. Charfi, A. Mouradian, and V. Vèque, "Networking functions for wireless sensor network applications: an SDN-based approach," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[42] I. Khan, F. Belqasmi, R. Glitho, N. Crespi, M. Morrow, and P. Polakos, "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 553–576, Firstquarter 2016.

[43] G. A. N. Segura, A. Chorti, and C. B. Margi, "Centralized and distributed intrusion detection for resource-constrained wireless SDN networks," *IEEE Internet of Things Journal*, 2021.

[44] M. Bin-Yahya and X. Shen, "HTM: Hierarchical trust management for software-defined WSNs," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[45] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[46] J. Ni, X. Lin, and X. S. Shen, "Toward edge-assisted internet of things: from security and efficiency perspectives," *IEEE Network*, vol. 33, no. 2, pp. 50–57, 2019.

[47] J. Wang, S. Jiang, and A. Fapojuwo, "A protocol layer trust-based intrusion detection scheme for wireless sensor networks," *Sensors*, vol. 17, no. 6, p. 1227, 2017.

[48] N. Zhang, N. Lu, R. Lu, J. W. Mark, and X. Shen, "Energy-efficient and trust-aware cooperation in cognitive radio networks," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 1763–1767.

[49] F. Ishmanov, A. S. Malik, S. W. Kim, and B. Begalov, "Trust management system in wireless sensor networks: design considerations and research challenges," *Transactions on Emerging Telecommunications Technologies*, vol. 26, no. 2, pp. 107–130, 2015.

[50] P. Zhou, S. Jiang, A. Irissappane, J. Zhang, J. Zhou, and J. C. M. Teo, "Toward energy-efficient trust system through watchdog optimization for WSNs," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 613–625, 2015.

[51] I. Souissi, N. B. Azzouna, and L. B. Said, "A multi-level study of information trust models in WSN-assisted IoT," *Computer Networks*, 2019.

[52] M. Jacobsson and C. Orfanidis, "Using software-defined networking principles for wireless sensor networks," in *Proc. 11th Swedish National Computer Networking Workshop*, 2015.

[53] T. Luo, H.-P. Tan, and T. Q. Quek, "Sensor OpenFlow: Enabling software-defined wireless sensor networks," *IEEE Communications Letters*, vol. 16, no. 11, pp. 1896–1899, 2012.

[54] S. Bera, S. Misra, S. K. Roy, and M. S. Obaidat, "Soft-WSN: Software-defined WSN management system for IoT applications," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2074–2081, 2016.

[55] W. Xiang, N. Wang, and Y. Zhou, "An energy-efficient routing algorithm for software-defined wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7393–7400, 2016.

[56] S. Manisekaran and R. Venkatesan, "An analysis of software-defined routing approach for wireless sensor networks," *Computers & Electrical Engineering*, vol. 56, pp. 456–467, 2016.

[57] A. De Gante, M. Aslan, and A. Matrawy, "Smart wireless sensor network management based on software-defined networking," in *Communications (QBSC), 2014 27th Biennial Symposium on*. IEEE, 2014, pp. 71–75.

[58] S. W. Pritchard, G. P. Hancke, and A. M. Abu-Mahfouz, "Security in software-defined wireless sensor networks: Threats, challenges and potential solutions," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 168–173.

[59] J. Esch, "Prolog to, "software-defined networking: a comprehensive survey"," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 10–13, Jan 2015.

[60] I. Farris, T. Taleb, Y. Khettab, and J. S. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Communications Surveys & Tutorials*, 2018.

[61] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[62] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and control element separation (ForCES) protocol specification," RFC 3746, Tech. Rep., 2010.

[63] J. Puente Fernández, L. García Villalba, and T.-H. Kim, "Software defined networks in wireless sensor architectures," *Entropy*, vol. 20, no. 4, p. 225, 2018.

[64] A.-C. G. Anadiotis, L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "Towards a software-defined network operating system for the IoT," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 579–584.

[65] H. Mostafaei and M. Menth, "Software-defined wireless sensor networks: A survey," *Journal of Network and Computer Applications*, 2018.

[66] K. Govindan and P. Mohapatra, "Trust computations and trust dynamics in mobile adhoc networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 279–298, 2011.

[67] H. Alzaid, M. Alfaraj, S. Ries, A. Jøsang, M. Albabtain, and A. Abuhaimed, "Reputation-based trust systems for wireless sensor networks: A comprehensive review," in *IFIP International Conference on Trust Management*. Springer, 2013, pp. 66–82.

[68] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020.

[69] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483–2495, 2018.

[70] V. B. Reddy, S. Venkataraman, and A. Negi, "Communication and data trust for wireless sensor networks using D-S theory," *IEEE Sensors Journal*, vol. 17, no. 12, pp. 3921–3929, 2017.

[71] R. J. Cai, X. J. Li, and P. H. J. Chong, "An evolutionary self-cooperative trust scheme against routing disruptions in MANETs," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 42–55, 2018.

[72] S. Talbi, M. Koudil, A. Bouabdallah, and K. Benatchba, "Adaptive data-communication trust mechanism for clustered wireless sensor networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[73] W. Meng, W. Li, C. Su, J. Zhou, and R. Lu, "Enhancing trust management for wireless intrusion detection via traffic sampling in the era of big data," *IEEE Access*, vol. 6, pp. 7234–7243, 2018.

[74] J. Chen, Z. Tian, X. Cui, L. Yin, and X. Wang, "Trust architecture and reputation evaluation for internet of things," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–9, 2018.

[75] A. B. Usman and J. Gutierrez, "Toward trust based protocols in a pervasive and mobile computing environment: A survey," *Ad Hoc Networks*, vol. 81, pp. 143–159, 2018.

[76] T. Gaber, S. Abdelwahab, M. Elhoseny, and A. E. Hassanien, "Trust-based secure clustering in WSN-based intelligent transportation systems," *Computer Networks*, vol. 146, pp. 151–158, 2018.

[77] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3527–3537, 2019.

[78] M. M. Hasan and H. T. Mouftah, "Optimization of watchdog selection in wireless sensor networks," *IEEE Wireless Communications Letters*, vol. 6, no. 1, pp. 94–97, 2016.

[79] Y. Shi, H. Zhang, J. Wang, F. Xiao, J. Huang, D. Zha, H. Hu, F. Yan, and B. Zhao, "Chaos: An SDN-based moving target defense system," *Security and Communication Networks*, vol. 2017, 2017.

[80] N. Saputro, S. Tonyali, A. Aydeger, K. Akkaya, M. A. Rahman, and S. Uluagac, "A review of moving target defense mechanisms for internet of things applications," *Modeling and Design of Secure Internet of Things*, pp. 563–614, 2020.

[81] M. Ge, J. B. Hong, S. E. Yusuf, and D. S. Kim, "Proactive defense mechanisms for the software-defined internet of things with non-patchable vulnerabilities," *Future Generation Computer Systems*, vol. 78, pp. 568–582, 2018.

[82] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Attack graph-based moving target defense in software-defined networks," *IEEE Transactions on Network and Service Management*, 2020.

[83] J. Y. Koh, D. Leong, G. W. Peters, I. Nevat, and W.-C. Wong, "Optimal privacy-preserving probabilistic routing for wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 9, pp. 2105–2114, 2017.

[84] J. Y. Koh, G. W. Peters, I. Nevat, and D. Leong, "Probabilistic routing in wireless networks with privacy guarantees," *Computer Communications*, vol. 151, pp. 228–237, 2020.

[85] M. M. Mahmoud and X. Shen, "A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1805–1818, 2011.

[86] B. Di Ying, D. Makrakis, and H. T. Mouftah, "Anti-traffic analysis attack for location privacy in WSNs," *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, no. 1, pp. 1–15, 2014.

[87] A. Liu, X. Liu, T. Wei, L. T. Yang, S. Rho, and A. Paul, "Distributed multi-representative re-fusion approach for heterogeneous sensing data collection," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 3, pp. 1–25, 2017.

[88] M. M. Mahmoud, X. Lin, and X. Shen, "Secure and reliable routing protocols for heterogeneous multihop wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1140–1153, 2013.

[89] G. Chai, M. Xu, W. Xu, and Z. Lin, "Enhancing sink-location privacy in wireless sensor networks through k-anonymity," *International Journal of Distributed Sensor Networks*, vol. 8, no. 4, p. 648058, 2012.

[90] R. E. Navas, F. Cuppens, N. B. Cuppens, L. Toutain, and G. Z. Papadopoulos, "MTD, where art thou? a systematic review of moving target defense techniques for IoT," *IEEE Internet of Things Journal*, 2020.

[91] Y. Wang, Q. Chen, J. Yi, and J. Guo, "U-TRI: Unlinkability through random identifier for SDN network," in *Proceedings of the 2017 Workshop on Moving Target Defense*, 2017, pp. 3–15.

[92] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: A comprehensive survey," *Security and Communication Networks*, vol. 2017, 2017.

[93] R. A. Shaikh, H. Jameel, B. J. d'Auriol, H. Lee, S. Lee, and Y.-J. Song, "Group-based trust management scheme for clustered wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 11, pp. 1698–1712, 2009.

[94] Y. L. Sun, W. Yu, Z. Han, and K. R. Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 305–317, 2006.

[95] A. Leal, J. F. Botero, and E. Jacob, "Improving early attack detection in networks with sFlow and SDN," in *Workshop on Engineering Applications*. Springer, 2018, pp. 323–335.

[96] Y. Kou, C.-T. Lu, and D. Chen, "Spatial weighted outlier detection," in *Proceedings of the 2006 SIAM International Conference on Data Mining*. SIAM, 2006, pp. 614–618.

[97] F. Bao, R. Chen, M. Chang, and J.-H. Cho, "Hierarchical trust management for wireless sensor networks and its applications to trust-based routing and intrusion detection," *IEEE Transactions on Network and Service Management*, vol. 9, no. 2, pp. 169–183, 2012.

[98] B. Sun and D. Li, "A comprehensive trust-aware routing protocol with multi-attributes for WSNs," *IEEE Access*, vol. 6, pp. 4725–4741, 2018.

[99] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[100] R. Sawilla and D. Skillicorn, "Partial cuts in attack graphs for cost effective network defence," in *2012 IEEE Conference on Technologies for Homeland Security (HST)*. IEEE, 2012, pp. 291–297.

[101] M. Bin-Yahya, O. Alhussein, and X. Shen, "Securing software-defined WSNs communication via trust management," *IEEE Internet of Things Journal*, pp. 1–15, 2021.

[102] M. Conti, F. De Gaspari, and L. V. Mancini, "A novel stealthy attack to gather SDN configuration-information," *IEEE Transactions on Emerging Topics in Computing*, 2018.

[103] Q.-S. Hua, M. Ai, H. Jin, D. Yu, and X. Shi, "Distributively computing random walk betweenness centrality in linear time," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 764–774.

[104] M. Bin-Yahya and X. Shen, "Secure and energy-efficient network topology obfuscation for software-defined WSNs," *IEEE Internet of Things Journal*, 2022.

[105] A. E. Eiben, J. E. Smith *et al.*, *Introduction to Evolutionary Computing*. Springer, 2003, vol. 53.

[106] U. Baroudi, M. Bin-Yahya, M. Alshammari, and U. Yaqoub, "Ticket-based QoS routing optimization using genetic algorithm for WSN applications in smart grid," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 4, pp. 1325–1338, 2019.

[107] Y. Xie, F.-X. Gui, W.-J. Wang, and C.-F. Chien, "A two-stage multi-population genetic algorithm with heuristics for workflow scheduling in heterogeneous distributed computing environments," *IEEE Transactions on Cloud Computing*, 2021.

[108] S.-C. Liu, Z.-G. Chen, Z.-H. Zhan, S.-W. Jeon, S. Kwong, and J. Zhang, "Many-objective job-shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach," *IEEE Transactions on Cybernetics*, 2021.

[109] X. Shi, W. Long, Y. Li, and D. Deng, "Multi-population genetic algorithm with ER network for solving flexible job shop scheduling problems," *PloS one*, vol. 15, no. 5, 2020.

[110] A. Król and G. Sierpiński, "Application of a genetic algorithm with a fuzzy objective function for optimized siting of electric vehicle charging devices in urban road networks," *IEEE Transactions on Intelligent Transportation Systems*, 2021.