School of Electrical and Computer Engineering

Detecting Wormhole and Byzantine Attacks in Mobile ad hoc Networks

Mohammad Rafiqul Alam

This thesis is presented for the Degree of

Master of Philosophy

of

Curtin University of Technology

May 2011

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature: ………………………………………….

Date: ………………………...

To my parents, Kalimullah and Rokeya Begum

Acknowledgements

I would like to gratefully acknowledge the enormous assistance of my supervisor, Dr. King-Sun Chan. His guidance was exceptional throughout my studies towards the Master of Philosophy in Electrical and Computer Engineering degree. In addition to being a supervisor and teacher, Dr. King-Sun Chan has been a colleague and friend.

I would also like to acknowledge the financial support provided to me by the Department of Electrical and Computer Engineering, Curtin University of Technology. My studies would not be possible without the tuition waiver scholarship that I received from them. Moreover, I would also like to thank Dr. Syed Isalm, who helped me a number of occasions with practical advice on various issues.

My sincere thanks go to the following Ph.D. students: Nadya, Ruhul, Paul, Remmy, Kevin, Stephen and Susanne. We together participated in a number of outdoor events, which was not only very enjoyable but also helped me to overcome a lot of struggling moments.

I greatly thank my parents for their continuous mental support and encouragement throughout my studies. My thanks also go to my brother, Dr. Muhammad S. Alam, who despite having physical problems guided my living in Perth in numerous occasions.

Abstract

The recent advancements in the wireless technology and their wide-spread utilization have made tremendous enhancements in productivity in the corporate and industrial sectors. However, these recent progresses have also introduced new security vulnerabilities. Since the wireless shared medium is completely exposed to outsiders, it is susceptible to attacks that could target any of the OSI layers in the network stack. For example, jamming of the physical layer, disruption of the medium access control (MAC) layer coordination packets, attacks against the routing infrastructure, targeted attacks on the transport protocol, or even attacks intended to disrupt specific applications. Unfortunately, the effects of applying the security techniques used in wired networks, such as access control and authentication, to wireless and mobile networks have been unsatisfactory due the unique features of such networks. As a result, achieving security goals for mobile ad hoc networks (MANET) has gained significant attention in recent years. Many critical applications of MANET, such as emergency rescue operations, military tactical communication, and business operations like mining and oil drilling platforms, require a friendly and cooperative environment.

The aim of this study is to design detection mechanisms for traditional wormhole and Byzantine wormhole attacks by using the topological comparison and round trip time (RTT) measurements. The first step for detecting traditional wormhole attack is that an initiator of the detection process populates its one-hop neighbor list, and also calculates the average round trip time (RTT_{avg}). Meanwhile, a list of suspected neighbors is generated on the basis of RTT_{avg} and RTT. Then, topological information is exchanged between the initiator and all the suspected neighbors to detect the presence of a wormhole link.

In this thesis, we also focus on detecting Byzantine wormhole attack in MANET. In the case of detecting such attacks, the initiator creates its one-hop neighbor list and calculates the average RTT_{avg}. The initiator also generates a suspected list of its three hop neighbors. In the next phase, the initiator exchanges topological information with all the one hop neighbors to detect the presence of any Byzantine wormhole tunnel. One of the major concerns for the topological comparison based approach is to give the initially suspected nodes a second chance to prove their reliability by exchanging topological information.

We have implemented the detection algorithms in ad hoc on demand distance vector (AODV) and optimized link state routing (OLSR) routing protocols. Then, performance evaluation of the proposed detection mechanisms is conducted. We also compared our proposed detection methods with some of the existing detection methods by simulation. The results show that our schemes can achieve better detection performance.

Table of Contents

1. Introduction **1**

1.1 Background 1

1.1.1 Applications of MANET 3

1.1.2 Routing Protocols in MANET 4

1.1.3 Security Attacks in MANET 10

1.2 Context of Research 21

1.3 Summary of Contributions 22

1.4 Works from This Thesis Published by Author 24

1.5 Organization of the Thesis 25

2. Current State of the Detection of Wormhole Attacks **26**

2.1 Combating Traditional Wormhole Attack 26

2.1.1 One Hop Delay Based Approaches 26

2.1.2 Methods Based on Graph Analyses 31

2.1.3 Special Hardware/Middleware Based Methods 33

2.2 Combating Byzantine Wormhole Attack 35

2.2.1 On-Demand Secure Byzantine Resilient Routing (ODBSR) 36

2.2.2 Secure Routing Against Collusion 40

2.3 Objectives of This Study 43

3. Detection of Traditional Wormhole Attacks Using RTT and Topological Comparison **45**

3.1 Round Trip Time (RTT) Measurement 45

3.1.1 RTT Measurement in TCP 45

3.1.2 Our Proposed RTT Measurement Method 46

3.2 Topological View of a Network 51

3.2.1 Topology of a Network 51

3.2.2 Effects of Wormhole on Topology 52

3.3 Detection of Traditional Wormhole Attacks in MANET 54

3.3.1 Neighbor List 55

3.3.2 Topological Comparison Algorithm 56

3.3.3 System Reactions When a Wormhole Attack is Detected........... 58
3.3.4 Security of the Detection Method... 58
3.4 Performance Evaluation Using ns-2 Simulator................................. 59
3.4.1 Network Simulator (ns-2).. 59
3.4.2 Generating Topologies in ns-2 59
3.4.3 Simulating Traditional Wormhole Attack.................................. 60
3.4.4 Performance Metrics 61
3.5 Wormhole Detection in AODV Routing Protocol 62
3.6 Wormhole Detection in OLSR Routing Protocol 66
3.7 Message Overhead and Detection Time.. 68
3.8 Summary 70

4. Detecting Byzantine Wormhole Attack Using Topological Comparison and RTT.. **71**

4.1 Effects of Byzantine Wormhole on Topology 71
4.2 Detection of Byzantine Wormhole Attacks in MANET 74
4.2.1 One Hop Neighbor List 75
4.2.2 Three Hop Neighbor List... 75
4.2.3 Topological Comparison Algorithm.. 77
4.3 Simulating Byzantine Wormhole 79
4.4 Byzantine Wormhole Detection in AODV Routing Protocol 80
4.5 Security of the Detection Method 83
4.6 Message Overhead 84
4.7 Summary 84

5. Conclusion and Future Work **85**

5.1 A Review of the Research Contributions 85
5.2 Future Works... 86

Appendix 1...87

Appendix 2...92

Bibliography...95

List of Figures

Figure 1.1 A mobile ad hoc network..2

Figure 1.2 Propagation of RREQ packets..5

Figure 1.3 Propagation of RREP packets...5

Figure 1.4 OLSR routing 7

Figure 1.5 Zone routing protocol 9

Figure 1.6 Classification of security attacks in MANET............................10

Figure 1.7 Blackhole attack in AODV routing 14

Figure 1.8 Link spoofing attack 14

Figure 1.9 Path of RREQ packets (S→D) in presence of wormhole tunnel 17

Figure 1.10 Path of RREP packets (D→S) in presence of wormhole tunnel 17

Figure 1.11 Two-hop Byzantine wormhole tunnel 20

Figure 2.1 HELLO$_{rep}$ aggregation 28

Figure 2.2 Timing diagram of TTM method..29

Figure 2.3 It is posssible to pack at most two nodes inside a Lune with inter-distance more than one..32

Figure 2.4 Timing diagram of NEVO (no wormhole)...............................34

Figure 3.1 HELLO broadcast..47

Figure 3.2 HELLO$_{rep}$ unicast 47

Figure 3.3 Measuring RTT at the MAC layer...50

Figure 3.4 Topology of 53 nodes as viewed on NAM animator.................51

Figure 3.5 Two-hop wormhole attack scenario 53

Figure 3.6 Actual *one hop* topology of node *s* and node *d* 53

Figure 3.7 Distorted topological view in presence of wormhole................53

Figure 3.8 Complete neighbor list of s...56

Figure 3.9 Complete neighbor list of d 56

Figure 3.10 Tunnel length vs. Wormhole detection rate.............................63

Figure 3.11 Tunnel length vs. Accuracy of alarms 64

Figure 3.12 Comparison with existing detection methods (30 nodes)........65

Figure 3.13 Wormhole length vs. Detection rate..67
Figure 3.14 Wormhole length vs. Accuracy of alarm.................................68
Figure 3.15 Tunnel Length vs. Detection Time...69
Figure 4.1 A two-hop Byzantine wormhole attack scenario.......................71
Figure 4.2 One hop neighbors of S 72
Figure 4.3 Three hop neighbors of S...72
Figure 4.4 Three hop neighbors of node a 72
Figure 4.5 Three hop neighbors of node b 72
Figure 4.6 Three hop neighbors of node c 72
Figure 4.7 Three hop neighbors of node x 72
Figure 4.8 One hop neighbor list of s..75
Figure 4.9 Three hop neighbors list of s 76
Figure 4.10 Tunnel length vs. Detection rate..82
Figure 4.11 Tunnel length vs Detection accuracy.......................................82
Figure A.1 Overlapping area of two nodes 92
Figure A.2 Probability of not having a common neighbor between two nodes when R=250m 94

List of Algorithms

Algorithm 1 Broadcast of HELLO to explore *three hop* neighbors ……...76

Algorithm 2 Byzantine wormhole detection ….. 78

Table of Acronyms

s	Sender of ENQ packets
r	Receiver of ENQ packet
$TRST_s$	$TRST$ list of s
SUS_s	SUS list of s
$TRST_r$	$TRST$ list of r
me	TRUE/FALSE, denotes whether s is in $TRST_r$
$suspect$	Number of nodes in $TRST_r \cap SUS_s$
$trust$	Number of nodes in $TRST_r \cap TRST_s$

1. Introduction

In recent years, achieving security goals for mobile ad hoc networks (MANET) has gained significant attention due to the challenges implied by their unpredictable characteristics. Although military tactical communication was considered as the primary application of MANET, commercial applications of this type of networks are increasing significantly. Nevertheless, the commercial success of this type of network depends on people's confidence in its security. Wormhole attack is one of the most advanced forms of security threats in MANET. In this research, two varieties of this type of attack (traditional and Byzantine wormhole) are studied. Then, their impact on the network topology is observed to design a topological comparison based detection scheme. The background study for this research project is presented in this chapter.

1.1 Background

A mobile ad hoc network (MANET) is a collection of wireless devices which can dynamically be set up without using any pre-existing infrastructure or central controller. In a MANET, nodes within each other's transmission range can communicate directly, whereas nodes outside each other's transmission range rely on other nodes to relay messages [1]. Hence, a multi-hop scenario is created where every node functions as a router. The key features of MANETs are presented below:

a) *Autonomous terminals*: In a MANET, each node is an autonomous terminal, which functions as both a host and router. In other words, besides having the basic processing abilities of a host, the nodes in a MANET can also perform switching functionalities as routers. So, the endpoints and switches are indistinguishable in MANET [2].

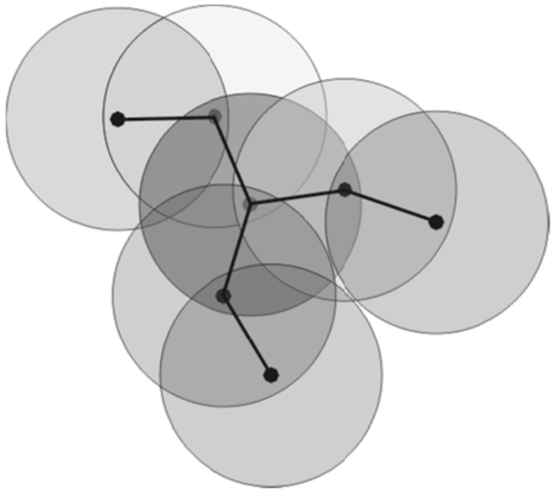


Figure 1.1 A mobile ad hoc network

b) *Dynamic configuration*: As the participating nodes may join or leave the network without any disruption to users, the topology in a MANET may alter rapidly and unpredictably. In addition, the network may consist of both bidirectional and unidirectional links [3]. The wireless channel is subject to interferences and errors, exhibiting volatile characteristics in terms of bandwidth and delay. A typical MANET topology is shown in Fig. 1.1.

c) *Distributed control:* In a MANET, there is no fixed infrastructure or central controller. The control and management of the network is distributed among the participating nodes. So, the participating nodes in a MANET cooperate with each other and each node acts as relay as needed to implement some core functions, such as security and routing [2].

d) *Low profile terminals*: The mobile nodes in a MANET have limited processing ability, small physical memory and limited power storage. Therefore, the devices require optimized algorithms and energy conservation. The attackers can easily breach the network security by sneaking through these low-end devices.

e) *Bandwidth-constrained:* The wireless links have significantly lower capacity than their hardwired counterparts. One effect of relatively low capacity is congestion, which is responsible for dropping the throughput of the network.

f) *Limited physical security:* Since the wireless medium is open for all, mobile ad hoc networks are more vulnerable to physical security threats than are wired networks. The boundary that separates the inside network from the outside world becomes blurred [4].

1.1.1 Applications of MANET

The wireless communication technology has been deployed in military since 1970s. This allows the military to maintain an information network between the soldiers, vehicles, and military information headquarters [2]. In fact, the preliminaries of mobile ad hoc networks came from this field. Besides military communication, MANETs can also be deployed in scenarios where the pre-existing infrastructure has been damaged due to natural calamities (e.g., earthquake, tsunami, bushfire etc.), or human interventions (e.g., terror attack, theft etc.). Another application of MANET would be in the mining industry, where the workers deep underneath the ground level would be able to communicate with the base station. In addition, MANETs are also suitable to be used in a university campus, where the participants in seminar, or students performing an experiment at different corners would be able to share each other's views. In general, ad hoc networks can be deployed anywhere where there is little or no communication infrastructure or the existing infrastructure is expensive or inconvenient to use.

1.1.2 Routing Protocols in MANET

A sender in an ad hoc network may not always be able to pass its packets directly to the intended receiver. So, routing mechanisms are required whenever an intended receiver is outside the transmission range of the sender [5]. The goal of the routing protocol is to discover the latest topology. The routing protocols in MANET can be classified into three categories:

a) *Proactive routing protocols*: In this family of routing protocol, all nodes exchange routing information periodically or whenever the topology changes. Since each node maintains a consistent view of the network, a route to the destination (if it can be reached) is always available. Examples of proactive routing protocols include: Destination-Sequenced Distance-Vector (DSDV) [6] or Optimized Lint State Routing (OLSR) [7].

b) *Reactive routing protocols*: In reactive routing, the route discovery process is initiated by a sender whenever it wants to send packets to a destination. The route is maintained until the destination becomes unreachable or is not needed anymore. Examples are: Ad hoc on-demand Distance Vector (AODV) [8], Dynamic Source Routing (DSR) [9], and Temporally Ordered Routing Algorithm (TORA) [10].

c) *Hybrid routing protocols*: The characteristics of proactive and reactive routing protocols are combined to avoid the shortcomings of the two families and to retain most of their benefits. Examples of hybrid routing protocols include: Zone Routing Protocol (ZRP) [11], and Wireless Adaptive Routing Protocol (WARP) [12].

In the following sections, we present illustration of two of the most popular routing protocols in ad hoc networking: ad hoc on-demand distance vector (AODV) and optimized link state routing (OLSR). In this research, our focus is on designing detection mechanisms for two variations of wormhole attacks in AODV and OLSR routing. We also present brief descriptions on other routing protocols (e.g., DSR, DSDV, and ZRP).

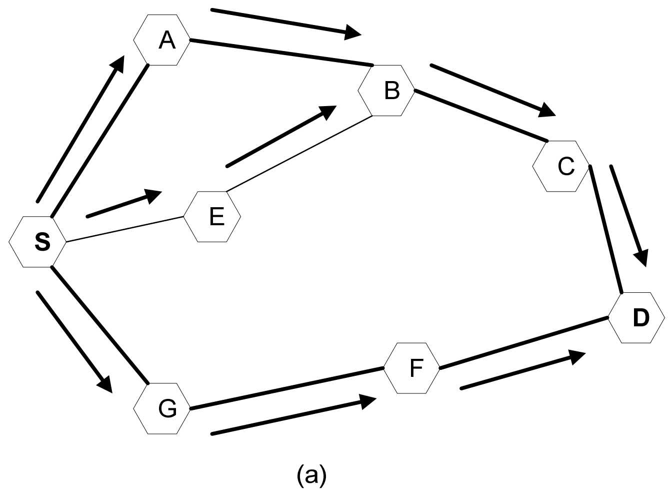


Figure 1.2 Propagation of RREQ packets

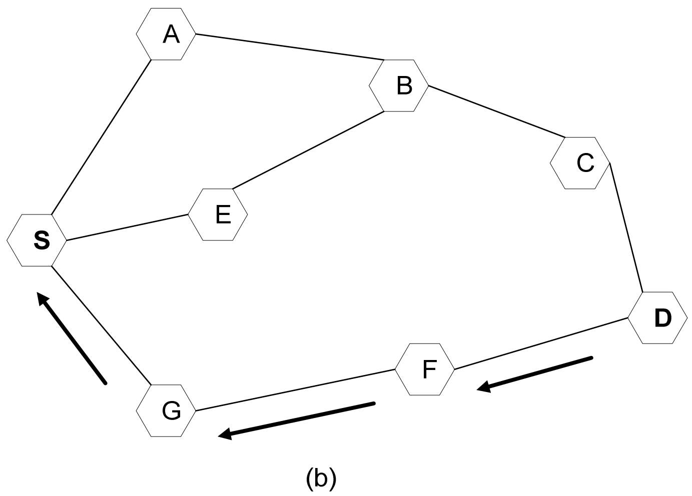


Figure 1.3 Propagation of RREP packets

1.1.2.1 Ad Hoc On-Demand Distance Vector (AODV)

AODV [8] is a reactive routing protocol developed for MANET which uses traditional routing table with one entry per destination. In this routing protocol, routes are established dynamically at intermediate nodes. Each node maintains sequence numbers to determine freshness of routing information and avoid routing loops. Another important feature is the maintenance of timer-based state, which is required to decide whether a routing table entry is expired or not.

The route discovery process in AODV starts with the broadcast of route request (RREQ) packets by a source (S), who wants to send a packet to a destination (D) for which it does not have any route information. A recipient of RREQ first checks the *sender ID* and *broadcast ID* included in the RREQ packet to make sure whether it has already received the same RREQ. If not, it stores the *sender ID* as a reference for reverse path, increments the *hop count* field, and rebroadcasts the RREQ in its vicinity. This process is continued until a route to the destination (D) is found. The propagation of RREQ and RREP packets in AODV routing is shown in Fig. 1.2 and Fig. 1.3 respectively. As shown in the figures, the broadcast RREQ from the source (S) is received by nodes A, E and G. Upon receiving the first RREQ, the destination (D) replies with RREP back to the source (S). In the given scenario, there are three ways to reach the destination (D) from the source (S): S-G-F-D, S-A-B-C-D, and S-E-B-C-D. Since S-G-F-D is the shortest path (3 hops), D first receives RREQ through this path. The RREP packet follows the reverse path where the RREQ arrived. So, RREP from D will reach node S through the path D-F-G-A, which is also selected as the route for exchanging packets between them.

1.1.2.2 Optimized Link State Routing

OLSR [7] is a proactive routing protocol based on the traditional link-state algorithm where each node maintains the topology information about the network. In OLSR routing, each node periodically exchanges link-state messages, such as HELLO and Topology Control (TC). Moreover, a multipoint relaying (MPR) strategy is used to minimize the size of the control messages and the number of rebroadcasting nodes. Each node in the network selects a set of nodes, which is known as multipoint relays, to retransmit its packets. Any node exclusive of the MPR set can read the packets but cannot retransmit them [13].

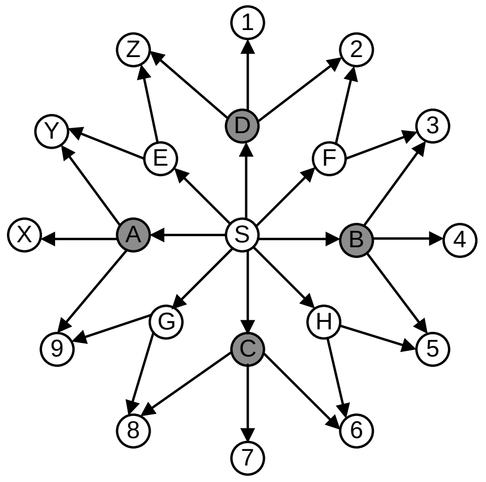


Figure 1.4 OLSR routing

Each node periodically broadcasts HELLO messages to find its *one hop* neighbors and *two hop* neighbors. Consequently, the sender selects a subset of *one hop* neighbors, known as MPR nodes, which covers all of its *two hop* neighbors. On the other hand, each node maintains another set of nodes, known as MPR selector, which includes the nodes that have selected it as an MPR node. For example, in Fig. 1.4, node S broadcasts HELLO messages and then selects nodes A, B, C and D to be its MPR nodes. This is because that the nodes A, B, C and D cover all the nodes, which are *two hops* away from the source node S. In OLSR routing protocol, TC messages through the MPR nodes are used to disseminate the topological information throughout the network. Hence, each can determine an optimum route to every destination by using topological information and the route to an intended destination is known when the data is transferred.

1.1.2.3 Other Routing Protocols

In this research project, our focus is on AODV and OLSR for the evaluation of our proposed detection methods. However, there are some other routing protocols (reactive/proactive/hybrid) also used in ad hoc networking. In this section short description of DSDV, DSR and ZRP is provided.

Destination-Sequenced Distance Vector (DSDV)

The DSDV [6] routing protocol is based on the Bellman-Ford [14] routing algorithm and it guarantees loop free routes. In this routing protocol, a table of all available destinations is maintained by each node. The number of hops to reach a destination and the sequence number for a destination are also included in the routing table. In order to reduce the overhead transmitted through the network, two packet types, “full dump”, and “incremental” are used in DSDV. The full dump packet carries all the available routing information and the incremental packet carries only the information changed since the last full dump [13].

Dynamic Source Routing (DSR)

In DSR [9], the source knows the complete hop-by-hop route to the destination and the route is stored in a route cache. When a node wants to send packets to another node for which it does not have the routing information, it starts a route discovery process by broadcasting route requests (RREQ) to its neighbors. Each node receiving an RREQ appends its identification and then rebroadcasts, or replies with a route reply if it is the destination or it has information about the destination. The RREP follows the reverse route taken by the RREQ packet. An advantage of DSR routing protocol is that multiple routes to a destination can be stored. Hence, the source can check its route cache for a valid route and if a valid route to the destination is found, there is no need for route discovery. Moreover, DSR routing protocol does not require periodic beaconing. As a result, nodes can enter sleep mode to conserve their power and it also saves bandwidth in the network [13].

Zone routing protocol (ZRP)

Reactive routing protocols involve long route request delays, whereas proactive routing protocols uses excess bandwidth to maintain routing information. However, the zone routing protocol (ZRP) can be classified as a hybrid (reactive/proactive) routing protocol. ZRP is based on the notion of *routing zone*, defined for each node in the network. A *routing zone* includes all nodes whose distance is at most a predefined *zone radius (ρ)*, expressed in hops. Therefore, all the routes in the network can be expressed as a sequence of nodes separated by approximately the *zone radius (ρ)* [11].

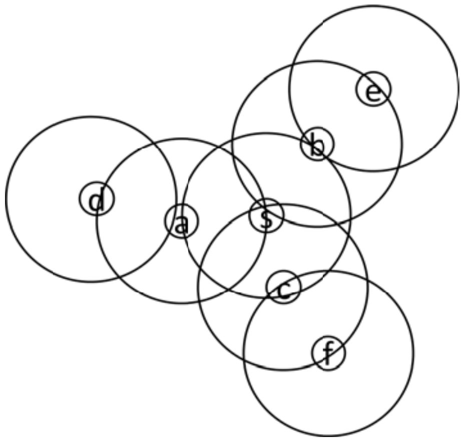


Figure 1.5 Zone routing protocol

An example of ZRP is shown in Fig. 1.5, where a source (*s*) wants to send data to a destination (*d*). In ZRP, a concept called *broadercasting* is used instead of broadcasting. According to *broadercasting* mechanism, node *s* verifies that *d* is not within its *routing zone* and, therefore, sends a query to the border of its own *routing zone.* As in Fig. 1.5 $\rho = 1$, nodes *a, b* and *c* receive query from node *s broadercast* the query. This process is repeated until the destination *d* is reached.

1.1.3 Security Attacks in MANET

Mobile ad hoc networks are vulnerable to a number security attacks due to their characteristics and nature. But the mobile devices participating in a MANET have limited resources and physical protection. Therefore, the use of strong cryptographic tools, tokens and smart cards pose huge challenge to the limited physical resources of the participating devices. The security threats on MANET can be classified by two criteria: mode of attack (active or passive) and origin of attack (internal or external). A classification of the security attacks in MANET is presented in Fig. 1.6.

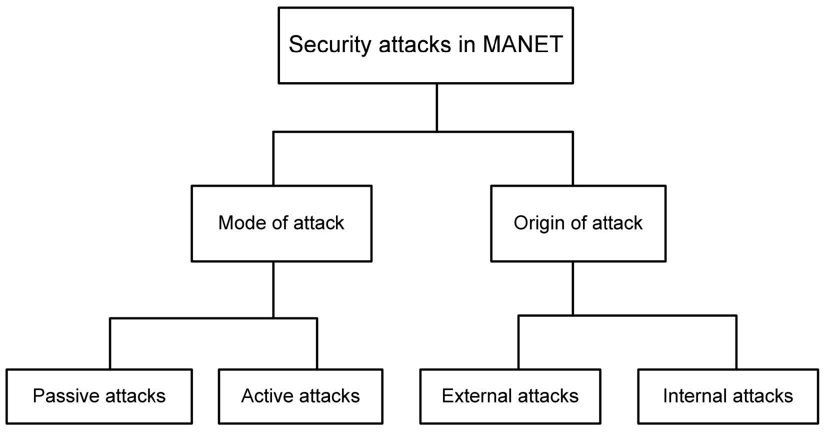


Figure 1.6 Classification of security attacks in MANET

1.1.3.1 Classification Based on the Mode of Attack

Passive Attack

The attackers in a passive attack can obtain the data exchanged in the network without disrupting any network operations. They can also launch an active attack by using the previously obtained information [15]. Due to the nature of the shared wireless communication medium, it is easier for an attacker to launch passive attacks in MANET than in wired networks. Examples of passive attacks include: *eavesdropping* which involves intercepting and reading messages by unintended receivers [16], and *traffic analysis* where the attackers analyze the data on who is communicating with whom, how often, how much and when [17].

Active Attack

In an active attack, the attackers disrupt the normal functionality of the network, which includes activities such as information interruption, modification, or fabrication. Examples of active attacks are: *sleep deprivation torture*, which targets the batteries; *jamming*, which results in channel unavailability by overusing it; *hijacking*, in which the attacker takes control of a communication between two entities and masquerades as one of them and *attacks against routing protocols*. Most of these attacks cause denial of service (DoS), which is degradation or complete halt in communication between nodes.

1.1.3.2 Classification Based on the Origin of Attack

External Attack

External attacks are launched by a node or a group of nodes that does not belong to the logical network. Therefore, the attackers are not capable of accessing the network and that limits their ability to disrupt the network services. Nevertheless, the attackers can attempt to jam the communication channel to interrupt the availability of the network. It is also possible that the attackers form a wormhole tunnel, which misguides two distant nodes in believing that they are direct neighbors of each other. In the extreme case, the attackers can eliminate a node from the network [18].

Internal Attack

Internal attacks are carried out by an internal compromised or malicious node which a part of the network domain. This is a more severe attack because the attacker knows secret information and possesses privileged access rights. So, the internal attackers have the same capabilities of outside attackers, plus the ability to participate in the network protocols and eventually deviate from the normal behavior of the protocols. Some possible internal attacks include *route disruption attacks* such as routing loops, black holes, grey holes, packet dropping, wormhole with selective forwarding, rushing attack, and Byzantine attacks (e.g. Byzantine wormhole attack) [18].

1.1.3.3 Attacks against the Routing Protocols

Network layer protocols extend the connectivity from neighboring *one-hop* nodes to all other nodes in MANET. So, the main network layer operations in MANETs are ad hoc routing and data packet forwarding. In other words, MANET routing protocols exchange routing messages between nodes and maintain routing states at each node. The data packets are forwarded by intermediate nodes along an established route to the destination. The family of routing attacks refers to any action of advertising routing updates that does not follow the specifications of the routing protocol [4]. By attacking the routing protocol, the attackers can inject themselves into the path between the source and destination. A variety of attacks that target the routing protocols in MANET have been presented in this section.

Flooding Attack

Ad hoc flooding attack acts as a denial of service (DoS) against all on-demand ad hoc routing protocols [19]. In particular, existing on-demand routing protocols, such as AODV [7], DSR [11], and some secure routing protocols, such as SAODV [20] [21], Ariadne [22], ARAN [23], cannot be immune from flooding attack. The aim of this type of attack is to exhaust the network resources, such as bandwidth and to consume a node's resources, such as computational and battery power or to disrupt the routing operation to cause severe degradation of the performance of the network [24]. For instance, in AODV protocol, a malicious node can send a large number of RREQs within a short period of time to a destination, which is not in the network domain. As a result, the RREQs will flood the whole network but no reply (RREP) will be generated, because the destination node does not exist.

Blackhole Attack

In a blackhole attack, the attacker attracts data packets and then drops them by distributing false routing information [25]. The attacker claims that it has an optimum route. As a result, other good nodes tempt to route data packets through the malicious node. For example, in AODV routing, the attacker can send to the source a fake RREP with a fabricated destination sequence number, which is equal to or higher than that in RREQ packets.

In Fig. 1.7 on the following page, an example of a blackhole attack in AODV routing protocol is shown. The source node S broadcasts RREQ to its neighbors to find a path to the destination D. The attacker A sends a fake RREP back to the source node S, claiming that it has an optimum route to D. As a result, node S will choose the route that passes through A.

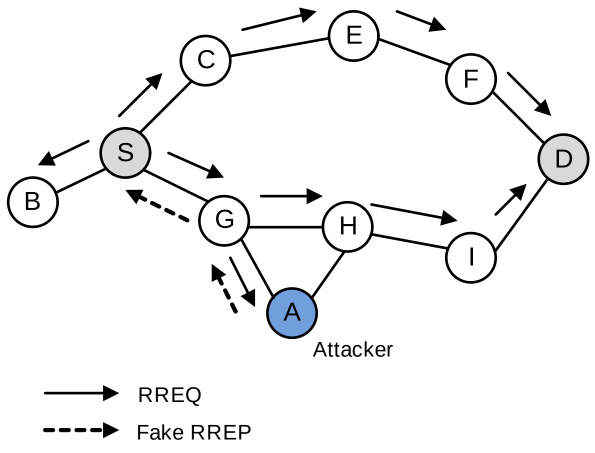


Figure 1.7 Blackhole attack in AODV routing

Link Spoofing Attack

In a link spoofing attack, an attacker advertises fake links with non-neighboring nodes to disrupt the routing operations [24]. This attack can pose devastating impacts on the routing protocols that uses link state information (e.g. OLSR). Particularly in OLSR routing protocol, an attacker can advertise a fake link with the target's *two-hop* neighbors. As a result, the target node selects the attacker to be its MPR and exchanges TC messages with the attacker. So, the attacker as an MPR can manipulate data and routing traffic which causes routing disruption.

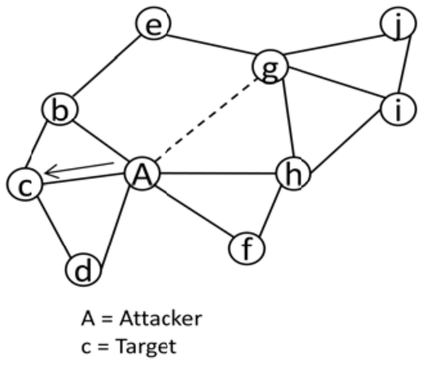


Figure 1.8 Link spoofing attack

In Fig. 1.8, a link spoofing attack scenario is presented where node A is the attacker and node c is the target. During the attack, node A declares a false link with node g which is one of the target's (node c) *two hop* neighbors. As a result, node c selects node A as the next hop for communicating with node g. So, node A can then drop or withhold the routing traffic generated by node c.

Wormhole Attack

Wormhole attack is one of the most sophisticated forms of routing attacks in MANET. In this attack, an attacker records packets at one location, tunnels them to another location of the network, where it is retransmitted by a colluding attacker. The tunnel can be established by using either out-of-band private link (e.g., a wired link, or a long-range wireless transmission), or logical link via packet encapsulation. As a result, the tunneled packets arrive either sooner or with less number of hops compared to the packets transmitted over multi-hop routes. Based on the tunnelling mechanism they use, wormholes can be classified into the following categories:

i. Out-of-band wormhole
ii. In-band wormhole

Before we discuss the implications of wormholes in MANET, the major differences between in-band and out-of-band wormholes are listed below:

a. In an out-of-band wormhole, the colluders create a direct link between the two end-points, whereas in-band wormhole does not use any external communication medium.
b. Out-of-band wormhole requires special hardware to support the communication between the two end-points. On the other hand, in-band wormhole does not require any special hardware or special routing protocol.
c. In out-of-band wormhole, the tunneled packets arrive faster than the multi-hop packets, but the in-band wormhole works much slower

than its counterpart. In both forms of wormhole, the colluding nodes create the illusion that two remote regions of a MANET are directly connected through nodes that appear to be neighbors.

d. In-band wormhole attack can be launched easily by any node in the network to another colluder or a set of colluders which may include one or more relay nodes. So, in-band wormholes are more likely to be used in real adversaries.
e. Out-of-band wormhole adds channel capacity to the network, whereas in band wormhole consumes network capacity and thereby causes internal service degradation [26].
f. In both forms of wormhole attack, the attackers can tunnel packets which are not even addressed to them [27]. They can do so even if the network provides confidentiality.

Since in-band wormhole attack is simple to be implemented, it is more likely to take place in real life scenarios. In this research project, our focus is on detecting in-band wormhole attack in MANET. So, in this dissertation, we use the terms "**in-band wormhole attack**", "**wormhole attack**", and "**traditional wormhole attack**" for the same meaning.

Wormhole could be a useful networking service while it provides a long link to the link layer [28]. However, the adversaries may use the wormhole link for their own purposes. The existence of wormhole links can disrupt the routing service in a number of ways. The attackers can attract a significant amount of traffic from their surroundings. If the attackers keep the wormhole tunnel active at all times and do not drop any packets, they would actually perform a useful service for the network [29]. But they can be responsible for disrupting the data flow by selectively dropping or modifying packets, generating unnecessary routing activities by turning off the wormhole link periodically, and recording packets for later analysis. In Fig. 1.9 and Fig. 1.10, a two-hop wormhole attack scenario is presented, where W_1 and W_3 are the main attackers and W_2 acts as a relay node. The attackers W_1 and W_3 encapsulate RREQs received from the nodes in their vicinity, and then forward them to the tunnel node W_2. The attackers decapsulate the packets received from the relay node W_2, and then rebroadcast them in their vicinity. For example, RREQs from nodes D and L will be tunneled from W_3 to W_1 and then rebroadcasted by W_1 and received by nodes E, C and S. These nodes will reply with RREP to acknowledge the RREQ. As a result a source node, for example S, will select a route to a destination (e.g. node D) which passes through the wormhole tunnel.

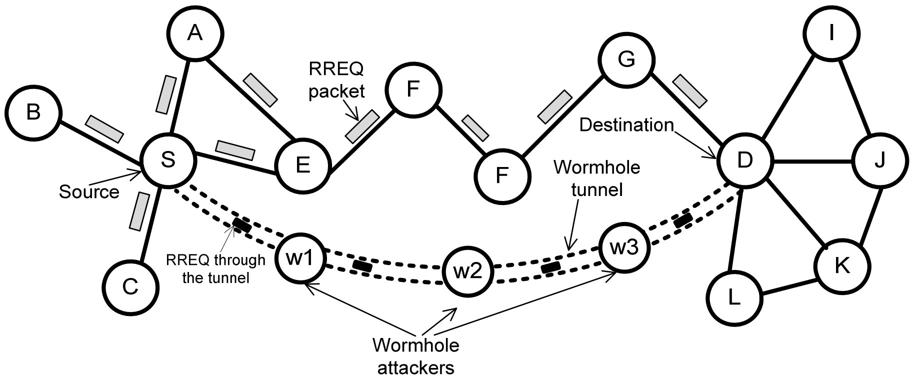


Figure 1.9 Path of RREQ packets (S→D) in presence of wormhole tunnel

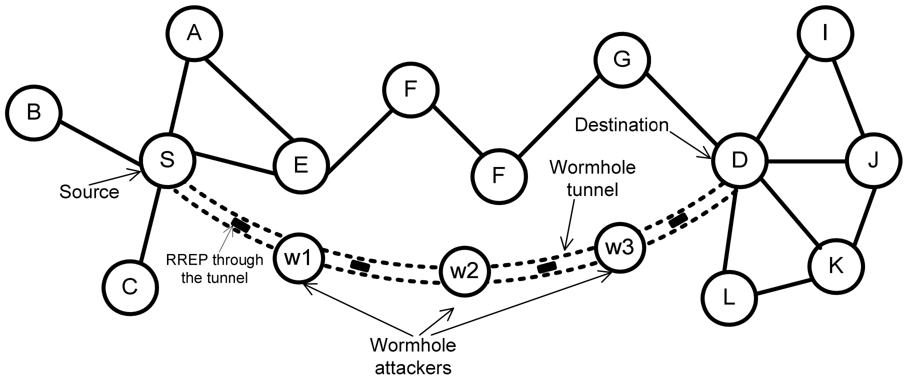


Figure 1.10 Path of RREP packets (D→S) in presence of wormhole tunnel

Byzantine Attack

The Byzantine term was introduced in [30], which addressed the problem of trying to reach agreement between Byzantine generals in the presence of traitors. In general, the term is used to denote participants whose actions cannot be trusted, or whose action do not conform with protocol specification [31]. In a MANET, the participating nodes are considered legitimate after a formal authentication procedure. Once authenticated, these nodes are given full control of the network and allowed to participate in network operation. This leads to the Byzantine wormhole problem when these authenticated nodes start misbehaving and disrupting the network operations [32]. The aim of the Byzantine nodes is to disrupt the communication of other nodes, but still participate in the routing protocol correctly. It is possible to deploy the following types of attacks by the Byzantine nodes in MANET: black hole attack, flood rushing attack, Byzantine wormhole attack, and Byzantine overlay network wormhole attack. In this research project our focus is on Byzantine wormhole attack, the most sophisticated form of Byzantine attacks. Before we illustrate the features of Byzantine wormholes, we present the major differences between traditional wormhole and Byzantine wormhole attack as follows:

a) In traditional wormhole attack, the colluders can fool two honest nodes into believing that there exists a direct link between them. But in Byzantine wormhole attacks, the wormhole link exists between the compromised nodes and not between the honest nodes.

b) In traditional wormhole attack, the colluders are invisible to the honest nodes. It is because of the fact that the colluders do not participate in any network operations. The nodes at the endpoints of the wormhole tunnel overhear the ongoing transmissions in their vicinity and also tunnel the routing packets originated by the nodes within their transmission range. In Byzantine wormhole attacks, the colluders are active participants in the network. In this form of wormhole attack, the attackers are authenticated nodes having full access to the network resources.

c) Traditional wormhole attacks fall in the category of external attacks in MANETs. This is because of the fact that the attackers can be external entities pursuing an attack after the network is formed. The attackers do not require authentication or cryptographic keys to form a tunnel in between two honest nodes placed in different network regions. On the other hand, Byzantine wormhole attackers are authenticated nodes, which have to be compromised to form a tunnel in between. So, Byzantine wormhole attack is an internal security attack in MANET.

The adversaries in a byzantine wormhole attack can create the tunnel either by using a private communication channel, such as a pair of radios and directional antennas, or by using packet encapsulation. The adversaries can use the low cost appearance of the wormhole links in order to increase the probability of being selected as part of the route, and attempt to disrupt the network by dropping all of the data packets. The byzantine wormhole attack is an extremely strong attack that can be performed even only two nodes in the network have been compromised. In Fig. 1.11, nodes W_1 and W_3 are the two colluders who take part in the routing process and also tunnels routing and data packets from their respective neighborhoods. The other attacker W_2 is only responsible for forwarding packets to and from nodes W_1 and W_3. For example, when a source node *S* wants to find a route to a destination *D*, the Byzantine attacker W_1 tunnels the RREQs from node *S* to the other colluder W_3 and via the relay node W_2. Nodes W_1 and W_3 will increment the hop count as they forward the RREQ packets. However, node W_2 is an intermediate node in the tunnel and it will not update any information on the RREQ packets. As a result, node S will select the path $S \rightarrow W_1 \rightarrow (W_2) \rightarrow W_3 \rightarrow D$, which is actually *four hops* long. As the intermediate node W2 will not maintain routing specifications, the communication path between node S and node D will appear to them as $S \rightarrow W_1 \rightarrow W_3 \rightarrow D$, which is *three hops* long.

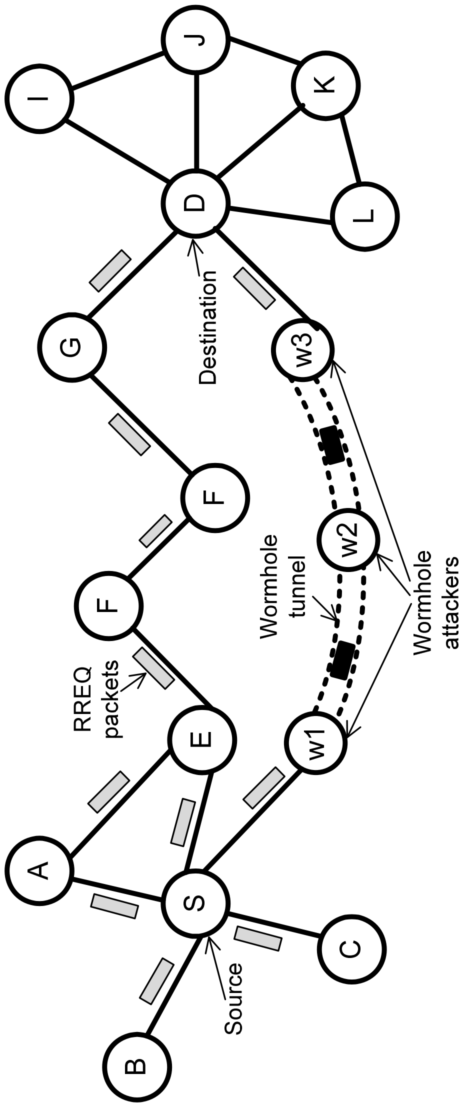


Figure 1.11 Two-hop Byzantine wormhole tunnel

1.2 Context of Research

This thesis investigates two of the most sophisticated malicious attacks on MANET: traditional and Byzantine wormhole attacks. The aim of this study is to design detection methods for both traditional and Byzantine wormhole attacks by using topological information and round trip time (RTT) measurements. To achieve better performance than the existing detection schemes, efforts are placed on taking decisions based on the outcome of topological comparisons, rather than only RTT.

At the first step of traditional wormhole detection, the originator of the detection process creates a list of its *one hop* neighbors, and also calculates the average round trip time (RTT_{avg}). A list of suspected neighbors is generated on the basis of RTT_{avg} and RTT. Then, topological information is exchanged between the initiator and all the suspected neighbors before declaring the presence of a wormhole link.

In the case of detecting Byzantine wormhole attack, the originator creates its one-hop neighbor list and calculates the average RTT_{avg} as in the case of detecting traditional wormhole attack. Unlike the detection of traditional wormhole attack, the main purpose of the first step here is to calculate the RTT_{avg}, which is later used as the key parameter to create suspected *three hop* neighbor list. Then, the initiator exchanges topological information with all its *one hop* neighbors to detect the presence of any Byzantine wormhole tunnel.

We have also compared the performance of our schemes with some of the existing detection methods reported in the literature. One of the major advantages of the topological comparison based approach is that the nodes which are initially suspected get a second chance to prove their reliability by exchanging their relative positions with respect to the originator.

1.3 Summary of Contributions

To the best knowledge of the author, this thesis makes the following original contributions to the knowledge of the field of study.

(a) Topological comparison based approach has been designed for detecting traditional Wormhole attacks. Most of the existing detection schemes rely on RTT measurement between two neighboring nodes, complex authentication schemes, or require special hardware/middleware, which may lead to significantly increased system complexity or unsatisfactory detection performance. In topological comparison based approach, RTT measurements are used to divide the neighbor list of a node into two segments: trusted (TRST) and suspected (SUS). Then, *topological information* is exchanged between an originator of detection process and all nodes in its SUS list.

(b) A topological comparison based method for detection Byzantine Wormhole attacks has been formulated. After observing the properties of byzantine wormhole attacks, we show that two attacked nodes always find themselves *three hops* away from each other. Hence, tunnels are detected by combining *one hop* and *three hop* neighborhood information. Most of the existing detection schemes for byzantine attacks considers implementing new secured protocols, or consider special network types, which use network coding or multicast routing protocols. The detection method presented in this thesis is versatile and also simple to implement.

(c) A round trip time (RTT) measurement technique, at the MAC layer of the protocol stack, has been formulated. All outgoing neighborhood detection packets from an originator include the time of departure from its MAC layer. On the other hand, every receiver records the arrival time of a neighborhood detection packet at the MAC layer and update the parameter used to measure the propagation time. This approach eliminates the queuing delays at the upper layers of the protocol stack. Hence, the processing time gives an indication of the distance between two nodes.

(d) In topological comparison based method, an initially suspected link gets the second chance to prove its credibility. It is considered that two legitimate nodes may suffer higher round trip time due to factors like congestion, attenuation and weak signal strength. Moreover, the wireless medium is open to both honest nodes and malicious nodes. So, making decisions based on round trip time (RTT) only may lead to unsatisfactory detection rates. The methods presented in this dissertation decide an attack in two phases. As a result, comparatively higher detection rate and accuracy of alarms are achieved.

(e) A comprehensive evaluation of the performance of the topological comparison based method has been carried out by means of simulations under a variety of tunnel length scenarios. The nodes in the network are placed randomly and the attackers are placed by observing the node positions such that the tunnel attracts as many nodes as possible. The results provide network designers and operators with a better understanding of the trade-off between different network scenarios and achievable performance.

1.4 Works from This Thesis Published by Author

The early work on detecting wormhole attacks in this thesis has been accepted for publication in the following conference proceedings:

M. Alam and K. S. Chan, *Detecting wormhole attacks in mobile ad hoc networks*, PEECS, Edith Cowan University, Perth, Australia, October 1, 2009.

M. Alam and K. S. Chan, *RTT-TC: A Topological Comparison Based Method to Detect Wormhole Attacks in MANET*, 12th IEEEICCT, Nanjing, China, November 11-14, 2010.

M. Alam and K. S. Chan, *Detecting Wormhole Attacks by Using Topological Comparison in OLSR Routing in MANET*, 4th ICSPCS, Gold Coast, Australia, December 13-15, 2010.

1.5 Organization of the Thesis

The organization of this thesis is as follows:

In Chapter 2, the literature review for the thesis is presented. Existing solutions to both traditional wormhole and Byzantine wormhole attacks are categorized which is followed by discussions of their limitations. Then, the objectives of this research are pointed out.

In Chapter 3, topological comparison based detection schemes for traditional wormhole attack is presented for both AODV and OLSR routing protocols in MANET. The performance of the proposed methods is compared with RTT-only method and some of the existing wormhole detection methods.

In Chapter 4, a topological comparison based detection scheme for Byzantine wormhole attack is presented. Chapter 5 concludes the thesis by providing a review of the research findings, and recommending some future research directions.

2. Current State of the Detection of Wormhole Attacks

In this chapter, existing detection schemes for both traditional and Byzantine wormhole attacks have been studied. The limitations of the existing methods as well as the scopes for designing a new detection mechanism are also discussed. Hence, the aim of the study is to design a detection approach which is easy to be implemented and achieves satisfactory detection performance.

2.1 Combating Traditional Wormhole Attack

The wormhole attack is one of the most sophisticated attacks in MANET. As this attack can be carried out in different ways, detection of this type of attack is a challenging task. In recent years, different approaches for detecting wormhole attacks have been proposed. They can be classified into three categories:

a) One hop delay based approaches.
b) Topological analysis based approaches.
c) Approaches relying on special hardware/middleware.

2.1.1 One Hop Delay Based Approaches

In [33], a wormhole detection method based on round trip time (RTT) and neighbor number is presented. Their method operates in the following three phases:

1. *construct neighbor list.*
2. *find route between source and destination.*
3. *find the location of the wormhole link.*

In this approach, it is assumed that the adversaries impose the following impacts on MANET:

- increase the number of neighbors of the nodes they target;
- shortens the path between the targeted nodes;
- increase the RTT between two successive nodes.

When the RTT between two nodes is considerably greater, they check the neighbor numbers. If the number of neighbors is greater than the average neighbor number, there is a suspect that a wormhole link is in between. The average number of neighbors, *d*, is calculated using the following formula:

$$d = \frac{(N-1)\,\pi\, r^2}{A} \tag{2.1}$$

In equation 2.1, *A* is the area of the region, *N* is the number of nodes in that region and *r* is the common transmission radius. This method assumes that all the nodes in the network contain the same hardware and software configuration. Moreover, they calculate the average neighbor number using a formula, which is more applicable in when the nodes in the network are evenly distributed.

Nait-Abdesselam et al. [34] proposed a wormhole detection method in OLSR protocol which attempts to pinpoint wormhole links before applying the detection algorithm. In addition to OLSR’s topology control (TC) message, two new control packets are used: $\text{HELLO}_{\text{req}}$ and $\text{HELLO}_{\text{rep}}$. The $\text{HELLO}_{\text{req}}$ message is used to request an explicit reply from the neighbor. If the $\text{HELLO}_{\text{rep}}$ from a node is not reached before a predefined timeout interval (*Timeout*), the originator of the detection process ranks that node as suspicious and stops communication with it until the end of wormhole verification process. To avoid overloading the network with too many $\text{HELLO}_{\text{rep}}$, a receiver of $\text{HELLO}_{\text{req}}$ delays the replies of multiple requests until it is scheduled to send its normal HELLO message, and piggybacks the replies to this HELLO message.The aggregation of $\text{HELLO}_{\text{rep}}$ is shown in the following figure (Fig. 2.1).

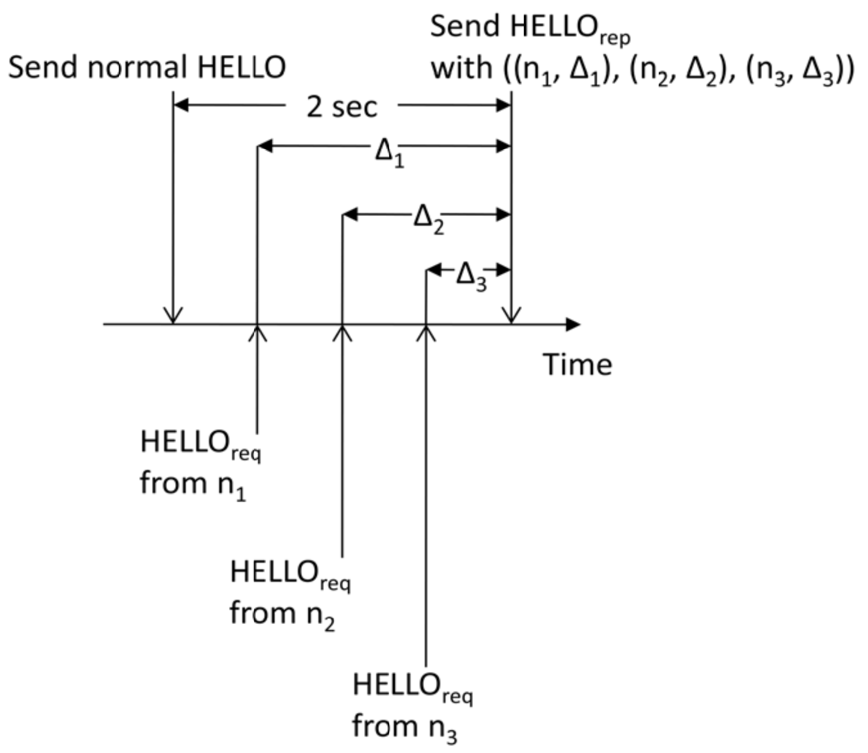


Figure 2.1 HELLO$_{rep}$ aggregation

Figure 2.1 shows the timing diagram where HELLO$_{rep}$ aggregates replies of three previously received HELLO$_{req}$ messages. The timeout interval (*Timeout*) for receiving HELLO$_{rep}$ is calculated using the following formula:

$$Timeout = 2R/V + T_{proc} \tag{2.2}$$

In equation 2.2, *R* denotes the maximum transmission range of each node, *V* is the propagation speed of the wireless signal (e.g., light speed *C*), and T_{proc} is the packet processing time and queuing delays within nodes. In the verification phase of this method, the originator sends *probing* packets to all the suspicious nodes. In response to the *probing* packet, a node sends its evaluation of the originator. Since both phases of this detection method depends on delay only approach, it may result in higher false alarm rate.

Van et al. in [35] presents a transmission time based mechanism (TTM) to detect wormhole attacks. TTM detects wormhole during route setup procedure by computing transmission time between every two successive nodes along the established path. Wormhole is identified based on the fact that transmission time between two fake neighbors created by wormhole is considerably higher than that between two real neighbors which are within each other's communication range. In this approach, each node in the established route calculates the RTT between it and the destination and sends this value back to the source. The source node collects all the RTT values, calculating RTTs between two successive nodes. The timing diagram of this method is shown below:

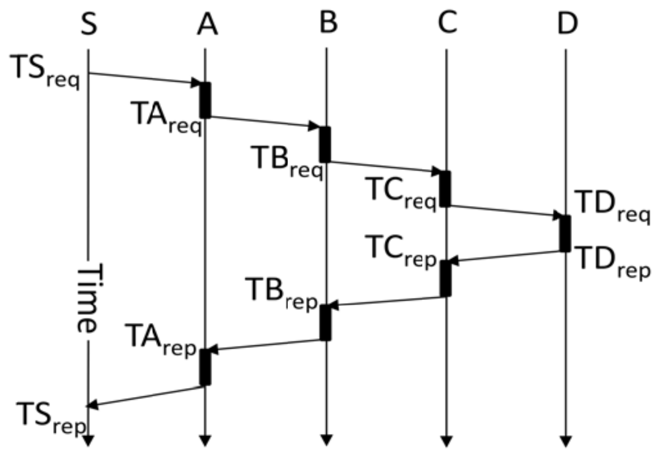


Figure 2.2 Timing diagram of TTM method

In Fig. 2.2, the time of sending RREQ and receiving RREP in each node along a route (S→A→B→C→D) is shown. Under normal circumstances $RTT_{S,A}$, $RTT_{A,B}$, $RTT_{B,C}$, $RTT_{C,D}$ are similar but if there is a wormhole link (e.g., between B and C), $RTT_{B,C}$ is considerably greater than other RTT values. This is another delay only detection method which may result in poor false alarm rate when two legitimate neighbors suffer link congestion or have different intra-nodal processing speed.

In [36], the authors propose a timing based countermeasure against the wormhole attack where a node can estimate its distance to a sender by multiplying Packet Travel Time (*PTT*) by the light speed (*c*). A node *A* accepts another node *B* as neighbor if the following condition is satisfied:

$$\frac{(t_{A,B} - t_A) - (t_B - t_{B,A})}{2} \times c \leq T_{max} \tag{2.3}$$

In equation 2.3, t_x is the sending time of x's HELLO message recorded by x, $t_{x,y}$ is the receiving time of y's HELLO message recorded by x and T_{max} is the maximum transmission range. So, if the propagation delay is greater than the maximum possible propagation delay, then wormhole attack is detected. Unfortunately, propagation delay is difficult to be accurately measured.

In another approach presented in [37], the authors propose a method called Wormhole Attack Prevention (*WAP*). It is assumed that each node remains in promiscuous reception mode so that it can always overhear ongoing transmissions. Each node also maintains a Neighbor node table that contains RREQ sequence number, neighbor ID, sending time and receiving time of the RREQ and count. This table is used to monitor the activities of the neighbors. A Wormhole Prevention Timer (WPT) is initiated as soon as a node sends a RREQ. The WPT is calculated as follows:

$$WPT = \frac{2 \times Transmission\ Range\ (TR)}{V_p} \tag{2.4}$$

In equation 2.4, TR denoted the transmission range of a node; V_p denotes the propagation speed of a packet (maximum with the speed of light). In *WAP*, the time interval between when a packet is forwarded by a neighbor since its transmission is measured. If that delay is too long (greater than WPT) then wormhole attack is detected. Unfortunately, this is another delay only mechanism that does not consider delays incurred by congestion or intra nodal processing speed.

2.1.2 Methods Based on Graph Analyses

Another approach of combating wormhole attacks is to use the *neighborhood information*. Maheshwari et al. in [28] proposed a wormhole detection algorithm which looks for forbidden substructure in the connectivity graph that should not be presented in a legal connectivity graph. The authors considered two following communication models for their proposed wormhole detection method:

- Unit disk graph (*UDG*) model, where each node is modeled as a disk of unit radius.
- General (known or unknown) communication model.

The key notion exploited in the UDG model is *Disk packing argument* – inside a fixed region, one cannot pack too many nodes without having edges in between. The forbidden substructures are actually those who violate the packing argument. In Fig. 2.3, the authors prove that there can only be two nodes inside a lune with inter distance larger than 1. In presence of wormhole, tow independent nodes in one region may share more than two common independent nodes. This constitutes a forbidden substructure, since in any valid UDG embedding of the connectivity graph the existence of more than two independent nodes inside a lune is not possible. In this approach, each node searches for forbidden substructure in its *k*-hop neighborhood. The main steps followed in the wormhole detection algorithm are:

- Find the forbidden parameter f_k
- Each node u determines its 2*k*-hop neighbor list $N_{2k}(u)$
- Node u determines the set of common *k*-hop neighbors with v where $C_k(u,v)=N_k(u) \cap N_k(v)$
- Node u determines the maximal independent set of the sub-graph on vertices $C_k(u,v)$.

- If the maximal independent set size equal or larger than f_k, node u declares the presence of a wormhole.

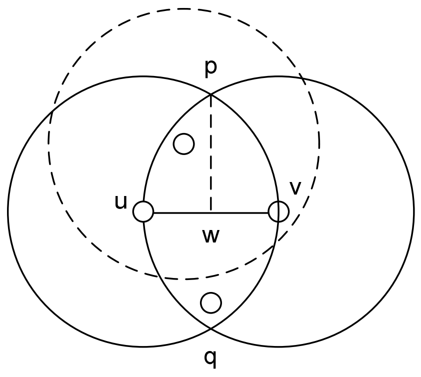


Figure 2.3 It is possible to pack at most two nodes inside a Lune with inter-distance more than one

The algorithm remains the same for detecting wormhole attacks in general communication model except the determination of the forbidden parameter f_k in the first step. This approach is very complicated and impractical when the communication model is not known. The performance of the detection method largely depends on forbidden parameter f_k. In the case of unknown communication model, which is very much expected in wireless ad hoc networks, the steps followed to determine f_k are not definite. Determination of f_k is crucial because the wormhole detection rate of is inversely proportional to it but when it is too small we may have greater false positives.

Lee et al. in [38] propose a method which checks whether a node that forwards a packet is a real neighborhood or not. In this approach, each node gathers information of its neighbors within two hops. Each newly joined node broadcasts an announcement which is valid until the next two hops. The requirement of maintaining two types of neighbors, keyed hash and TTL limit the applicability of this method in a distributed system where exists a wide variety of participants.

2.1.3 Special Hardware/Middleware Based Methods

Specialized methods use a special hardware device, strict time synchronization or special network protocol. Packet leashes are used in [27] to detect and defend against wormhole attacks in MANET. In this approach, the maximum distance a packet can travel is restricted by means of packet leashes. Two types of leashes are used: temporal leash and geographical leash. In temporal leash, a sender includes the sending time in the packet. A receiver of the packet compares the sender's time with the receiving time to check whether the packet has traveled too far. An authentication protocol, TIK, has been proposed to authenticate the nodes in the network. The authors also assumed that all nodes in the network have tightly synchronized clocks. On the other hand, in geographical leash approach, a sender includes its own geographical location and the sending time in the packet. The receiving nodes compare these values with its location information and receiving time. As a result, a receiver can compute an upper bound on the distance between the sender and itself. The temporal leash approach requires strict clock synchronization and the geographical leash approach requires special hardware for knowing geographical location of a node. We know that strict clock synchronization is hard to achieve in a dynamic environment. In addition, the accuracy of GPS devices in presence of physical obstacles is low. As a result, both the approaches presented in [22] are expensive in terms of their hardware requirements.

In [39], the authors propose to use directional antennae to combat wormhole attacks. In this approach, each node maintains accurate sets of their neighbors. Wormhole attacks are avoided if the attacker nodes are recognized as false neighbors. The security relies on using directional antennae to obtain relative direction information and cooperation among nodes to verify possible neighbors. This method suffers from antenna's directional errors which sometimes eliminates trustable links and thus increases the false alarm rate.

In NEVO [40], a firmware up-gradation of the MAC layer is needed so that the sender can passively monitor the forwarding of broadcast type packets by its neighbors. The detection method is designed based on the following observations:

- If *t* represents the time taken to traverse a packet (e.g., RREQ) from a sender (e.g., node *x*) to a one hop neighbor (e.g., node y), the neighbor *y* must rebroadcasts the packet within *t* + δ seconds to be considered as a true neighbor.
- If node *y* is a fake neighbor, it takes at least 3*t* + δ seconds to rebroadcast the packet.

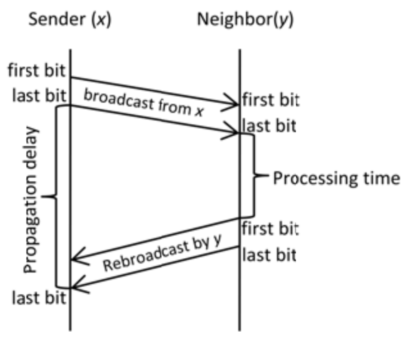


Figure 2.4 Timing diagram of NEVO (no wormhole)

In Fig. 2.4, the timing analysis of NEVO with no attack is shown. According to the timing diagram, the propagation delay is measured as the time elapsed from when the last bit of a packet is sent by the sender to when the last bit of the same packet is overheard. In presence of wormhole tunnel (in between *x* and *y*) the propagation delay is increases significantly.

2.2 Combating Byzantine Wormhole Attack

In Byzantine wormhole attacks, more than one compromised nodes collude to form a wormhole tunnel. Here, the colluders are authenticated nodes which also take part in general network operations. Moreover, the wormhole link in byzantine wormhole attacks exists between two compromised nodes, while in traditional wormhole two honest nodes are tricked to believe that there exists a direct link between them [41]. Many solutions proposed against traditional wormholes are ineffective in the case of Byzantine wormhole because of the trust of the wormhole link end points, which are also adversarial. Existing solutions for Byzantine wormhole attacks mainly focus on the networks that employ the following approaches:

a) network coding (e.g., RLNC [42])

b) multicast routing (e.g., multicast AODV [43])

c) general routing (e.g., AODV, OLSR)

In this dissertation, we focus on Byzantine wormhole attacks in reactive/proactive routing protocols. So, in this section, we discuss the existing detection methods targeted to the general routing protocols, such as reactive or proactive routing. In most of the recent works, secure routing protocols have been proposed to defend against byzantine wormhole attacks. We present brief overviews of two such secured routing protocols: On-Demand Secure Byzantine Resilient routing (ODBSR [41]), and Secure Routing Against Collusion (SRAC [44]).

2.2.1 On-Demand Secure Byzantine Resilient Routing (ODBSR)

ODBSR protocol is an on-demand routing protocol for wireless ad hoc networks that detects Byzantine behavior and avoids it. The protocol is designed to locate a fault free path in an ad hoc network, even when a majority of nodes have been compromised [31]. ODBSR addresses both failures and attacks within a unified framework. A fault is defined as any disruption that results in significant loss or delay. Upon detection of the attack, ODBSR enters probing mode with the goal of discovering the attack location.

Unlike other protocols, ODBSR does not use number of hops as path selection metric. In ODBSR [41], the shortest path is selected based on a reliability metric, which captures reliability and adversarial behavior based on past history. Each node in the network maintains its own list, referred to as *weight list*, and dynamically updates the list when it detects faults. These faulty links are avoided using a secure route discovery protocol that incorporates the reliability metric. The ODBSR routing protocol can be separated into three successive phases:

- Route discovery in an adversarial environment
- Byzantine fault detection
- Link weight management

2.2.1.1 Route Discovery

In the route discovery phase, the ODBSR protocol follows the following five steps:

1. *Request Initiation*: The source creates and signs a request, and then the source broadcasts the route request to its neighbors. The source's signature allows the destination and intermediate nodes to

authenticate the request and prevents an adversary from creating a valid route request.

2. *Request Propagation*: When receiving a request, an intermediate node first verifies the source signature on the request and checks its list of recently seen requests for a matching request. If there is no matching request in its list, and the source's signature is valid, it stores the request in its list and rebroadcasts the request. If there is matching request, the node does nothing.

3. *Request Receipt/Response Initiation*: Upon receiving a new request from the source, the destination verifies the authenticity of the request, and creates a response. The destination then signs the response and broadcasts it.

4. *Response Propagation*: Upon receiving a response, the node computes the total weight of the path by summing the weight of all the links. If the total weight is less than any previously forwarded matching response, the node verifies the signatures included in the packet. If the entire packet is verified, the node appends its identifier at the end of the packet and broadcasts the response.

5. *Response Receipt*: When the source receives a response, it performs the same verification as the intermediate nodes as described in the previous step. If the path in the response is better than the best path received so far, the source updates the route used to send packets to that specific destination.

2.2.1.2 Byzantine Fault Detection

Once the path is established, data can flow between source and destination. The ODBSR protocol is based on authenticated acknowledgements of the data packets. If a valid acknowledgement is not received within a timeout, ODBSR assumes that the packet was lost. The detection mechanism is presented below in more detail:

Fault Detection Mechanism**:** The *fault detection mechanism* follows the following steps:

1. The destination returns an acknowledgment to the source for every received data packet.
2. The source keeps track of the recent losses (acknowledgements not received). If the losses violate the acceptable threshold, the protocol registers a fault between the source and destination, and starts a binary search on the path to locate the faulty link.
3. The source controls the search by specifying on data packets the list of intermediate nodes that must send acknowledgements in addition to the destination. The nodes required to send acknowledgements are referred to as *probe nodes*. An adversary is unable to drop traffic without dropping the list of probes and eventually being detected.
4. The list of probes defines a set of non-overlapping intervals that cover the whole path, where each interval covers the sub-path between the two consecutive probes that form its endpoints. When a fault is detected on an interval, the interval is divided in two by inserting a new probe. This new probe is added to the list of probe appended to future packets. The process of subdivision continues until a fault is detected.

Acknowledgement Specification: For each successfully received data packet, the destination generates an acknowledgement containing the sequence number of the data packet and an HMAC for authentication. Each probe appends its own HMAC over the entire acknowledgement packet accumulated so far, and forwards it along the reverse path towards the source. Finally, when a source receives the acknowledgment packet, it attempts to verify the accumulated HMACs starting from the end of the packet. The protocol registers a loss on the interval between the last valid HMAC and the first encountered invalid HMAC. If the source times out the acknowledgement, a loss is registered on the interval between the source and the first probe.

Interval and Probe Management: A loss is attributed to an interval between two probes when the source successfully received and verified an acknowledgment from the closer probe, but did not from the further probe. When the loss rate on a n interval exceeds ρ, which is the acceptable threshold loss rate, the interval is divided in two.

Shared Key Establishment: ODBSR uses extensive use of pair-wise symmetric keys shared between the source and each node along the path. The authors proposed a technique for on-demand creation of these keys using the assumed public key infrastructure. Digital signatures are used to authenticate the packets before the shared key is established.

2.2.1.3 Link Weight management

An important aspect of the ODBSR protocol is its ability to avoid faulty links in the route discovery process by using a metric called link weight, which captures faulty and adversarial behavior. The goal is to penalize faulty links and to reward good behavior. When a fault is registered on a link, the link's weight is doubled. This ensures that the protocol will eventually avoid selecting paths containing that link during future route discoveries.

In addition to the weight, a counter is associated with each identified faulty link. This counter represents the remaining time before the link weight will be reset back to its initial value. If μ is the number of packets dropped while identifying a faulty link and ρ is the threshold loss rate, then the link's counter is increased by μ/ρ. Each nonzero counter is reduced by $1/m$ for every successfully delivered packet, where m is the number of links with nonzero counters.

2.2.1.4 Handling Byzantine Wormholes

ODBSR protocol approach to mitigate Byzantine wormholes is motivated by the observation that the primary attack when a wormhole exists is the dropping of packets that attempt to travel through wormhole, rather than the wormhole formation. A wormhole link will appear to ODBSR as a faulty link existing between two nodes. Once the wormhole's link weight has been increased sufficiently, ODBSR will avoid it and select the next best alternate path. ODBSR's ability to mitigate the wormhole will be reduced if many wormhole links are present. Moreover, if a wormhole is simply attracting traffic while waiting for a suitable time to disrupt it, or already subjecting that traffic to delays or packet corruption, the wormhole will remain invisible to ODBSR [45]. In addition, the colluding attackers may create a large number of fictious links, all of which must be identified as bad before the ODBSR protocol succeeds [46].

2.2.2 Secure Routing Against Collusion

In [44], the authors proposed an algorithm which detects Byzantine attacks by using both message and route redundancy during route discovery. An optimal routing algorithm, which uses routing metric combining both a node's trustworthiness and performance, is also presented. Both of the proposed algorithms can be integrated into existing routing protocols, such as AODV and DSR. In SRAC protocol, a node makes a routing decision based on its trust of its neighboring nodes and the performance provided by them. The secured routing protocol proposed in [44] has the following steps:

1. During the route discovery process, a source sends RREQs to its neighbors. In RREQ packets, in addition to regular information, the node also attaches its security related information. The RREQ packets have the following format:

$$m_q + h(m_q + k_1) + E(E(k_1, K_{z,pub}), K_{s,pri}) \qquad (2.5)$$

In equation 2.5, k_1 is a randomly generated key which acts as shared secret key between the source s and one its neighbors z, $K_{z,pub}$ is the public key of z, $K_{s,pri}$ is the private key of node s, m_q stands for the message used in RREQ, E( ) stands for encryption algorithm and h( ) is a keyed hash MAC algorithm. Node *z* replies with RREP packet as follows:

$$m_p + h(m_p + k_1) + E(E(k_1, K_{s,pub}), K_{z,pri}) \qquad (2.6)$$

In equation 2.6, m_p stands for the message used in RREP. Similarly, s establishes a shared key with each of its one-hop neighbors. Moreover, by using the double hash and signature operations, node *s* can establish shared keys between itself and its *n*-hop neighbors.

2. When an intermediate node receives an RREQ, it calculates the Trustworthiness-QoS index (TQI) by using the following formula:

$$D(p) = \beta(1 - T_x(p; j))Q_x(p), \quad \text{for } p \; \varepsilon \; P_{s\rightarrow x} \qquad (2.7)$$

In equation 2.7, $P_{s\rightarrow x}$ is the set of paths that start from a source node s to a destination node x, $\beta > 0$ is a constant used to scale the value of the cost function *D(p)*, $T_x(p; j)$ is the trustworthiness of the path assigned by node x, and $Q_x(p)$ is the objective function which is the average number of packets in the network. Then, the intermediate node attaches the link trustworthiness and QoS information to the RREQ packet and forwards it to its next hop. This process is repeated until it reaches the final destination.

3. At the destination, the node waits either for a fixed number of RREQs or a timeout before it makes a decision. The destination node unicasts RREP back to the source over the link which has the lowest TQI. When the source node receives the RREP, it starts data communication by using the route.

4. Once the route is established, the intermediate nodes monitor the link status of the next hops in the active routes. Those that do not meet the performance and trustworthiness requirements will be eliminated from the route.

5. When a link breakage in an active route is detected, a route error (RERR) packet is used to notify the other nodes that the loss of that link has occurred.

The SRAC protocol intends to optimize a combined objective function of security and performance parameters. The complexity of the solution to the routing problem is greatly reduced compared to the existing secured routing protocols. However, some assumptions of this protocol may not be practical. According to SRAC protocol, each node is required to receive multiple copies of the same route discovery message before sending back an acknowledgment. For example, to detect the collusion of n compromised nodes that are consecutively located on a route, a receiver must have at least n+1 copies of the same message, and one of the copies is more trusted than the others. Moreover, the redundant use of shared keys between a source and each intermediate and the destination node may result in a scalability problem. For example, if there are n nodes along a route, then the dynamic key management scheme needs to create and distribute $(n-1)^2/2$ keys to the nodes on the route. Hence, it is not appropriate for the networks with a large number of low-resource nodes.

2.3 Objectives of This Study

It can be noted that most of the existing detection methods reported in this chapter rely on RTT, specialized devices or graph analysis. The RTT based detection methods are unreliable in most cases, because of the assumption that the link containing wormhole tunnel suffers significantly long delay compared to other real links. Link delays also may be incurred in case of congestion, intra nodal processing speed and queuing delay.

Furthermore, in some studies, specialized devices, such as GPS, directional antennae etc., have been utilized. These studies introduced a new paradigm of wormhole detection by making use of external technology. However, in MANETs, the participants are low performance devices and they have energy constraints. So, utilizing such external technology may not be efficient in terms of cost and availability.

In some studies, detection methods based on graph analysis is proposed. These approaches make use of the graphical representation of a network and analytically avoid wormhole tunnels. The computational complexity of these methods is much higher than other detection approaches. Moreover, graphical analysis based approaches depends on the prior knowledge of the communication model. Unfortunately, in a dynamic network, such as MANET, the system model may change without any prior notification.

The detection mechanisms and secured routing protocols reported in the previous sections may be secure but not feasible or *vice versa*. On the other hand, MANETs have some vulnerable characteristics, such as open peer-to-peer architecture, no fixed infrastructure (e.g., router), and easily accessible wireless channel. As a result, there is no well defined place/infrastructure where a single security solution can be deployed [4]. In addition, the computation capabilities of mobile devices are limited. For example, PDAs can hardly perform computation-intensive tasks like cryptographic calculations. Hence, designing a security solution that achieves higher detection as well as desirable network performance is a challenging task. In this study, a topological comparison based approach to detect both traditional and Byzantine wormhole attacks is presented. In the topological comparison based approach, the topological misbehavior, such as bogus-shortcuts among the nodes is used as a symptom of the presence of a wormhole tunnel. Two distant nodes find themselves as direct neighbors in presence of a traditional wormhole tunnel, whereas Byzantine wormhole tunnel makes them believe that they are constantly three-hops away from each other. This approach is simple as well as effective in terms of wormhole detection rate and wormhole detection accuracy. The details of the topological comparison based approach are presented in the following chapters.

3. Detection of Traditional Wormhole Attacks Using RTT and Topological Comparisons

In presence of traditional wormhole tunnel, the RTT between two fake neighbors is much longer than between two true neighbors. However, longer RTT does not confirm the existence of a wormhole tunnel. This is because other than the wormhole tunnel there are factors (e.g., congestion, intra-nodal processing speed, geographical barrier etc.) that can also contribute to long RTT. In this chapter, a detection method for traditional wormhole attacks is presented. The proposed method detects traditional wormhole attack by using a topological comparison algorithm.

3.1 Round Trip Time (RTT) Measurement

In the field of telecommunications, Round Trip Time (RTT) is defined as the time interval between when a packet is sent and when the corresponding acknowledgement is received. One of the most common applications of RTT is finding the best possible route in a communication network. It can range from a few milliseconds (thousandths of a second) under ideal conditions between closely positioned nodes to several seconds under adverse conditions between nodes separated by a large distance. In the context of computer networking, RTT is also known as the *ping time* which can be determined by using the *ping* command.

3.1.1 RTT Measurement in TCP

One easy way of measuring RTT is to record the time when a packet is sent and calculate the elapsed time when the acknowledgement (ACK) is received. Unfortunately, in TCP there is no way to tell whether a received acknowledgement (ACK) is for an original or retransmitted packet. This is known as "retransmission ambiguity" problem [47]. P. Karn, one of the authors of [47] proposed an algorithm known as "Karn's algorithm" which addresses the problem by ignoring round-trip times of retransmitted packets.

In TCP, a sender records how long it takes for a packet to be acknowledged by producing a sequence of RTT samples (s_1, s_2, s_3….). TCP implementations estimate the future RTT of a connection by sampling the behavior of the packets sent over it and averaging those samples into an smoothed round trip time (SRTT). The formula used in SRTT is as follows:

$$SRTT_{i+1} = (\propto \times SRTT_i) + ((1-\propto) \times s_i) \qquad (3.1)$$

In equation 3.1, $SRTT_{i+1}$ is the new calculated value, $SRTT_i$ is the current estimate of the round trip time, and α is a constant between 0 and 1. The constant α controls how rapidly the smoothed round trip time adapts to change.

3.1.2 Our Proposed RTT Measurement Method

3.1.2.1 Asynchronous Clock

We propose to measure the round trip time (RTT) between a source and its n hop neighbors by broadcasting HELLO packets. The recipients of HELLO packet either rebroadcast it until the nth hop is reached or respond with a unicast $HELLO_{rep}$. In HELLO packet, the ***hops_to_leave*** header is used to indicate the number of hops it should travel. Besides**,** the ***broadcast time*** is recorded by the sender so that RTT can be calculated when a $HELLO_{rep}$ is reached. Each node maintains an exponentially weighted average round trip time (RTT_{avg}) for its n hop neighbors. The RTT and RTT_{avg} are calculated using the following formulae:

$$RTT_i = reciving\ time\ of\ HELLO_{rep} - broadcast\ time\ of\ HELLO \qquad (3.2)$$

$$RTT_{avg(0)} = RTT_0 \tag{3.3}$$

$$RTT_{avg(i)} = \left(\propto \times RTT_{avg(i-1)}\right) + ((1-\propto) \times RTT_i) \tag{3.4}$$

The difference between the SRTT used in TCP and the RTT$_{avg}$ in our proposed method is in the samples used in the exponentially weighted moving average (EWMA) formula. We use the RTTs between a sender and all *n* hop neighbors as samples. In the following figures we present a scenario to discuss the calculations of RTT and RTT$_{avg}$ for two hop neighbors (*n* = 2).

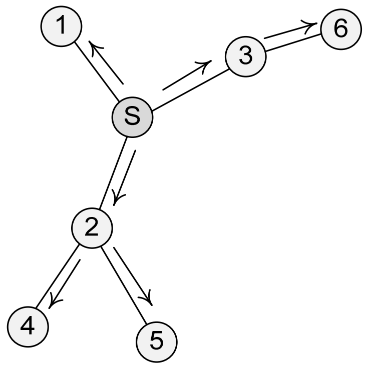


Figure 3.1 HELLO broadcast

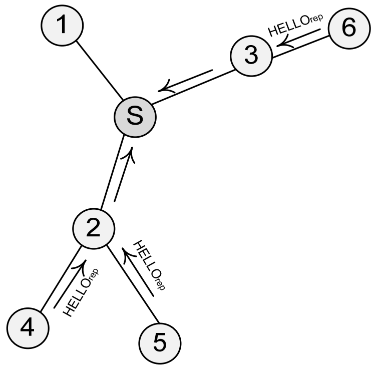


Figure 3.2 HELLOrep unicast

In Fig. 3.1, the source node (S) broadcasts HELLO messages in the network to calculate the RTT of its two hop neighbors (***hops_to_leave*** = *2*). Since nodes 1, 2 and 3 are within the transmission range of node S, they receive the HELLO from node S. They decrement the ***hops_to_leave*** field and rebroadcast the HELLO. It should be noted that repetitive HELLOs from the same source are dropped. Finally, the HELLO from node S reaches nodes 4, 5 and 6, which are two hops away. They send $\text{HELLO}_{\text{rep}}$ back to the source (S). The successive values RTT_{avg} are calculated upon each reception of $\text{HELLO}_{\text{rep}}$ at node S. For example, the $\text{HELLO}_{\text{rep}}$ from node 4, represented by $HELLO_{rep}^{4}$, reaches the source (S) before $HELLO_{rep}^{5}$ and $HELLO_{rep}^{6}$. Hence, the calculations at node S are as follows:

$$RTT_0^S = reciving\ time\ of\ HELLO_{rep}^4 - broadcast\ time\ of\ HELLO \tag{3.5}$$

$$RTT_{avg(0)} = RTT_0 \tag{3.6}$$

When node S receives $HELLO_{rep}^{5}$ it calculates the RTT and updates the value of RTT_{avg} as follows:

$$RTT_1^S = reciving\ time\ of\ HELLO_{rep}^5 - broadcast\ time\ of\ HELLO \tag{3.7}$$

$$RTT_{avg(1)} = \left(\propto \times RTT_{avg(0)}\right) + ((1-\propto) \times RTT_1) \tag{3.8}$$

Similarly, node S receives $HELLO_{rep}^{6}$ it calculates the RTT and updates the value of RTT_{avg}:

$$RTT_2^S = reciving\ time\ of\ HELLO_{rep}^6 - broadcast\ time\ of\ HELLO \tag{3.9}$$

$$RTT_{avg(2)} = \left(\propto \times RTT_{avg(1)}\right) + ((1-\propto) \times RTT_2) \tag{3.10}$$

In this particular case, the samples used for the exponentially weighted moving RTT_{avg} are R_0, R_1 and R_2 which are the RTTs between node S and nodes 4, 5 and 6 respectively. Hence, rather than calculating a SRTT for each individual neighbor, our method maintains a distributed RTT_{avg} taking individual RTTs as sample.

In a dynamic network like MANET, where nodes can join/leave the network without any prior notification or a node may be closed down due to energy constraints, a node can't be certain about the number of its *n* hop neighbors. So, it is desirable to have a predefined timeout interval, for example $timeout_{rtt}$, after which a node will discard any $\text{HELLO}_{\text{rep}}$ from its neighbors. We can correlate this $timeout_{rtt}$ with the retransmission time-out (RTO) of TCP which represents the amount of time the sender will wait for a given packet to be acknowledged. In TCP, the RTO is calculated from the smoothed round trip time (SRTT) and the deviation of round trip time ($\Delta\, RTT_i$), using the following formula:

$$RTO_i = 4 * \Delta\, RTT_i + SRTT_i \qquad (3.11)$$

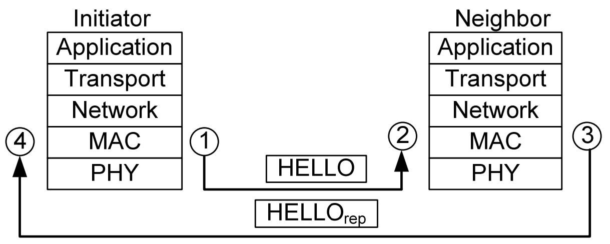
Since our goal is to create *n* hop neighbor list and divide it into two segments (*Trusted* and *Suspected,* which are introduced in later sections) so that we can apply our topological comparison scheme on the *Suspected* neighbors only, we set the $timout_{rtt}$ interval in relation to *n* as follows:

$$timeout_{rtt(i)} = k \times RTT_{avg(i)}, where\ k \geq n \qquad (3.12)$$

In equation 3.12, k is the *delay factor*, chosen such that there is a little chance that the $\text{HELLO}_{\text{rep}}$ from a real *n* hop neighbor will reach the sender after $timeout_{rtt}$. But there are some scenarios such as a wormhole attack, congested link, intra nodal processing speed, for which $\text{HELLO}_{\text{rep}}$ from a neighbors may reach the source after the $timeout_{rtt}$. If $HELLO_{rep}^{x}$ is reached the sender beyond $timeout_{rtt}$ then RTT_{x} is not used as a sample.

3.1.2.2 Synchronous Clock

If the clocks are synchronized, RTT can be measured at the MAC layer of the protocol stack. Two additional fields, ***$prop_t$*** and ***time***, are used in the HELLO packet. The value of ***$prop_t$*** and ***time*** denotes the actual propagation delay and local clock respectively. In this approach, ***$prop_t$*** and ***time*** are updated when a HELLO packet is reached the MAC layer. This approach produces better timing analysis because upper layer delays (specially the delay associated with routing in the network layer) are avoided. In Fig. 3.3, the RTT measurement in a synchronous network is shown.



Step 1: $prop_t$ = 0, time = current time
Step 2 & 4: $prop_t$ = $prop_t$ + current time -time
Step 3: $prop_t$ = $prop_t$, time = current time

Figure 3.3 Measuring RTT at the MAC layer

In the figure, four major steps of calculating RTT is shown. In step 1, when the HELLO packet reaches the sender's MAC layer, ***$prop_t$*** is set to 0, ***time*** is set to current clock, and then the HELLO packet is broadcasted in the network. In the 2nd step, when the HELLO packet reaches the MAC layer of the receiver, the one way propagation delay from the source to the receiver is calculated and saved in ***$prop_t$***, before the HELLO packet is pushed up to the upper layers. Upon receiving a HELLO packet at the application layer, HELLO$_{rep}$ packets are generated. Unlike the HELLO sender, the sender of the HELLO$_{rep}$ attaches the propagation delay calculated in step 2 in ***$prop_t$*** and then unicasts the HELLO$_{rep}$ back to the source. In this way, upper layer delays can be avoided. Eventually, when the HELLO$_{rep}$ reaches the MAC layer of the initiator, the round trip time is calculated using the same formula used in step 2.

3.2 Topological View of a Network

3.2.1 Topology of a Network

In the field of communication, network topology can be defined as the layout of the connected devices. The topology of a network can be viewed in two ways: physical topology or logical topology. The physical topology depicts the physical design of the network and where the workstations are positioned. On the other hand, the logical topology describes how information is transferred through the network. An example of network topology for 53 nodes in a MANET can be shown as below:

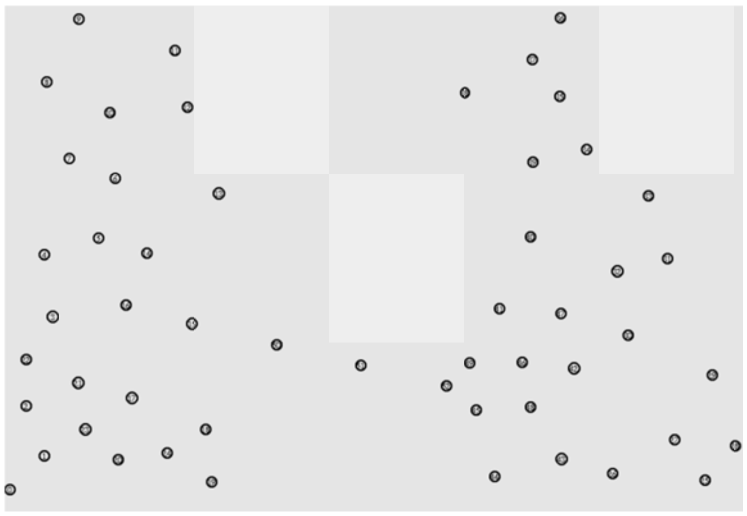


Figure 3.4 Topology of 53 nodes as viewed on NAM animator

In this thesis, the term **"local topology"** or **"topology"** is referred to as the *n* hop neighbor list of a node. For example, the one hop neighbor list of a node represents all the nodes which are within the transmission range (generally 250m in common MANET scenarios). In other words, two nodes are considered as neighbors only when they exchange a HELLO and a corresponding HELLO$_{rep}$ between them. The geographic positioning of the nodes has not been considered in this research project. It is assumed that under normal circumstances, with 250m transmission radius, an *n* hop neighbor must be within *n* × 250m away from the source. However, one of the devastating scenarios where violation of this nature of *n* hop neighbors occurs is in the presence of a wormhole tunnel. Although there are some other factors such as congestion, queuing delay, or intra nodal processing speed which can also increase the RTT between two nodes but the distortion of topology only occurs in presence of a wormhole tunnel (traditional or Byzantine). In the following section the effect of wormhole tunnel on the topological view is presented.

3.2.2 Effects of Wormhole on Topology

The attackers create the wormhole illusion by tunneling HELLO packets between two remote nodes. As a result, two remote nodes consider themselves as direct neighbors. This false link information is propagated to other nodes across the network via TC messages (in OLSR) or RREP (in AODV). The result is the creation of two routing “black holes”, one at each endpoint of the tunnel [48]. The term “black hole” is defined in [48] as the ability of the tunnel endpoints to attract traffic.

For the success of the wormhole tunnel, the attackers make sure that the tunnel does not *collapse*. A wormhole tunnel collapses when its tunnel endpoints fail to forward packets between remote network regions. A solution to the *wormhole collapse* problem is using intermediate colluders to relay packets between the tunnel endpoints. In presence of intermediate relay nodes, the tunnel endpoints are able to communicate persistently during the attack. Using multiple intermediate colluders may provide additional resilience to topology changes and a potentially stealthier wormhole attack [48]. In this study, it is assumed that there is at least one colluder in between the tunnel endpoints.

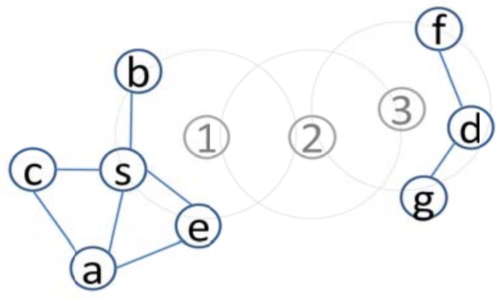


Figure 3.5 Two-hop wormhole attack scenario

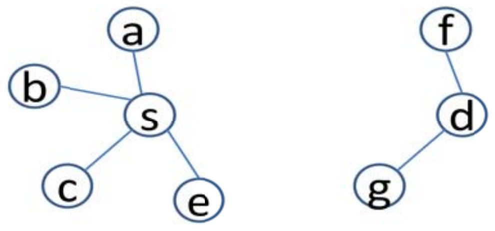


Figure 3.6 Actual *one hop* topology of node *s* and node *d*

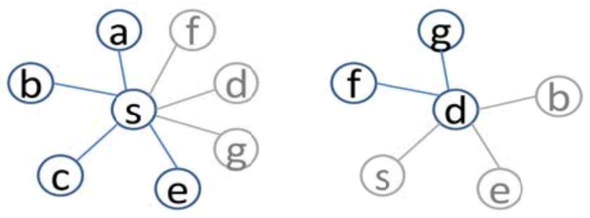


Figure 3.7 Distorted topological view in presence of wormhole

In Fig. 3.5, a two hop wormhole attack scenario is presented. Initially, node s does not have any known route to d and therefore, it starts a route discovery process looking to find a path to d. The RREQs generated from node s is tunneled by the attacker nodes *1*, *2* and *3*, and then rebroadcasted in the region where *d* is placed. Node *d* sends back RREP to node s when it receives the RREQ. Since the attackers do not disclose their identities, both s and d finds themselves as direct neighbors. Consequently, they start sending data packets which travel through the wormhole tunnel.

Furthermore, when node s broadcasts HELLO packets to find its *one hop* neighbors, nodes d, *f* and g also receive that because of the tunneling of the wormhole attackers. Hence, nods *s* creates its *one hop* neighbor list which includes a, *b*, *c*, *e*, d, *f* and *g.* Similarly, node *d* enlists the nodes s, *b* and *e* in its *one hop* neighbor list. As shown in Fig. 3.6 and Fig. 3.7, it is clear that in the presence of wormhole tunnel, the topological view of the network is distorted.

3.3 Detection of Traditional Wormhole Attacks in MANET

In this section, a detection method is presented which uses round trip time (RTT) measurement and topological comparison for detecting traditional wormhole attacks in MANET. The proposed scheme is based on the following observations on wormhole attack:

- Two fake neighbors (e.g., s and d as in Fig. 3.5), with a wormhole tunnel in between, usually experience longer RTT compared to the RTT between two real neighbors (e.g., s and *b* as in Fig. 3.5). Two nodes suspect each other to be fake neighbor when the following inequality holds:

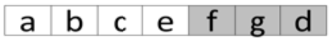
$$RTT > k \times RTT_{avg} \tag{3.13}$$

In equation 3.13, ***k*** is the *delay factor* which depends on the length of the tunnel, and RTT_{avg} is the average RTT between a node and all its real neighbors. In this wormhole detection method the value of *k* is considered to be 3, because it is assumed that two fake neighbors are at least 3 hops away from each other. Equation 3.12 and equation 3.13 are similar, but they have different applications. The earlier is used for calculating the maximum time a node should wait for receiving $\text{HELLO}_{\text{rep}}$s from its neighbors; however, the latter is used for calculating round trip time between a source and its *k* hop neighbours.

- In most cases, two real neighbors have at least one common real neighbor between them, but usually it is not true for two fake neighbors. The probability of not having a common neighbor is shown in Appendix 2. In Fig. 3.5, nodes *s* and *d* are deceived by the wormhole attackers. So, they believe each other as direct neighbors even though they don't have any common real neighbors.

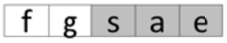
3.3.1 Neighbor List

Each node in the network maintains a ***Neighbor List*** and an associated RTT_{avg} of ***one hop*** neighbors. The ***Neighbor List*** population process is initiated by broadcasting HELLO packet in the network. Two nodes are considered as neighbors when they exchange HELLO and $\text{HELLO}_{\text{rep}}$ between them. Since wormhole tunnel induces packet latency, the $\text{HELLO}_{\text{rep}}$ from a fake neighbor reaches the source much later than that from a real neighbor. This ***packet latency*** is used as a metric to separate the ***Neighbor List*** into two segments: ***Trusted*** (*TRST*) and ***Suspected*** (SUS). If the RTT between the source and a neighbor is more than ***k*** times the current RTT_{avg}, the neighbor is placed into the ***Suspected*** (*SUS*) segment. It should be noted that in this wormhole detection method both RTT and RTT_{avg} are measured locally.



a	b	c	e	f	g	d

Figure 3.8 Complete neighbor list of s



f	g	s	a	e

Figure 3.9 Complete neighbor list of d

Referring again to the example provided in Fig. 3.5, node *s* broadcasts HELLO packet in the network. Node *a*, *b*, *c* and *e* reply with $HELLO_{rep}$ to node *s*. In presence of wormhole tunnel (*1*-*2*-*3*), HELLO from node *s* is rebroadcasted at the other part of the network. Hence, the nodes (*d*, *f* and *g*) within the transmission range of the attacker node *3* also receive the HELLO from node *s* and reply with $HELLO_{rep}$. Their replies arrive at node *s* significantly later than the replies from *a*, *b*, *c* and *e*. Therefore, node *s* can suspect *d*, *f* and *g* to be fake neighbors (and therefore may be attacked by a wormhole). In Fig. 3.8 and Fig. 3.9, complete ***Neighbor List***s of nodes *s* and *d* are shown where the ***Suspected*** parts are shaded gray. However, the ***Neighbor List*** of a node may not necessarily always reflect the actual situation. For example, if the link $s \rightarrow a$ is congested and the response from node *a* reaches node *s* significantly late, then node *a* would be treated as suspected neighbor. This situation is not desirable and can be overcome by exchanging topological information between nod *s* and node *a*.

3.3.2 Topological Comparison Algorithm

Since the nodes attracted by a wormhole tunnel get a distorted view of the network topology, two far away nodes consider each other as direct neighbor. In Fig. 3.5, for example, nodes *s* and *d* consider themselves as *one hop* neighbors, even though they are *four hops* away from each other.

However, when they compare the *TRST* part of their respective ***Neighbor List***s, they find nothing in common. So, topological comparison gives those real neighbors, which are included in the *SUS* part of a ***Neighbor List***, a second chance to prove their reliability. The wormhole detection method presented here is triggered when a source finds non empty *SUS* part in the ***Neighbor List***. Two more packet types are used for doing the topological comparison: ENQ and ENQ_{rep}. The steps of the topological comparison phase are presented below:

- After the neighbor discovery is done a node sends ENQ packets to the suspected neighbors (***Suspected*** part of the ***Neighbor List***).
- In response to ENQ packet, a node sends back ENQ_{rep}, which includes its own TRST list.
- The source node compares the received TRST list with its own TRST list. If a node is found to be attacked by a wormhole, its ID is saved in another list DET. Otherwise, the node is deleted from SUS list and included in the TRST list of the source.

The outcome of the wormhole detection method depends on the values of ***me***, ***suspect*** and ***trust***. There are four decisive states of the detection algorithm:

1) $(\boldsymbol{me = FALSE, suspect \geq trust})$**:** When s is not included in $TRST_r$, and the number of elements in $TRST_r \cap SUS_s$ is at least equal to the number of elements in $TRST_r \cap TRST_s$. As a result, node *s* decides that the link *s*→*r* is attacked by wormhole and *r* is inserted in the DET_s.
2) $(\boldsymbol{me = FALSE, trust} > suspect)$**:** When s is not included in $TRST_r$, and the number of elements in $TRST_r \cap TRST_s$ is more than the number of elements in $TRST_r \cap SUS_s$. As a result, node *s* decides that the link *s*→*r* is safe and *r* is deleted from the SUS_s and inserted in to $TRST_r$.
3) $(\boldsymbol{me = TRUE, trust \geq 0}\ \&\ suspect = 0)$**:** When $s \in TRST_r$ and $TRST_r \cap TRST_s$ can be empty but $TRST_r \cap SUS_s$ must be empty.

As a result, node s decides that the link $s \rightarrow r$ is safe and r is deleted from the SUS_s and inserted in to $TRST_r$.

4) $(\boldsymbol{me = TRUE, trust \geq 0}\ \&\ suspect = 0)$**:** When $s \in TRST_r$ and $TRST_r \cap TRST_s$ is empty and $TRST_r \cap SUS_s$ is non empty. As a result, node s decides that the link $s \rightarrow r$ is attacked by wormhole and r is inserted into DET.

3.3.3 System Reactions When a Wormhole Attack is Detected

The originator of the detection process reacts to minimize the damage by halting any communication with a wormhole attacked node. For instance, if the link $s \rightarrow r$ is attacked by wormhole, the originator (s) stops sending packets to node r. Node s can also inform its trusted one hop neighbors about node r being attacked, so that, they can also react by halting any communication with r.

3.3.4 Security of the Detection Method

The topological comparison based detection method can itself be vulnerable to modification attacks. For example, the wormhole attackers (tunnel endpoints) can send TRST and SUS lists of their own. In other words, they can alter a TRUST or SUS list propagating through the tunnel. So, it is important that the contents of unicast packets like ENQ and ENQ_{rep} are protected. Unfortunately, designing authentication and cryptographic schemes is outside the scope of this study. So, we propose to use the security measures presented by Y. Hu et al. in [27] to further improve the security of the network.

3.4 Performance Evaluation Using ns-2 Simulator

3.4.1 Network Simulator (ns-2)

Network Simulator (also popularly called ns-2) is a "discrete event" simulator and is heavily used in ad hoc networking research. It provides necessary support for simulating wired and wireless networks. The *ns-2* simulator is coded in two languages: C++ and OTcl. Simulation objects are mirrored in both realms—that means that if one defines a node and some variables associated with a node, the node variables are accessible from code in either language. The intent of this design is to put computationally intensive code in a compiled language (C++), where it can execute fast, while allowing the user to configure the simulator in a more user-friendly scripting language-- in this case, OTcl, or object-oriented Tcl.

3.4.2 Generating Topologies in ns-2

Network scenarios have been generated using Tcl scripts. To evaluate the performance of the wormhole detection method, topologies are generated dynamically using random number generator to place nodes within an area ranging from 800m × 800m to 1400m × 1400m. A minimum distance (ranging from 90m to 200m) is maintained between each node in the network. The wormhole attackers are also placed randomly in between two randomly selected (target) nodes in different network segments. The distance between each wormhole attacker is varied on the basis of network area and number of nodes. Another exciting feature of randomly generated scenarios is that the topology changes for every simulation run. Hence, the performance evaluation of the wormhole detection method presented in this study is reliable.

3.4.3 Simulating Traditional Wormhole Attack

In the traditional wormhole attack, two colluding nodes tunnel packets from their vicinity and attract as many nodes as possible. In between the colluders there are relay nodes placed to increase the length of the tunnel. The relay nodes are responsible for forwarding packets through the tunnel. It should be noted that both the attackers and the relay nodes remain silent to other participating nodes. This means that they neither participate in any network operations, such as routing, nor respond to neighbor discovery or topological comparison packets. They hide their identities by encapsulating the targeted packets.

Two new application layer agents have been designed in ns-2 to simulate the wormhole attack. Their functionalities are described below:

- ***myAgent:*** This agent is used to allocate HELLO, $\text{HELLO}_{\text{rep}}$, ENQ and ENQ_{rep} packets. Moreover, the topological comparison between a source node and a suspected neighbor is also conducted by this agent. It is attached to every honest node as well as the attackers (excluding the relay attackers). This agent also stores the outcome of the wormhole detection method as a list (DET) of neighbors which are attacked by a wormhole.
- ***Tunnel:*** The Tunnel agent is attached to only the wormhole attackers which are responsible for tunneling packets from one location to the other. So, both ***myAgent*** and ***Tunnel*** agents are attached to the two endpoints of the wormhole tunnel. This agent is responsible for encapsulating the targeted packets to another packet type (TUNNEL), which is only transmitted through the tunnel. When an encapsulated packet reaches an endpoint, the TUNNEL type packet is de-capsulated to obtain the original packet.

3.4.4 Performance Metrics

The performance of the wormhole detection method presented in this chapter is measured in regards to the following two metrics:

- ***Detection rate:*** The term ***"Detection rate"*** takes into account the number of nodes that are possibly attacked by a wormhole and how many of them are successfully detected. The following formula is used to determine the *detection rate*:

$$Detection\ Rate = \frac{Total\ no.of\ wormhole\ links\ detected}{Total\ no.of\ wormhole\ links} \tag{3.14}$$

- ***Accuracy of alarms:*** The ***accuracy of alarm*** represents the efficiency of the wormhole detection method when it detects possible attacks by using topological comparison. It takes into account the number of links declared as attacked by a wormhole and how many of them are actually affected. The following formula is used to determine the ***accuracy of alarm***:

$$Accuracy\ of\ alarms = \frac{Total\ no.of\ wormhole\ links\ detected}{Total\ no.of\ alarm} \tag{3.15}$$

3.5 Wormhole Detection in AODV Routing Protocol

In AODV routing, the wormhole attackers tunnel the RREQs from one part of the network to another part. For example, in Fig. 3.5, the RREQ from node *s* is recorded by the attacker node *1* and then tunneled to the other attacker node *3,* which rebroadcasts the same RREQ to its vicinity. As a result the route *s→1→2→3→d* is selected as the path for communication between *s* and *d*. The wormhole detection method presented in the previous section works in AODV routing as follows:

- When a source s wants to send data packets to a destination *d* but does not have a route to *d*, node s broadcasts RREQ in the network. The broadcast time is recorded as T_{rreq}.
- Each node calculates a ***response timeout*** suggesting a time interval before which the RREP from a *one hop* neighbor should reach. The ***response timeout*** is calculated using the following formula:

$$response\ timeout = \frac{2R}{C} + 4 \times \Delta \qquad (3.16)$$

In equation 3.16, R is the transmission radius (usually 250m) of a node, C is the speed of light (the maximum speed a packet can travel) and Δ is the time spent in the protocol layers. Usually, Δ represents the time taken by a packet to travel from the application layer to the physical layer and vice versa.

- When an RREP for a destination *d* is received at node s, it calculates the RREP ***response time*** as follows:

$$response\ time = T_{clock} - T_{rreq} \qquad (3.17)$$

- If the RREP suggests that the intended destination is *one hop* away and the ***response time*** is less than the ***response timeout***, node *s* starts sending data packets to node *d*. If the ***response time*** is greater than the ***response timeout***, node *s* triggers the wormhole detection method.

- At this stage, the source node *s* compares its own *one hop* neighbor list with the *one hop* neighbor list of node *d*.

In this section the performance of the wormhole detection method is evaluated with AODV routing protocol. The implementation of the detection has been presented in section 3.4. In addition, the performance the method presented in this thesis is compared with the method proposed by Z. Tun et al. [33], T. V. Phuong et al. [35] and the RTT-only phase (not executing the topological comparison phase) of the proposed detection method. The results show that both high detection rate and accuracy of alarms can be achieved with topological comparison based approach. This is because the suspected nodes get a second chance to justify their credibility by exchanging relative positioning with the source.

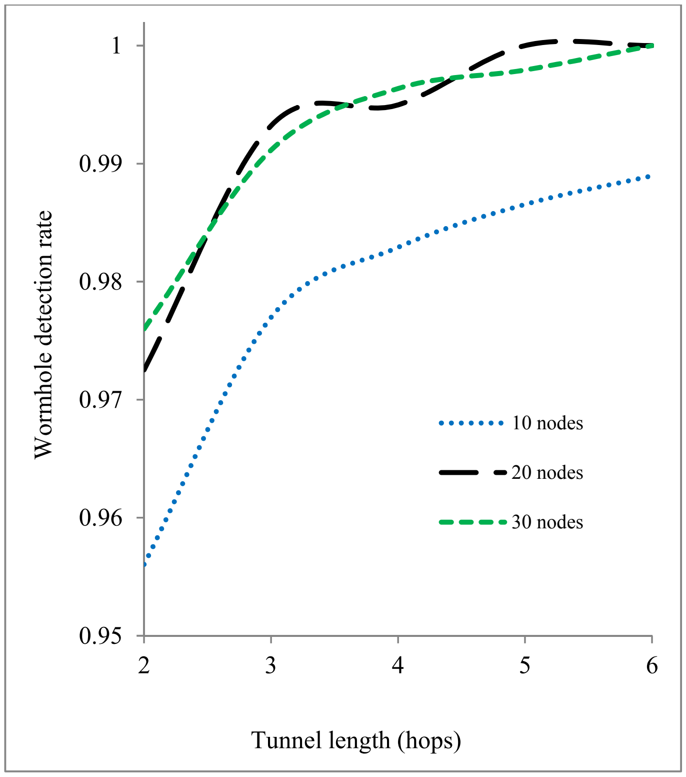


Figure 3.10 Tunnel length vs. Wormhole detection rate

Fig. 3.10 shows the *detection rate* versus *tunnel length* for different network sizes ranging from 10 nodes to 30 nodes. It can be seen that the *detection rate* of the topological comparison based wormhole detection approach shows an increasing trend as the length of the wormhole tunnel is increased. This is because that with longer tunnel length the probability of the actually attacked neighbors being included in the ***Suspected*** part of the source's ***Neighbor List*** is almost certain due to the long RTT between them. In addition, with larger network sizes more genuine neighbors are likely to be removed from the ***Suspected*** list and thus increases the *detection rate*. The *detection rate* curves are almost identical for larger network sizes because the rate of change in network size is much higher than the rate of change in number of *one hop* neighbors.

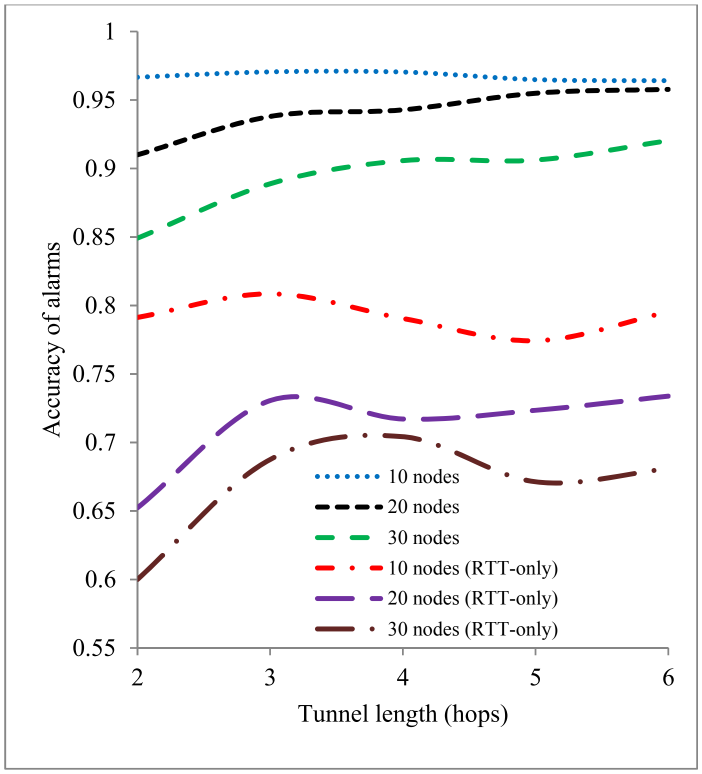


Figure 3.11 Tunnel length vs. Accuracy of alarms

In Fig. 3.11, the *accuracy of alarm* chart is shown as a function of the tunnel length. It can be seen that for longer tunnel length, higher accuracy can be achieved. It is because that, with longer tunnel length, the RTT between a pair of fake neighbors is longer and thus less genuine neighbors to be included in a ***Suspected*** list. However, we can see a little dip in the accuracy of alarm when the tunnel length is 4-5 and rise again. This may be due to simulation randomness. To demonstrate the effectiveness of the topological comparison, we compare our scheme with RTT-only (i.e., not executing neighbor list comparison). Fig. 3.11 also shows the *accuracy of alarms* of RTT-only versus tunnel length for different network sizes. It can be seen that our scheme achieves much higher *accuracy of alarms* as the topological comparison removes many genuine neighbors from the suspected neighbor list.

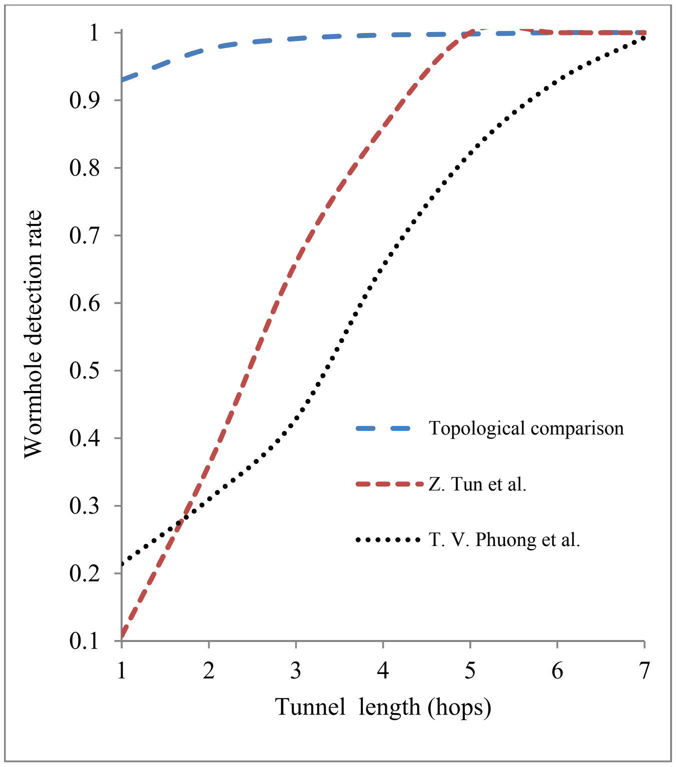


Figure 3.12 Comparison with existing detection methods (30 nodes)

Figure 3.12 shows a performance comparison between the topological comparison based approach and the RTT based methods presented by Z. Tun et al. in [33] and T. V. Phuong et al. in [35]. The topological comparison based approach performs much better when the tunnel length is smaller (e.g., less than 5 hops). As the authors in [33] considers the long RTT between two fake neighbors and the number of neighbors, for smaller tunnels it becomes difficult for this approach to identify the real neighbors from a list of suspected neighbors. However, for larger tunnel lengths their *detection rate* is identical because the possibility of real neighbors to be included in the ***Suspected*** list is small. On the other hand, the authors in [35] detects wormhole attacks based on the transmission time between every two successive nodes along the established path. So, when the tunnel length is small, the transmission time between legitimate nodes and the wormhole attackers are almost identical which makes the detection difficult.

3.6 Wormhole Detection in OLSR Routing Protocol

In OLSR routing, each node periodically exchanges link-state messages, such as HELLO and Topology Control (TC). It also uses a multipoint relaying (MPR) strategy, which minimizes the size of the control messages and the number of rebroadcasting nodes. In wormhole attack scenario, an attacker encapsulates the HELLO messages from its vicinity and tunnels them to another attacker. The colluding attacker de-capsulate the tunneled message and then rebroadcast the same HELLO message to its vicinity. For example, in Fig. 3.5, the HELLO messages from nodes *s*, *b* and *e* are tunneled by the wormhole attacker node *1* to the colluder node *3* via the relay node *2*. Eventually, nodes *d*, *f* and *g* receive the same HELLO messages. As a result, nodes *s*, *b* or *e* will choose *d*, *f* or *g* as MPR and vice versa. This leads to exchange of some Topology Control (TC) packets through the wormhole tunnel (*1-2-3*). Since only the MPR nodes are responsible for forwarding TC packets, selecting MPRs that possess flawed network topology may lead to routing disruption and ultimately result in performance degradation of the network as a hole.

The interval between two successive HELLO messages is predefined. We propose that after *n* number of HELLO transmissions, a node sends $\text{HELLO}_{\text{det}}$ which represents the HELLO for *one hop* neighbor discovery phase of the wormhole detection method. The subsequent phases of the wormhole detection process are as mentioned in section 3.3. The number *n* depends on the desired security level. The performance evaluation of the topological comparison based wormhole detection method is presented in Fig. 3.13 and Fig. 3.14. In Fig. 3.13, the topological based approach is compared with the method proposed by the authors in [34]. The graph shows that the *detection rate* for the topological comparison based method is significantly higher than the other method. In addition, higher *accuracy of alarm* can also be achieved as shown in Figure 3.14.

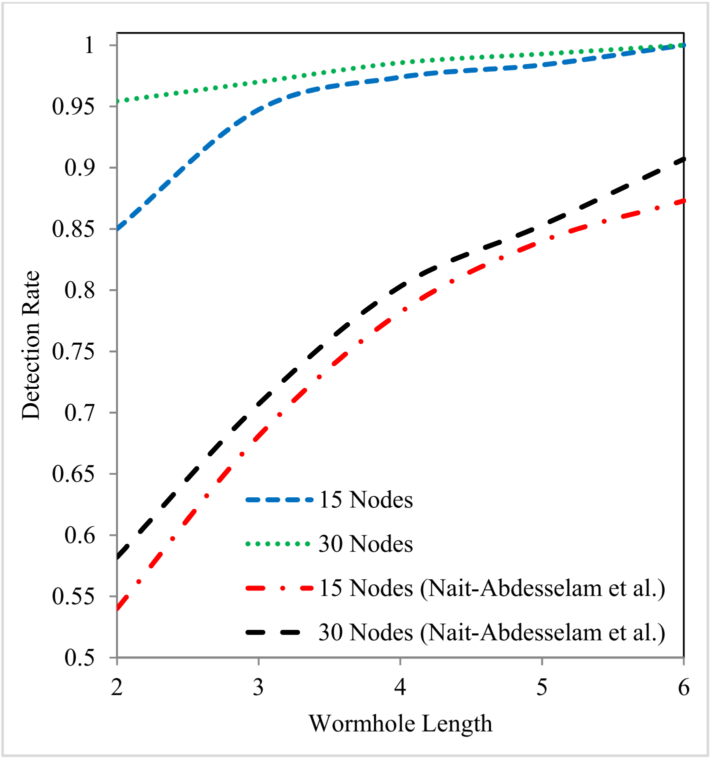


Figure 3.13 Wormhole length vs. Detection rate

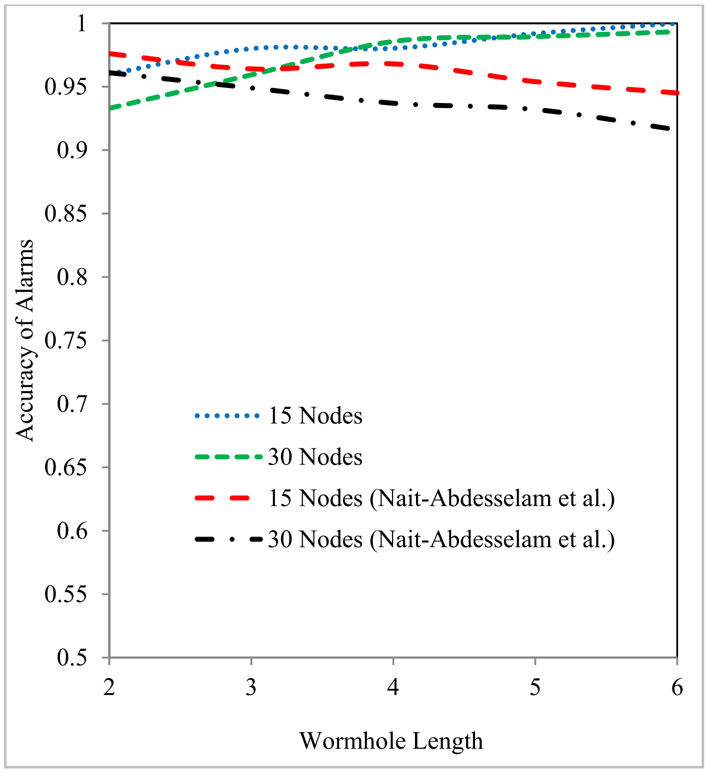


Figure 3.14 Wormhole length vs. Accuracy of alarm

3.7 Message Overhead and Detection Time

The topological comparison based method gives the initially suspected nodes a second a chance to verify the reliability. However, this approach introduces some message overhead to the system. In the wormhole detection method proposed in this thesis, we use HELLO, HELLO$_{rep}$, ENQ, ENQ$_{rep}$ messages. Since our proposed wormhole detection method achieves very high detection rate and thus ensures security to the system, the associated message overhead can be tolarated.

We have also studied the detection time for the topological comparison based detection method. We have assumed that a suspected node knows the one hop neighborhood information and responds immediately after reciving ENQ message. Therefore, the delay components considered are:

- The time elapsed between when the originator broadcasts HELLO to the network and receives a HELLO$_{rep}$ from a suspected node;

- The time elapsed between when the originator sends ENQ message and the time and when it receives corresponding EN_{Qrep};
- The running time of the topological comparison algorithm.

In Fig 3.15, the detction time for different tunnel lengths is shown. Since the HELLO, $HELLO_{rep}$,ENQ and ENQ_{rep} messages travel through the tunnel, the detection time increases with the tunnel length.

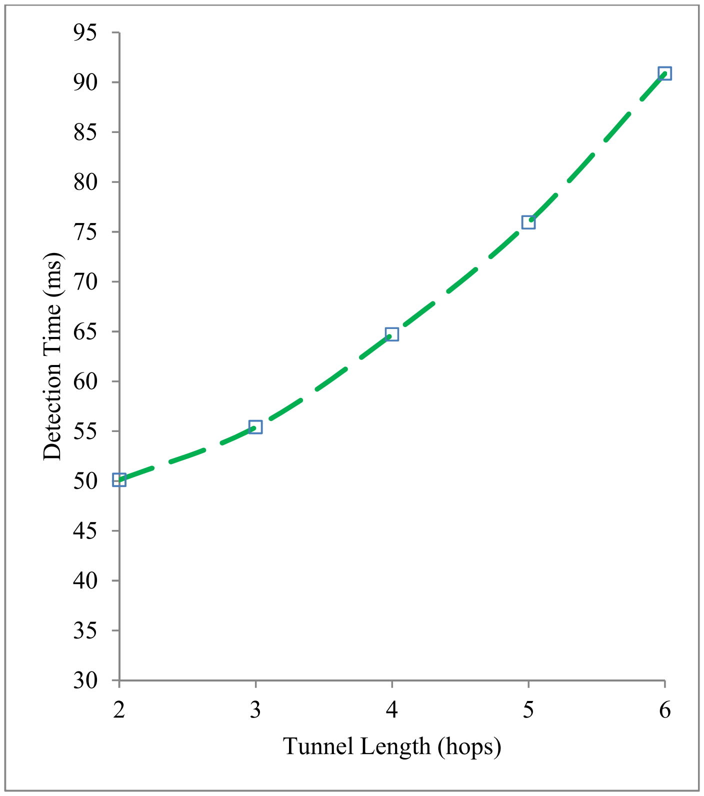


Figure 3.15 Tunnel Length vs. Detection Time

3.8 Summary

In this chapter, the topological comparison based approach of detecting traditional wormhole attack has been presented. The performance evaluation is done in AODV and OLSR routing protocol. Extensive computer simulations using the network simulator (ns-2) tool have been carried out to simulate different network scenarios, involving different tunnel lengths and network sizes. Comparison with some of the existing detection methods and with the RTT-only phase of the proposed method has been presented in terms of detection rate and accuracy of alarm. It is found that the topological comparison based approach achieves higher detection rate as well as higher accuracy than other RTT based detection methods.

4. Detecting Byzantine Wormhole Attack Using Topological Comparison and RTT

In a Byzantine wormhole scenario, the endpoints of the tunnel actively participate in the network operations (e.g., routing and security). In fact, they are authenticated nodes who collude to launch an attack. The attackers can record, modify (since they are authenticated nodes) or drop data packets passing through the tunnel. In the worst case scenario, the attackers can cause a denial of service (DoS) attack. This chapter presents a topological comparison based method for detecting Byzantine wormhole attack in MANET.

4.1 Effects of Byzantine Wormhole on Topology

A Byzantine wormhole tunnel is formed by at least two authenticated nodes that record packets at one location, and tunnel them to another location of the network. As a result, they may inject themselves in between two honest nodes. Furthermore, the attackers can record the data packets, transmitted through the tunnel for later analysis. Byzantine wormholes are more sophisticated than traditional wormhole because of the attackers' active participation in the network. They are capable of behaving like an honest node and carry on an attack simultaneously. Hence, Byzantine wormhole attack can be classified as an ***active internal attack***. In this section, the impact of Byzantine wormholes on the network topology is presented in details.

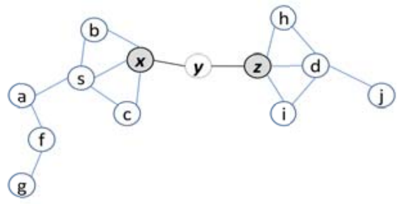


Figure 4.1 Two hop Byzantine wormhole attack scenario

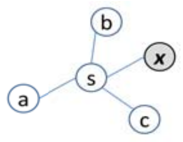


Figure 4.2 One hop neighbors of *S*

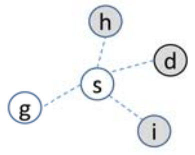


Figure 4.3 Three hop neighbors of *S*

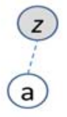


Figure 4.4 Three hop neighbors of node a

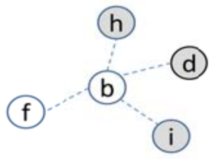


Figure 4.5 Three hop neighbors of node b

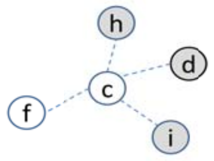


Figure 4.6 Three hop neighbors of node c

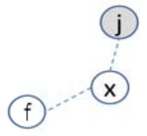


Figure 4.7 Three hop neighbors of node x

In Fig. 4.1, a two hop Byzantine wormhole attack scenario is presented, where node *x*, *y* and *z* are authenticated nodes that collude to perform the attack. Node *y* is responsible for forwarding encapsulated packets between the colluders. So, node *y* can be defined as a relay node. Unlike traditional wormhole attack, the attackers respond to neighbor discovery packets (HELLO). If a HELLO for *one hop* neighbor discovery (***hops_to_leave*** = 1) is received by an attacker, it replies with HELLO_{rep} back to the source. Otherwise, if the HELLO is for *n* hop neighbors discovery (***hops_to_leave*** = *n*), it decrements the ***hops_to_leave*** field, encapsulates the HELLO packet, and tunnels it towards the other colluder. However, the relay node does not decrement the ***hops_to_leave*** to ensure that the tunnel can attract traffic towards it by offering better path in terms of hop count.

For example, according to Fig. 4.1, the source (*s*) may choose to send the data packets to *d* the path $s \rightarrow x \rightarrow (y) \rightarrow z \rightarrow d$. Although there are *four hops* (physical topology) between them, both node *s* and *d* believe that they are *three hops* (logical topology) away. Figure 4.2 shows the *one hop* topology of node *s*. It can be seen that the target node *s* includes an attacker (*x*) in its *one hop* topology. The *three hop* topology of node *s* is shown in Fig. 4.3. The nodes that are shaded gray, in Fig. 4.3, are suspected to be attacked by wormhole. In Fig. 4.4, 4.5, 4.6 and 4.7, the *three hop* physical topologies of node a, b, c and x are shown respectively. It should be noted that a, b, c and x are the *one hop* neighbors of node *s*.

So, the targets (placed at different network regions) of a Byzantine wormhole tunnel find themselves *three hops* away from each other. In addition, a target's suspected *three hop* neighbors are also suspected by its *one hop* neighbors. These effects of Byzantine wormhole tunnel are considered for designing a topological comparison based detection method.

4.2 Detection of Byzantine Wormhole Attacks in MANET

In this section, a Byzantine wormhole detection method is presented. The proposed detection method makes use of both *one hop* and *three hop* topologies. It is based on the following observations/assumptions:

- Two nodes, targeted by Byzantine wormhole attackers, find themselves *three hops* away from each other, regardless of the tunnel length. This is because of the assumption that the intermediate relay nodes do not change the hop count.

- A link containing Byzantine wormhole tunnel offers shorter paths in terms of hop count so that a lot of traffic can be attracted.

- The RTT between two fake *three hop* neighbors is greater than *k* times the RTT_{avg} for *one hop* neighbors. However, some real *three hop* neighbors may be suspected because of some common delay factors (e.g., congestion, queuing delay etc.) as stated earlier in Chapter 3.

- A node can discover its *n* hop neighborhood information by broadcasting HELLO packet with the ***hops_to_leave*** field set to *n*. A recipient of HELLO either replies with $HELLO_{rep}$ in case ***hops_to_leave*** equals 1, or rebroadcast the same HELLO after decrementing ***hops_to_leave***.

- A *three hop* neighbor list has two parts: ***Trusted_three_hop*** and ***Suspected_three_hop***. Unlike the traditional wormhole detection, a *one hop* neighbor list does not have trusted or suspected segment. This is because, Byzantine wormhole attacks are detected by comparing *three hop* topology (not *one hop* topology) of a source with that of its *one hop* neighbors.

- A node (*s*) may suspect that a Byzantine wormhole tunnel exists between itself and its *three hop* neighbor (*x*) if the following statement is true:

  "Node *x* is included in node *s*'s suspected *three hop* neighbor list and node *x* is not considered as a trusted *three hop* neighbor by any of node *s*'s *one hop* neighbor."

In the following subsections, we first explain how the *one hop* and *three hop* neighbor lists are generated and then we explain how topological comparison is applied to detect Byzantine wormhole attack.

4.2.1 One Hop Neighbor List

Each node maintains a list (***N***) of *one hop* neighbors and an exponentially weighted moving average of RTT (RTT_{avg}) associated to it. The traditional wormhole detection algorithm, presented in Chapter 3, can be used as a module here to prevent any hidden attackers (traditional wormhole) from tunneling the HELLO. The following figure (Fig. 4.8) shows the *one hop* neighbor list of node *s* as of the topology presented in Fig. 4.1.

a	b	c	x

Figure 4.8 One hop neighbor list of s

4.2.2 Three Hop Neighbor List

The *three hop* neighbor discovery process is initiated by broadcasting HELLO packet with ***hops_to_leave*** set to 3. Each recipient of the HELLO packet either responds with a unicast $HELLO_{rep}$ or rebroadcasts the same HELLO based on the value of ***hops_to_leave***. The following condition is used for dividing the *three hop* neighbor list into two parts:

$$\mathrm{RTT}_{3\_\mathrm{hop}} > k \times RTT_{avg(one\ hop)} \tag{4.1}$$

A node is placed in the ***Suspected_three_hop*** part if the condition holds; otherwise, it is placed in the ***Trusted_three_hop*** part. In this approach, we assume that the value of k is 3.

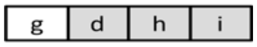


Figure 4.9 Three hop neighbors list of s

```
If hops_to_leave > 1 Then
       //check if already received the same packet
       If (!search(sender_ID)) Then
            Broadcast_ID = sender_ID;
            //Decrement the hops_to_leave count
            hops_to_leave--;
            Rebroadcast the HELLO;
       Else
              Drop HELLO;
       End If
Else If hops_to_leave = 1 Then
       //unicast HELLO$_{rep}$ to the broadcaster
       Send HELLO$_{rep}$ to Broadcast_ID;
End If
```

Algorithm 1 Broadcast of HELLO to explore *three hop* neighbors

In Algorithm 1, a controlled broadcasting technique is used where repetitive HELLO packets from the same sender are dropped. When a sender broadcasts a HELLO packet it records the time of broadcast so that RTT can be calculated upon receiving HELLO$_{rep}$. In Fig. 4.9, the complete *three hop* neighbor list of node *s* is shown where the suspected nodes are shaded gray.

4.2.3 Topological Comparison Algorithm

In Byzantine wormhole detection method, topological comparison is done between an originator and its *one hop* neighbors. The topological comparison algorithm is executed when the originator finds it has a non empty ***Suspected_three_hop*** list. The steps of the topological comparison algorithm are presented below:

- After the neighbor discovery phase, the originator sends ENQ packets to its *one hop* neighbors. This probes the *one hop* neighbors to send back their complete *three hop* neighbor lists.
- A recipient of ENQ packet replies with ENQ_{rep} by attaching its complete *three hop* neighbor list (***Trusted_three_hop*** and ***Suspected_three_hop***).
- Upon receiving an ENQ_{rep} packet, the originator compares its own suspected *three hop* neighbor list (***Suspected_three_hop***) with the trusted *three hop* neighbor lists (***Trusted_three_hop***) of the *one hop* neighbors.
- The originator also maintains a list of attacked nodes in DET. A node (x) is declared as attacked by Byzantine wormhole and therefore its ID is inserted into DET if the following conditions are true for node x:
    - Node x is included in the suspected *three hop* neighbor list of the originator.
    - Node x is not considered as a trusted *three hop* neighbor by any of node s's *one hop* neighbor.
- Due to a large RTT_{avg} of the *one hop* neighbors, the originator may initially trust a fake *three hop* neighbor (x). To avoid such cases the originator compares its own trusted *three hop* neighbor list (***Trusted_three_hop***) with the suspected *three hop* neighbor list (***Suspected_three_hop***) of its *one hop* neighbor. A node (x) is eliminated from the originator's trusted *three hop* neighbor list and

therefore declared as attacked by Byzantine wormhole tunnel if the following conditions are satisfied for node *x*:

- Node *x* is included in the trusted *three hop* neighbor list of the originator.
- Node *x* is considered as a suspected *three hop* neighbor by at least two *one hop* neighbors of the originator.

```
//send ENQ packets to each one hop neighbor
For all $x \in N$
     Send ENQ to x;
End for
//processing the ENQ$_{rep}$ packet
If nonempty ( $N$)
           and $Trusted\_three\_hop_x \cap Suspected\_three\_hop_s \neq \emptyset$ Then
     Delete m from $Suspected\_three\_hop_s$,
     where $m \in Trusted\_three\_hop_x \cap Suspected\_three\_hop_s$;
     Insert m into $Trusted\_three\_hop_s$;
End If
If $Suspected\_three\_hop_x \cap Trusted\_three\_hop_s \neq \emptyset$ Then
     Counter++;
     If counter > 1 Then
            Delete m from $Trusted\_three\_hop_s$,
            where $\in Suspected\_three\_hop_x \cap Trusted\_three\_hop_s$;
            Insert m into $Suspected\_three\_hop_s$;
     End If
End If
//Transfer the outcome into DET list
DET = $Suspected\_three\_hop$;
```

Algorithm 2 Byzantine wormhole detection

In Algorithm 2, the steps of Byzantine wormhole detection method are presented. The algorithm starts with broadcasting ENQ packets to the *one hop* neighbors and ends up with a DET list which contains all the nodes that are attacked by a Byzantine wormhole tunnel.

4.3 Simulating Byzantine Wormhole

In Byzantine wormhole attack, the attackers as well as the relay nodes are authenticated nodes. So, they participate in the network operations and perform an attack simultaneously. Two new application layer agents have been designed in ns-2 to simulate Byzantine wormhole attack. Their functionalities are described below:

- ***regularAgent:*** This agent is attached to each node and used for allocating **PT_REGULARAGENT** type packets, such as HELLO, $\text{HELLO}_{\text{rep}}$, ENQ and ENQ_{rep}. Both *one hop* and *three hop* neighbor lists are maintained by this agent. Moreover, the topological comparison algorithm and detection of Byzantine tunnels are implemented in ***regularAgent***.
- ***Tunnel:*** This agent does the encapsulation, silent forwarding, decapsulation and rebroadcasting of the **PT_REGULARAGENT** type packets. The ***Tunnel*** agent is utilized by the attackers (including the relay nodes) so that **PT_REGULARAGENT** type packets can be tunneled from one location to the other. Upon receiving a **PT_REGULARAGENT** type packet, an attacker (not the relay nodes) encapsulates it into a **TUNNEL** type packet. The **TUNNEL** type packets can only be processed (forward or decapsulate) either by a relay node or by an attacker. In this way, regular packets from one network location can be tunneled to another location.

In Chapter 3, two methods of measuring RTT between a pair of nodes is discussed. Modification of the MAC layer is also necessary if clocks are synchronized. Two additional fields (***prop_t*** and ***clock***) are used in **PT_REGULARAGENT** type packets for doing RTT measurement in a synchronized system. The values of ***prop_t*** and ***clock*** are updated in the MAC layer of a node so that upper layer delays are avoided.

4.4 Byzantine Wormhole Detection in AODV Routing Protocol

The following steps are followed to implement topological comparison based Byzantine wormhole detection method in AODV routing protocol:

- Two additional headers, represented by ***hop_count*** and ***alert_on***, are used in RREP packet. The ***hop_count*** field tracks the number of hops the RREP packet has traveled. On the other hand, the ***alert_on*** field is used to alert the source about a possible Byzantine wormhole attack. By default, ***alert_on*** is set to 0 which means that there is no suspicion about a Byzantine wormhole tunnel. However, if a RREP relay node suspects the route is under attack, the ***alert_on*** field is set to 1 before the RREP packet is forwarded.
- An additional type of packet (**CLEAR)** is used to let the source know the outcome of the topological comparison algorithm.
- When a source (*s*) wants to send data packets to a destination (*d*) but does not have a route, it broadcasts RREQ.
- A recipient either rebroadcasts the RREQ or replies with RREP back to the source. A node only sends RREP if it it has a route to the destination.
- The RREP propagates through the reverse path of the RREQ. If the RREP is generated by node *d*, ***hop_count*** is set to 0. Otherwise, ***hop_count*** is set according to the routing table entry for node *d*. Each relay node of the RREP packet increments the *hop_count*.
- When a relay node (*r*) receives an RREP suggesting that the destination (*d*) is at least *three hops* away from it, the RREP is held and the following steps are followed:
  - Node *r* sends HELLO to node *x* which is *three hops* downstream from node *r*.
  - Upon receiving the $\text{HELLO}_{\text{rep}}$ from node *x*, node *r* calculates the RTT and compares it with the RTT_{avg} of *one hop*

neighbors. (It is assumed that each node periodically updates its *one hop* neighbor list and the associated RTT_{avg}.)

- If the RTT comparison suggests that node *x* is a suspected *three hop* neighbor and therefore may be attacked by a Byzantine wormhole tunnel, the topological comparison phase is triggered. At this stage, node *r* releases the RREP packet towards the sender (*s*) with the ***alert_on*** field set to 1.
- The topological comparison phase is initiated by sending ENQ to the *one hop* neighbors. A recipient of ENQ packet calculates the RTT with node *x*. Then the status of node *x* (trusted/suspected) is attached to the ENQ_{rep} packet.
- If ENQ_{rep} from all *one hop* neighbors suggests that node *x* is a suspicious *three_hop* neighbor and therefore may be under attack, node *r* sends a **CLEAR** message to the source indicating that the route to the destination is under Byzantine wormhole attack.

- When the source receives the RREP with ***alert_on*** set to 0, it starts sending data packets to node *d*. However, if the ***alert_on*** field is set to 1 by relay node (*r*), the source halts the data transmission until a **CLEAR** message received from node *r*.
- If the **CLEAR** message indicates that the destination is safe, the source may starts transmitting data packets. On the other hand, if the **CLEAR** message suggests that the destination is under attack, the source stops any further communication until for a predefined time interval.

The performance is evaluated in terms of *detection rate* and *accuracy of alarm* for 30 nodes. Nodes are placed randomly in a 1000m×1000m area, and the attackers are placed in between the target nodes. Finally, the performance of the topological detection approach is compared with the RTT-only phase.

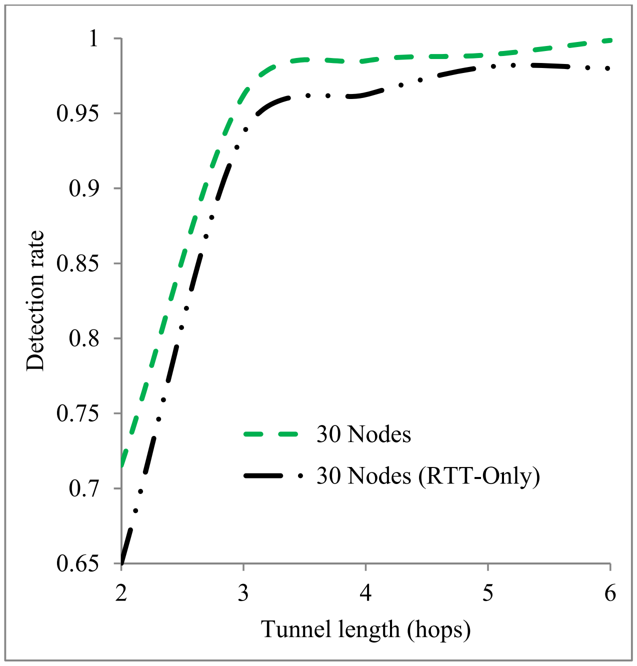


Figure 4.10 Tunnel length vs. Detection rate

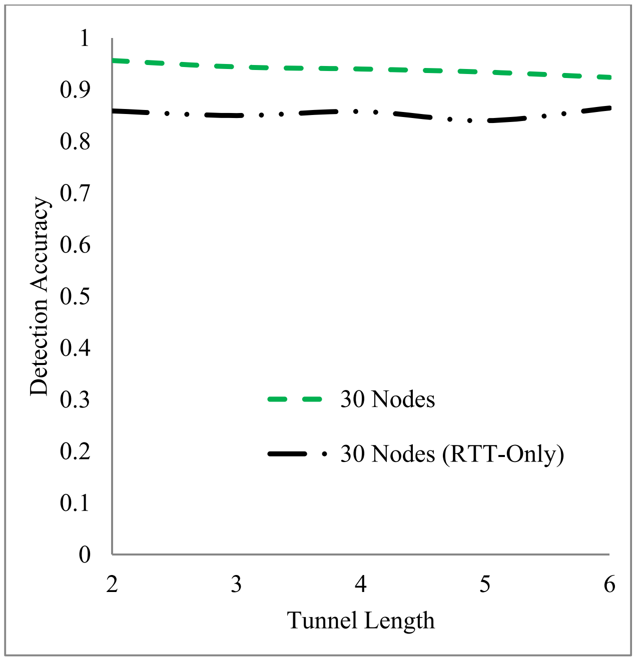


Figure 4.11 Tunnel length vs. Detection accuracy

In Fig. 4.10, the *detection rate* vs. *tunnel length* is presented. The *detection rate* is defined as the proportion of the number of wormhole links detected and the total number of wormhole links. The *detection rate* increases as the tunnel length is increased. It is due to the fact that longer tunnel involves more relay nodes. So, the RTT of wormhole links is longer and easier to be identified. Moreover, when the topological comparison is done it becomes more obvious that a fake *three hop* neighbor will be detected. It can also be seen that the *detection rate* of the topological comparison based approach is higher than the RTT-only approach. This is because the topological comparison based approach gives the suspected neighbors a second chance to prove their reliability. Even if a fake *three hop* neighbor is initially included in the *Trusted_three_hop* list, topological comparison can detect it.

In Fig. 4.11, the *detection accuracy* vs. *tunnel length* is shown. The *detection accuracy* is defined as the proportion of the number of link containing wormhole tunnel and the total number of detection. It can be seen that the *detection accuracy* of the topological comparison based approach is consistently high (around 95%), irrespective of the tunnel length. Moreover, the *detection accuracy* of the RTT-only approach is lower than the topological comparison based approach. It is because that the RTT-only approach detects wormhole links by only considering the round trip time (RTT). As a result, if a real neighbor is suffering congestion, it is also detected as wormhole attacked in RTT-only method.

4.5 Security of the Detection Method

The Byzantine wormhole detection method itself may be a target of modification attack. Therefore, we propose to encrypt the unicast packets like ENQ, ENQ_{rep}, RREP, and CLEAR to minimize the adversary effect of modification attack. As mentioned in Chapter 3, designing authentication and cryptographic schemes is outside the scope of this study. So, we suggest to use the authentication and encryption methods proposed in [27] to further improve the security of the network.

4.6 Message Overhead

In the Byzantine wormhole detection method proposed in this thesis, we use HELLO, $HELLO_{rep}$, ENQ, ENQ_{rep} and CLEAR messages. Although this incurs some message overhead to the system, it can be tolearted by considering the performance of the detection method and the range of difficulties the attacker may impose to the system.

4.7 Summary

The impact of Byzantine tunnels on the topology of a network has been discussed in this chapter. Besides, a topological comparison based approach of detecting Byzantine wormhole attack has been presented. We then incorporated our Byzantine wormhole detection method into AODV routing protocol. Computer simulations have been carried out using the network simulator (ns-2) tool. The simulation results show that the topological comparison based approach can achieve both high detection rate and high accuracy of alarms.

5. Conclusion and Future Work

5.1 A Review of the Research Contributions

The aim of this thesis is to develop topological comparison based detection methods for both traditional and Byzantine wormhole attacks. The major contributions of this research can be listed as follows:

- A new RTT measurement scheme has been developed to measure RTTs between a node and all its *n* hop neighbors. The aim of this RTT measurement scheme is to create a suspected *n* hop neighbor list.
- A topological comparison based detection method for traditional wormhole attack has been presented in Chapter 3. In this method, the originator of the wormhole detection process creates a suspected *one hop* list and then runs the topological comparison algorithm. The applicability of this method in AODV and OLSR routing protocols has also been discussed. Besides, the performance of this detection method (in terms of *detection rate* and *accuracy of alarm*) is compared with some of the existing methods. The results suggest that the topological comparison based approach performs better than the existing methods.
- In Chapter 4, one more topological comparison scheme has been presented to detect Byzantine wormhole tunnels. Unlike the traditional wormhole detection method, *three hop* topologies are compared between the originator and its *one hop* neighbors. The AODV implementation of this method has also been presented. Then, the performance in terms of *detection rate* and *accuracy of alarm* has been measured. The results suggest that both high *detection rate* and *accuracy of alarm* can be achieved.

5.2 Future Work

More performance evaluation of the topological comparison based wormhole detection methods can be done by considering larger network size and complex topologies in dense networks. In addition, node mobility can also be included. Furthermore, a complete secured routing protocol can be developed which not only does the routing but also secures the routes from wormhole attacks (traditional or Byzantine).

Appendix 1

TCL Script

```
#--------------------------Define options for wireless simulation-------------------
------
set opt(chan)                       Channel/WirelessChannel
set opt(prop)                       Propagation/TwoRayGround
set opt(netif)                      Phy/WirelessPhy
set opt(mac)                        Mac/802_11
set opt(ifq)                        Queue/DropTail/PriQueue
set opt(ll)                         LL
set opt(ant)                        Antenna/OmniAntenna
set opt(x)                          1600
set opt(y)                          1600
set opt(ifqlen)                     50
set opt(seed)                       0.0
set opt(adhocRouting)               AODV
set opt(nn)                         30
set opt(t_l)                        3
set opt(stop)                       100.0
set opt(dist_min)                   90
set opt(dist_max)                   200
set opt(dist_t_min)                 200
#---------------------------------------------------------x------------------------
#---------------------------------------------Define necessary objects----------------
------set ns [new Simulator]
set topo [new Topography]
set tracefd [open 500_1000.tr w]
set namtrace [open 500_1000.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $opt(x) $opt(y)
$topo load_flatgrid $opt(x) $opt(y)
set god [create-god [expr $opt(nn)+$opt(t_l)]]
set chan_ [new $opt(chan)]
#---------------------------------------------------------x------------------------
------
proc finish { } {
         global ns namtrace tracefd
         $ns flush-trace
         close $namtrace
         exit 0
}
$ns node-config    -adhocRouting      $opt(adhocRouting) \
                   -llType            $opt(ll) \
                   -macType  $opt(mac) \
                   -ifqType           $opt(ifq) \
                   -ifqLen            $opt(ifqlen) \
                   -antType           $opt(ant) \
                   -propType $opt(prop) \
                   -phyType  $opt(netif) \
                   -channelType      $opt(chan) \
                   -topoInstance     $topo \
                   -agentTrace       ON \
                   -routerTrace      ON \
                   -macTrace         OFF
for { set i 0} { $i < [expr $opt(nn)+$opt(t_l)]} { incr i } {
         set node_($i) [$ns node]
         $god new_node $node_($i)
         $node_($i) random-motion 1
}
set list1 {}
set list2 {}
set r1 [new RNG]
$r1 seed 0
set r2 [new RNG]
$r2 seed 0
for { set j 0 } { $j  < $opt(nn) } { incr j } {
         set len [llength $list1]
         set len1 [llength $list2]
         if { $j < [expr round($opt(nn)/2)] } {
                   set lim_x [expr $opt(x)/2-130]
                            set lim_y [expr $opt(y)]
                            set strt_x 0
                   set strt_y 0
         } else {
                   set lim_x $opt(x)
                   set lim_y $opt(y)
                   set strt_x  [expr ($opt(x))/2+130]
                   set strt_y  0
         }
         if { $len == 0 && $len1 == 0 } {
                   set x_c 5
                   set y_c 5
                   lappend list1 $x_c
                   lappend list2 $y_c
                   set len [llength $list1]
                   set len1 [llength $list2]
         } elseif { $j == [expr round($opt(nn)/2)] } {
                   set x_c [$r1 uniform $strt_x $lim_x]
                   set y_c [$r2 uniform $strt_y $lim_y]
                   lappend list1 $x_c
                   lappend list2 $y_c
                   set len [llength $list1]
                   set len1 [llength $list2]
         } else {
                   set i 0
                   while { $i == 0 } {
                            set ind 0
                            set count 0
                            set x_c [$r1 uniform $strt_x $lim_x]
                            set y_c [$r2 uniform $strt_y $lim_y]
                            foreach m $list1 {
                                     set dist [ expr sqrt(pow(($x_c - [ lindex $list1
$ind ]), 2) + pow(($y_c - [ lindex $list2 $ind ]), 2)) ]
                                     if { $dist < $opt(dist_min)  } {
                                              incr count
                                     }
                                     if { [expr $ind + 1] < [llength $list1] } {
                                               incr ind
                                     }
                            }
                            if { $count == 0 } {
                                     foreach m $list1 {
                                              set dist [ expr sqrt(pow(($x_c - [
lindex $list1 $ind ]), 2) + pow(($y_c - [ lindex $list2 $ind ]), 2)) ]
                                              if { $dist < $opt(dist_max) } {
                                                       lappend list1 $x_c
                                                       lappend list2 $y_c
                                                       incr i
                                                       break
                                              }
                                     }
                            }
                   }
         }
}
puts "size of the list [llength $list1]"
for { set j 0 } { $j < $opt(nn)} { incr j } {
         $node_($j) set X_ [lindex $list1 $j]
         $node_($j) set Y_ [lindex $list2 $j]
         $node_($j) set Z_ 0
         set a($j) [new Agent/myAgent]
         $ns attach-agent $node_($j) $a($j)
         $ns initial_node_pos $node_($j) 20
}
#-----------TRYING a different thing---------#
set wo 1
puts "Starting to setup the wormhole"
set max_count 0
set source -1
set target_node -1
set distance 0
set index [expr round($opt(nn)/2)-1]
set lower_index [expr $index-10]
while { $index >= $lower_index} {
         set counter_dis 0
         for {set n [expr round($opt(nn)/2)]} {$n < $opt(nn) } {incr n} {
                  set dist [ expr sqrt(pow(([ lindex $list1 $n ] - [ lindex $list1
$index ]), 2) + pow(([ lindex $list2 $n ] - [ lindex $list2 $index ]), 2)) ]
                  if {$dist > [expr ($opt(t_l))*$opt(dist_t_min)] && $dist < [expr
($opt(t_l)+1)*$opt(dist_t_min)] } {
                                     set counter_dis [expr $counter_dis+1]
```

```
                    }
                    if { $counter_dis > $max_count } {
                              set max_count $counter_dis
                              set source $index
                              set target_node $n
                              set distance $dist
                              set target_slp [expr ([lindex $list2 $index]-[lindex
$list2 $n])/([lindex $list1 $index]-[lindex $list1 $n])]
                    }
          }
          set index [expr $index-1]
}
puts "Finished counting max"
if { $max_count > 0 } {
          puts "max_count = $max_count"
          puts "source = $source"
          puts "target_node = $target_node"
          puts "target_slope = $target_slp"
          set target $target_node
          set ind $source
          set brd $source
          puts "Target ($target) is [ expr sqrt(pow(([ lindex $list1 $target ] - [
lindex $list1 $ind ]), 2) + pow(([ lindex $list2 $target ] - [ lindex $list2 $ind ]),
2)) ]m or $distance away from the source($source)"
          puts "Slope of the line joining the source and target = $target_slp"
          set control [expr $opt(t_l)+1]
          set additive [expr ([lindex $list1 $target] - [lindex $list1
$ind])/$control]
          puts "additive = $additive"
          set lim_x [lindex $list1 $target]
          set lim_y [lindex $list2 $target]
          set c [expr ([lindex $list2 $ind] - $target_slp * [lindex $list1 $ind])]
          for { set j $opt(nn) } { $j < [expr $opt(nn)+$opt(t_l)] } { incr j }  {
                    set additive [expr [expr ([lindex $list1 $target] - [lindex $list1
$ind])]/$control]
                    set strt_x [lindex $list1 $ind]
                    set strt_y [lindex $list2 $ind]
                    set i 0
                    while { $i == 0} {
                              set x_c [expr [lindex $list1 $ind]+$additive]
                              set y_c [expr $x_c*$target_slp + $c]
                              set dist [ expr sqrt(pow(($x_c - [ lindex $list1 $ind ]),
2) + pow(($y_c - [ lindex $list2 $ind ]), 2)) ]
                    }
                    set control [expr $control-1]
                    $node_($j) set X_ $x_c
                    $node_($j) set Y_ $y_c
                    $node_($j) set Z_ 0
                    lappend list1 $x_c
                    lappend list2 $y_c
                    if {$wo == 1 || $wo == $opt(t_l)} {
                              set a($j) [new Agent/myAgent]
```

```
                        $ns attach-agent $node_($j) $a($j)
                } else {
                        set b($j) [new Agent/Tunnel]
                        $ns attach-agent $node_($j) $b($j)
                }
                $ns initial_node_pos $node_($j) 20
                incr wo
                $node_($j) color "Red"
                set ind $j
                puts "Colluder [expr $wo-1] placed at [expr round($dist)]m away"
        }
        puts "size of the list [llength $list1]"
        for { set j 0} { $j < [expr $opt(nn)+$opt(t_l)] } { incr j } {
                if {$j <= $opt(nn) || $j == [expr $opt(nn)+$opt(t_l)-1]  } {
                $ns at 0.5+j/10 "$a($j) reset_nodes"
                }
        }
        set tcp1 [new Agent/TCP]
        set sink1 [new Agent/TCPSink]
        $ns attach-agent $node_($brd) $tcp1
        $ns attach-agent $node_($target) $sink1
        $ns connect $tcp1 $sink1
        set ftp1 [new Application/FTP]
        $ftp1 attach-agent $tcp1
        $ns at 0.0 "$ftp1 start"
        $ns at 10.0 "$a($brd) one_hop_neighbor"
        $ns at 15.0 "$a($brd) 3_hop_neighbors"
        $ns at 25.0 "$a($brd) detect_one_hop"
        $ns at 40.0 "$a($brd) detect_3_hop"
        $ns at 60.0 "$a($brd) find_wormhole"
        $ns at $opt(stop).001 "$a($brd) print_result"

}
for { set j 0} { $j < [expr $opt(nn)+$opt(t_l)] } { incr j } {
        $ns at $opt(stop).005 "$node_($j) reset"
}
$ns at $opt(stop).010 "finish"
$ns run
```

Appendix 2

The probability of not having a common one hop neighbor between two neighbors in the network

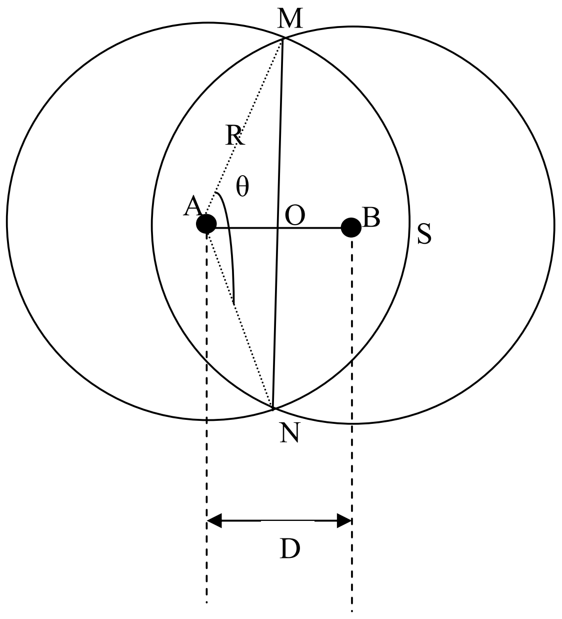


Figure A.1 Overlapping area of two nodes

In Fig. A.1, the overlapped transmission area of two nodes, A and B, is shown. Both A and B have common transmission radius (R) and the distance between them is D. If we calculate the area of the overlapped transmission zone, we can calculate the probability that no other nodes in that area as shown in [49].

Area of the sector (AMSN) $= \frac{1}{2} * radius * arc\ length$

$= \frac{1}{2} * R * S$

$= \frac{1}{2} * R^2 * \theta$

$= R^2 \cos^{-1}(\frac{D}{2R})$

Area of the triangle (AMN) $= \frac{1}{2} * base * height$

$= \frac{1}{2} * MN * AO$

$= ON * \frac{D}{2}$

$$= \sqrt{R^2 - \frac{D^2}{4}} * \frac{D}{2}$$

Area of the overlap, A(D) $= 2 * (Area\ of\ the\ sector - Area\ of\ the\ triangle)$

$$= 2 * (R^2 \cos^{-1}\left(\frac{D}{2R}\right) - \sqrt{R^2 - \frac{D^2}{4}} * \frac{D}{2})$$

Now, as shown in [49], the probability that there are 0 nodes in the overlapped area A(D) can be written as:

$$\text{P0 (D)} = e^{-\rho A(d)} * \frac{\rho A(d)^0}{0!}$$
$$= e^{-\rho A(d)}$$

Since the distance (D) between two nodes can be maximum R, the distance density function can be represented as:

$$\text{P(D)} = \frac{1}{R^2} * \frac{\partial D^2}{\partial D}$$

$$= \frac{2D}{R^2}$$

Therefore, the probability that there will be no other nodes in the overlapped transmission zone A(D) can be written as:

$$\text{P} = \int_0^R P(D) * P0(D)\, dD$$

$$= \int_0^R \frac{2D}{R^2} * e^{-\rho A(D)}\, dD$$

$$= \int_0^R \frac{2D}{R^2} * e^{-\rho * 2*(R^2 \cos^{-1}\left(\frac{D}{2R}\right) - \sqrt{R^2 - \frac{D^2}{4}} * \frac{D}{2})}\, dD$$

$$= [\frac{D^2 * e^{\left(\rho * D * \sqrt{R^2 - \frac{D^2}{4}} - 2 * R^2 * \rho * \cos^{-1}\frac{D}{2R}\right)}}{R^2}] \text{ where } 0 \leq D \leq 250$$

$$= e^{-1.18 * \rho * R^2}$$

We assume that the transmission radius of each node in the netwoek is 250 meter. The probabability graph for different node density is shown in Fig. A.2 in the next page. In the figure, we can see that in most cases two nodes will have a common one hop neighbor.

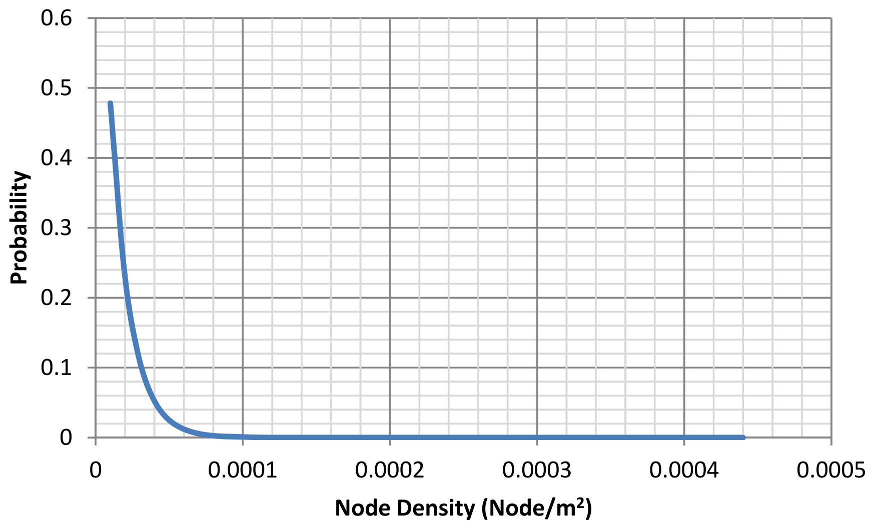


Figure A.2 Probability of not having a common neighbor between two nodes when R=250

Bibliography

1. C. Perkins, *Ad hoc networking*, Addison-Wesley, 2000.
2. J. Sun, *Mobile ad hoc networking: an essential technology for pervasive computing.* Proceedings of International Conferences on Infotech & Infonet, Beijing, China, C: p. 316–321.
3. M. Bansal, R. Rajput, and G. Gupta, *Mobile ad hoc networking (MANET): routing protocol performance issues and evaluation considerations.*Mobile Ad-hoc Network (MANET) Working Group, IETF (1998).
4. H. Yang, H. Luo, F. Ye, S. Lu, *et. al.*, *Security in mobile ad hoc networks: challenges and solutions.* IEEE Wireless Communications, 2004. 11(1): p. 38-47.
5. M. Lasermann, *Characterizing MANET topologies and analyzing their impact on routing protocols.* Diploma Thesis, Stuttgart University, Germany, 2002.
6. C. Perkins and P. Bhagwat. *Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers*. In Proceedings of SIGCOMM '94 Conference on Communications, Architectures, Protocols, and Applications, (London, UK, Sept. 1994), p. 234-244.
7. C. Adjih, A. Laouiti, P. Minet, *et. al.*, *Optimized link state routing protocol.* Work in Progress, IETF draft, MANET Working Group, INRIA Rocquencourt, France, 2003.
8. C. Perkins and E. Royer. *Ad-hoc on-demand distance vector routing*. In Workshop on Mobile Computing and Systems Applications, 1999.
9. D. Johnson and D. Maltz, *Dynamic source routing in ad hoc wireless networks.* In Mobile computing, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996: Ch. 5, p. 153-181.

10. V. Park and M. Corson, *Temporally-ordered routing algorithm (TORA) version: 1 functional specification*. Internet-Draft, draft-ietf-manet-tora-spec-00. txt, November 1997.

11. Z. Haas and M. Pearlman, *The zone routing protocol (ZRP) for ad hoc networks.* Internet draft, Mobile Ad-Hoc Network (MANET) Working Group, IETF, 1998.

12. P. Khengar and A. Aghvami, *Warp-the wireless adaptive routing protocol.* In Proceedings of IST Mobile Communications Summit 2001, 2001, p. 480–485.

13. M. Abolhasan, T. Wysocki, and E. Dutkiewicz, *A review of routing protocols for mobile ad hoc networks.* Ad Hoc Networks 2 (1), 2004, p. 1-22.

14. L. Ford and D. Fulkerson, *Flows in networks.*Princeton, New Jersey: Princeton University Press, 1962, p. 11.

15. D. Djenouri, L. Khelladi, and A. Badache, *A survey of security issues in mobile ad hoc and sensor networks.* IEEE Communications surveys & tutorials, 2005. 7(4): p. 2-28.

16. B. Wu, J. Chan, J. Wu, *et. al.*, *A survey of attacks and countermeasures in mobile ad hoc networks.* Wireless Network Security, 2007: p. 103-135.

17. C. Gray, J. Byrnes, and S. Nelakuditi, *Pair-wise resistance to traffic analysis in MANETs.* SIGMOBILE Mobile Computing and Communications Review, 2008. 12(1): p. 20-22.

18. A. Cardenas, N. Benammar, G. Papageorgiou, *et. al.*, *Cross-layered security analysis of wireless ad hoc networks.* In Proceedings of 24th Army Science Conference, 2004.

19. P. Yi, Z. Dai, S. Zhang, *et. al.*, *A new routing attack in mobile ad hoc networks.* International Journal of Information Technology, 2005. 11(2): p. 83-94.

20. M. Zapata, *Secure ad hoc on-demand distance vector routing.* ACM SIGMOBILE Mobile Computing and Communications Review, 2002. 6(3): p. 106-107.

21. M. Zapata and N. Asokan. *Securing ad hoc routing protocols*. In Proceedings of 1st ACM Workshop on Wireless Security, 2002.

22. Y. Hu, A. Perrig, and D. Johnson, *Ariadne: A secure on-demand routing protocol for ad hoc networks.* Wireless Networks, 2005. 11(1): p. 21-38.

23. K. Sanzgiri, B. Dahill, B. Levine, *et. al., A secure routing protocol for ad hoc networks*. In Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP '02), November 2002, p. 78-87.

24. B. Kannhavong, H. Nakayama, Y. Nemoto, *et. al.*, *A survey of routing attacks in mobile ad hoc networks.* IEEE Wireless Communications, 2007: p. 85-91.

25. Y. Hu and A. Perrig, *A survey of secure wireless ad hoc routing.* IEEE Security and Privacy magazine, 2004. 2: p. 28-39.

26. J. Baras, S. Radosavac, G. Theodorakopoulos, *et. al., Intrusion detection system resiliency to byzantine attacks: The case study of wormholes in OLSR*. In Proceedings of Military Communication Conference (MILCOM '07), 2007, p. 1-7.

27. Y. Hu, A. Perrig, and D. Johnson, *Wormhole attacks in wireless networks.* IEEE Journal on Selected Areas in Communications, 2006. 24(2): p. 370-380.

28. R. Maheshwari, J. Gao, and S. Das. *Detecting wormhole attacks in wireless networks using connectivity information*. In Proceedings of INFOCOM '07, May 2007, p. 107-115.

29. R. Poovendran and L. Lazos, *A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks.* Wireless Networks, 2007. 13(1): p. 27-59.

30. L. Lamport, R. Shostak, and M. Pease, *The Byzantine generals problem.* ACM Transactions on Programming Languages and Systems (TOPLAS), 1982. 4(3): p. 401.

31. B. Awerbuch, R. Curtmola, D. Holmer, *et. al.*, *Mitigating byzantine attacks in ad hoc wireless networks.* Department of Computer

Science, Johns Hopkins University, Technical Report Version 1, March 2004.

32. A. Sangi, J. Liu, and L. Zou. *A performance analysis of AODV routing protocol under combined byzantine attacks in MANETs*. In Preceedings of International Conference on Computaional Intelligence and Software Engineering (CiSE '09), December 2009. p. 1-5.

33. Z. Tun, and A. Maw, *Wormhole attack detection in wireless sensor networks.* World Academy of Science, Engineering and Technology, 2008.

34. F. Nait-Abdesselam, B. Bensaou, and T. Taleb, *Detecting and avoiding wormhole attacks in wireless ad hoc networks.* IEEE Communications Magazine, 2008. 46(4): p. 127-133.

35. T. Phuong, N. Canh, Y. Lee, *et. al. Transmission Time-Based mechanism to detect wormhole attacks*. In Proceedings of The 2nd IEEE Asia-Pacific Service Computing Conference, 2007.

36. M. Khabbazian, H. Mercier, and V. Bhargava, *Severity analysis and countermeasure for the wormhole attack in wireless ad hoc networks.* IEEE Transactions on Wireless Communications, 2009. 8(2): p. 736-745.

37. S. Choi, D. Kim, D. Lee, *et. al*. *WAP: Wormhole attack prevention algorithm in mobile ad hoc networks*. In Proceedings of IEEE International Conference on Ubiquitous and Trustworthy Computing, June 2008. p. 343-348.

38. G. Lee, J. Seo, and D. Kim. *An Approach to mitigate wormhole attack in wireless ad hoc networks*. In Proceedings of IEEE International Conference on Information Security and Assurance, 2008.

39. L. Hu and D. Evans. *Using directional antennas to prevent wormhole attacks*. In Proceedings of the 11th Network and Distributed System Security Symposium, 2004, p. 131-141.

40. X. Su and R. Boppana. *Mitigating wormhole attacks using passive monitoring in mobile ad hoc networks*. In Proceedings of IEEE GLOBECM '08, December 2008, p. 1-5.

41. B. Awerbuch, R. Curtmola, D. Holmer, *et. al.*, *ODSBR: An on-demand secure Byzantine resilient routing protocol for wireless ad hoc networks.* ACM Trans. Inf. Syst. Secur., 2008. 10(4): p. 1-35.

42. T. Ho, M. Medard, R. Koetter, *et. al.*, *A random linear network coding approach to multicast.* IEEE Transactions on Information Theory, 2006. 52(10): p. 4413-4430.

43. E. Royer and C. Perkins. *Multicast operation of the ad-hoc on-demand distance vector routing protocol*. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1999.

44. M. Yu, M.C. Zhou, and W. Su, *A secure routing protocol against byzantine attacks for MANETs in adversarial environments.* IEEE Transactions on Vehicular Technology, 2009. 58(1): p. 449-460.

45. D. Sterne, G. Lawler, R. Gopaul, *et. al., Countering false accusations and collusion in the detection of in-band wormholes.*In Proceedings of 23rd Annual Computer Security Applications Conference, 2007, p. 243-256.

46. J. Eriksson, M. Faloutsos, and S. Krishnamurthy, *Toward collusion-robust link-state routing in open networks.* University of California, Riverside Technical Report, 2005

47. P. Karn and C. Partridge, *Improving round-trip time estimates in reliable transport protocols.* ACM SIGCOMM Computer Communication Review, 1995. 25(1): p. 66-74.

48. P. Kruus, D. Sterne, R. Gopaul, *et. al., In-band wormholes and countermeasures in OLSR networks*. Securecomm and Workshops, 2006. p. 1-11.

49. C. Bettstetter, *On The Minimum Node Degree and Connectivity of a Wireless Multihop Network.* MOBIHOC, 2002. p. 80 -91.