Handling Attacks on Routing Protocols in Ad hoc Networks

A thesis submitted to University of Delhi

in partial fulfillment of the requirements

for award of the degree of

Doctor of Philosophy in Computer Science

by

Sandhya Khurana

Department of Computer Science

University of Delhi

Delhi-110007 India

June, 2011

Handling Attacks on Routing Protocols in Ad hoc Networks

A thesis submitted to University of Delhi

in partial fulfillment of the requirements

for award of the degree of

Doctor of Philosophy in Computer Science

by

Sandhya Khurana

Department of Computer Science

University of Delhi

Delhi-110007 India

June, 2011

Dedicated to my son

Declaration

The thesis entitled "*Handling Attacks on Routing Protocols in Ad hoc Networks*", which is being submitted for the award of the degree of Doctor of Philosophy, in Computer Science, is a record of original and bona fide research work carried out by me in Department of Computer Science, University of Delhi, Delhi, India.

The work presented in this thesis has not been submitted to any other university or institute for the award of any degree or diploma.

Sandhya Khurana
Department of Computer Science,
University of Delhi,
Delhi, India.

Certificate

This is to certify that the thesis entitled "*Handling Attacks on Routing Protocols in Ad hoc Networks* " being submitted by Sandhya Khurana in the Department of Computer Science, University of Delhi, Delhi, for the award of degree of Doctor of Philosophy is a record of original research work carried out by her under the supervision of Dr. Neelima Gupta.

The thesis or any part thereof has not been submitted to any other University or Institute for the award of any degree or diploma.

Supervisor	**Head**
Dr. Neelima Gupta	Dr. Neelima Gupta
Department of Computer Science	Department of Computer Science
University of Delhi	University of Delhi
Delhi, India.	Delhi, India.

Acknowledgment

First of all I thank to God, the Almighty, who gave me the strength and the patience to accomplish this work.

There are not words enough to thank my advisor, I express my sincere gratitude to my guide Dr. Neelima Gupta for the continuous support of my Ph.D study and research, for her motivation, enthusiasm, and immense knowledge. Her continued guidance helped me in all the time of research and writing of this thesis. She was kind enough to give time for discussion even after office hours.

Besides my guide, I am also deeply thankful to my head of department, Dr. M.K. Das, for his cooperation during the research.

I am also thankful for my father who has been main inspiration for me to to pursue my Ph.D. Ever since I was a child he was always telling me that a Ph.D. will put you at the pinnacle of your profession; it was always his dream to see me a Doctor one day. I am also thankful for the support and love that I got from my family members: my son and my husband.

Last but not the least, my sincere thanks also goes to support staff at Department of Computer Science who helped and supported me throughout studies.

Sandhya Khurana

Abstract

Mobile ad hoc networks ($MANET$s) have been proposed to support dynamic scenarios where no infrastructure exists. Each node in the network acts as a host as well as a router and, forwards traffic to other nodes. $MANET$s can be set up quickly and at low cost in contrast to infrastructure networks which may be wired or wireless. Military networks like air force networks between airplanes and navy networks between ships, network between various sites in emergency disaster relief and interaction between attendees at a meeting are some of the common examples where ad hoc networks are preferred.

Efficient and secure routing is the heart of any network and is especially challenging in an infrastructure-less network where participating nodes are portable and mobile. Routing in $MANET$s is vulnerable to threats since communication in ad hoc networks is dependent upon the cooperation of nodes for forwarding packets. Therefore, it is a challenge for researchers to embed solutions to assuage attacks in existing routing protocols. Passive eavesdropping, active impersonation, message replay and message distortion are some of the common attacks in ad hoc networks.

Our work contributes towards mitigating attacks on routing protocols in ad hoc networks. We provide solutions to handle three types of attacks namely blackhole attack, wormhole attack and attack due to selfish nodes. Network performance is known to degrade significantly in presence of blackhole and selfish nodes. Wormhole tunnels have a severe impact on the neighborhood discovery process which forms the backbone of routing in $MANET$s. It is also known to partition the network by making one side of the network unreachable from the other side. Since no protocol is known to handle all types of attacks, we also propose a solution to compute a path that is exposed minimum to the nodes under the danger of attack.

We first propose a scheme to mitigate both blackhole attack and attack due to selfish nodes in a single solution. The proposed algorithm also assuages multiple blackhole nodes in the network. No algorithm is known to handle both blackhole nodes and selfish nodes simultaneously. Our solution improves upon the existing solutions to handle blackhole/ selfish node in terms of overheads and hardware requirements.

Next, we present an end-to-end solution to alleviate the effect of wormhole attack. This solution also improves upon the existing solutions in terms of overheads and infrastructure requirements.

Lastly, we present an algorithm to find a path that is farthest from the nodes under the danger of attack. The problem has hot been addressed in the context of ad hoc networks earlier. In a highly mobile environment of $MANETs$ it is legitimate to look for a path where an intruder can not get in easily. The path is computed in optimal $O(|P|)$ time where $|P|$ is the length of the path. Assuming that the packets are received from the shortest path first, the algorithm computes a shortest such route.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Mobile Ad hoc Networks ($MANETs$) are wireless networks without any infrastructural support. With the expansion in the production of technology, mobile computing has seen exponential growth in the past few decades [RR02]. As compared to the products available a decades ago, greatly improved products supporting wireless data communication are now available in the market. The bandwidth available to laptop computers over radio and infrared links is easily 10 to 100 times more than that available just ten years ago. Projections have been made that by the year 2020 there will be more than a 50 billion [Kri11] devices supporting wireless communication and more than a billion wireless telephone handsets will be purchased annually. With new wireless applications added to these devices every few months, ad hoc networks will become useful in many ways. For instance university campuses are becoming large ad hoc networks as students and faculty learn to rely on their handheld devices and laptop computers for their communication and computing needs. Messaging and browsing can be managed either through an available wireless infrastructure or by ad hoc connectivity according to whatever is most convenient at the moment. Similarly, at hospitals busy doctors and nurses may either rely on the administrative infrastructure or may want to instantiate direct links outside the infrastructure. Visiting staff and paramedics may also need to confer with residents and transfer data to and from a patient's equipment. These operations were earlier facilitated by infrastructural support but are now often met without interaction with the infrastructure.

As the nodes in $MANET$s have limited transmission range, most of the operations are performed by cooperation amongst the nodes. This along with the wireless nature of the links make ad hoc networks vulnerable to various kinds of security attacks. Many of the applications, particularly military networks, require a high level of security. Meeting the service requirements in terms of security poses a major challenge for researchers in ad hoc networks.

Routing is one of the basic functions performed by the nodes in any network. Since packets in ad hoc networks pass through a series of nodes, lack of central authority and wireless links expose the routed messages to several types of malicious attacks. Attacks range from passive eavesdropping in which an attacker may get access to secret information thereby violating the confidentiality, to active impersonation, message replay, and message distortion. Attacks may be by an external source which is not a part of the network and hence does not have valid signatures or could be from a compromised node within the network. Chances of a node being compromised in a hostile environment (e.g., a battlefield) with relatively poor physical protection are non-negligible. Since the external attackers do not have valid digital signatures, it is easier to detect such attacks as compared to the one by an internal compromised nodes using cryptographic schemes. However as pure cryptographic schemes are compute-intensive, attempts are on to protect the networks using a little or no cryptography.

Several types of attacks [Per88, SA99, Dou02, HPJ03b, NSSP04, AJ04, PSL06, JWY06] on ad hoc networks have been discussed in literature. Some of these cripple the network by disrupting the route of the legitimate packets while others inject too many extra packets in the system thereby consuming system resources like bandwidth, memory/computational power of nodes. Malicious nodes also attack by inserting erroneous routing updates, replaying old routing information, changing routing updates, or advertising incorrect routing information so that the network is not able to provide service properly. Two main approaches are used to mitigate attacks in ad hoc networks. The first approach aims at detecting the malicious nodes while computing the route in the network and re-routing the packets around it, mostly along the shortest path among them. Most of these protocols [KGA06, BH01, MM02a, BB02b, HJP02, HPJ02, PH02, SDL$^+$02, YNK02] are based on existing ad hoc routing protocols like Dynamic Source Routing (DSR) [JM96], Ad-hoc On demand Distance Vector ($AODV$) [PRD03] and Destination Sequence Distance Vector ($DSDV$) [PB94] redesigned to handle attacks. The second approach [ZL00, ZLH03, HL03] separates the detection of malicious nodes from routing.

A **blackhole** attacker tries to include itself into the path during route discovery without actually having a path to the destination. Once included in the path, it may drop packets thereby downgrading the communication in the network. It can also drop received routing messages instead of relaying them in order to reduce the quantity of routing information available to other nodes and having the effect of making the destination unreachable. Parsons et al. [PE09] have shown that the packet loss ratio (PLR) of DSR increases significantly in presence of blackhole nodes ($BHNs$). We also observe that the packet delivery ratio (PDR) of $AODV$ decreases drastically when a blackhole is present in the network. The attacker can also store the data and perform traffic analysis.

Attacks like reducing the amount of routing information available to other nodes, failing to advertise certain routes may be also due to **selfish** behavior of a node. Sometimes a node does not participate in normal functioning of the network in order to save battery life for its own purposes. The intention of a selfish node (SN) is not to cripple the network; however, its impact can not be undermined. Michiardi et al. [MM02b] and Kargal et al. [KKSW04] have shown that the performance of DSR algorithm degrades drastically as the number of selfish nodes increase in the network. We show that the same is true for the $AODV$ algorithm also.

Another very challenging attack in ad hoc networks is the **wormhole attack**. As the mobile devices use a wireless medium to transmit information, a malicious node can eavesdrop the packets, tunnel them to another location in the network and retransmit them at the other end. The tunnel so created forms a **wormhole**. The adversary can copy the neighbor discovery hello messages and replay them from a distant location almost instantly. Nodes in the neighborhood of the distant location receive the hello messages and reply to them. The replies are copied and replayed at the original location. Thus the two nodes which are actually located distantly get the illusion that they are neighbors located within the range of each other. As routing in $MANETs$ rely heavily on the neighborhood information, it severely disrupts the normal routing function. A wormhole can also relay route request and response messages between distant nodes creating appearance of shorter routes to destinations. If the wormhole peacefully transports all the traffic from one location in the network to another location that is far away, it is beneficial for the network operations as it improves the network connectivity. Unfortunately, once the traffic is routed through the wormhole, the attacker can gain full control over the traffic and start its malicious actions by selectively dropping data packets which will lower the network throughput or store all the traffic and perform cryptanalysis attacks later. The attacker can decide to drop data packets that pass through the wormhole at some critical situations. The attacker can periodically turn the wormhole link off and on. Turning the wormhole link off suddenly means that all the nodes that used the wormhole to reach other nodes will have to add new routes. Thus, the network will be clogged with many route requests disrupting the operation of the network leading to denial-of-service (DoS) attack. Furthermore, the attacker can turn a good node to a sinkhole [KW03] and may cause that node to be mistakenly blacklisted by other nodes in the network. The severity of the wormhole attack comes from the fact that it is difficult to detect such replays using cryptography as even encrypted or digitally signed signals can be copied and replayed; hence it is effective even in a network where confidentiality, integrity, authentication, and non-repudiation (via encryption, digesting and digital signature) are preserved. The wormhole attack can be launched regardless of the message authentication code (MAC) and routing or cryptographic protocols used in the network.

Since most of the existing protocols do not handle all types of attacks it is imperative to find a solution that would reduce the impact of attacks on routing. One way of doing this is to determine a path which is farthest from the nodes which are under the danger of attack. In our work we have called such nodes as **Endangered Nodes** and such paths as Minimum Exposed Paths to Attacks ($MEPA$). To the best of our knowledge this problem has not been addressed earlier in the context of ad hoc networks. One application of the above problem is in a scenario where a node may wish to avoid to route its packets through the nodes of its enemies. Since the enemy may be highly mobile, we would like to find a route as much away from it as possible. While we can use cryptography in an attempt to protect our information we have little control over which parties can observe our encrypted packets. Even without decryption, merely observing traffic patterns and volume can provide useful intelligence. One may also wish to avoid nodes of a network for legal reasons.

1.2 Our Work

In this work, we have focused on attacks due to blackhole nodes, selfish nodes and wormhole nodes. Two of our proposed solutions detect these attacks and compute a shortest secure path. The third algorithm assumes that the nodes which are under the danger of attacks are known (say by using intrusion detection mechanisms) and computes a secure path which is exposed minimum to the such endangered nodes.

1.2.1 Attacks due to Blackhole and Selfish nodes

1. BLACK HOLE ATTACK : In routing protocols like $AODV$ where an intermediate node may reply to a route request with a route to destination, a malicious node may respond positively to a request for a shortest route even though it does not have a valid route to the destination. Since the node does not have to check its routing table it is the first one to respond to route discovery request in most cases. When the data packet sent by the source reaches the malicious node it drops the packets rather than forwarding them to the destination making a **black hole** there.

2. SELFISH NODES : Due to limited power supply in ad hoc networks sometimes nodes reserve power for their own use and do not cooperate in forwarding packets. The intention of a node is not to cripple the normal functioning of the network but to save its resources for its own purpose. Such a node is called a **selfish node**. Two types of selfish nodes are defined depending upon the extent of their non-cooperation in network operations.

   (a) Selfish node of Type 1 uses energy only for its communication and neither forwards controls packets nor data packets.

(b) Selfish node of Type 2 forwards control packets but does not forward data packets. Here we make an assumption that once a node stops forwarding data packets, it does not involve itself into route establishment also unless it regains its energy levels. Let E be the initial maximum energy of a node. When energy of the node falls within $(T1, E]$ the node behaves properly and executes both routing functions and packet forwarding. When energy falls in $(T2, T1]$, the node forwards control packets but disables data packet forwarding. Since now the node no longer wants to participate in data packet forwarding and its intention is not to disrupt the normal functioning of the network it is legitimate to assume that it will no longer participate in route establishment until its energy is restored. Within a limited time interval the node is recharged and its energy level is set back to the initial value.

Related Work

Approaches to handle blackhole nodes either use cross-verification [DLA02, YM06] or watchdog mechanism [MM00, PM03, Ban08, AGD08] along with Intrusion Detection System (IDS) [HL04, HFLY03, RFdAG08] and learning theory [KNK$^+$07]. Most of the approaches using cross-verification flood the verification packets and hence incur a lot of communication overhead. Watchdog mechanisms require the nodes to listen to their neighbor nodes in promiscuous mode. Switching the modes between promiscuous mode and transmit/receive mode is not easy and is error prone [KKSW04]. Approaches using IDS or learning theory are compute-intensive and incur large storage and communication overhead as they collect and analyze large amount of data for anomaly detection. In some approaches [SYP04, TS07] the source node waits for some time, collects some paths and selects the path that shares one or more nodes with at least one more path. It is based on the hypothesis that if two paths share a node, it is unlikely that it is under attack. The approach suffers with the delay in establishing the route besides the fact that the probability of a blackhole node on the path is non-zero.

Most of the work to handle selfish nodes either propose a reputation based trust system [MM00, BB02a, MM02a, PW02, BB02b, LY02, YZV03, HWK04, GS04, KKSW04, RSDE05, WSST05, STW06] or are based on providing economic incentives. Reputation based systems suffer with the drawback of spreading rumors or false accusations. Moreover most of these approaches require neighborhood monitoring in promiscuous mode. Incentive based schemes [BH00, BH01, BH03, JHB03, ZCY03] treat packet forwarding as a service that can be priced, and introduce some form of virtual currency to encourage packet forwarding. However, these schemes require tamper-proof hardware or virtual banks. Where tamper-resistant devices in general might be next to impossible to realize, approaches requiring virtual banks need a fixed communication infrastructure to implement the incentive schemes which is not applicable for a pure ad hoc network.

The existing approaches mentioned above address only a single type of attack. Moreover, most of these approaches do not handle collaborative and multiple attacks. The proposed solutions to handle multiple/collaborative attacks [BdOZI09, RFdAG08, KNK$^+$07, RFN05, HL04, HFLY03, Bha02] are either based on Intrusion Detection System (IDS) or are recursive application of the approach proposed for a single attack. To the best of our knowledge no algorithm handles more than one type of attack in a single scheme. Handling more than one type of attack in a single scheme is a major challenge for researchers. Bhargava et al. [BdOZI09] have suggested a scheme to classify the attacks on the basis of observed behavior and then take corrective measures accordingly.

Our Contribution

We present a solution called 'Reliable Distance Vector routing protocol to handle Blackhole and Selfish ($RDVBS$) nodes' [KGA06] that mitigates both blackhole attack as well as attack due to selfish nodes in a single scheme. The proposed scheme handles multiple/collaborative attacks also. Path discovery in $RDVBS$ can be thought of as consisting of two phases. Phase I is a slight modification of path-discovery phase of $AODV$. Instead of keeping one route reply, we keep all the replies so that an alternate secure path can be discovered in case a blackhole node is present on a shortest path. When the source receives a route reply the reliability of the path is checked by sending verification packets, reply to which can be generated only by the destination node. If there is a black node on the path, the destination will not receive the control packet (as there is no path from the black node to the destination) and hence no reply would be generated along that path. A secure path is established along the path through which a reply to this packet is received (of course, existence of a secure path is assumed here). Here we point out that the verification packets are not flooded but are multicast to a selected group of nodes (through which route replies were received). Once a path free of black node is discovered, control packets are sent periodically to maintain the reliability of the path, i.e. to detect if any selfish node has crept into the path.

We compare our algorithm with Deng et al.'s approach (henceforth referred to as $DENG$) as it also uses cross-verification to mitigate blackhole attack. We show that our algorithm outperforms their solution in terms of routing overhead (RO) without affecting other parameters like Average End-to-End Delay ($AEED$) and Packet Delivery Ratio (PDR). The impact of selfish nodes on $AODV$ was also studied. We show that the PDR of $AODV$ drops by more than 55% when 50% of nodes in the network are selfish and it degrades by $10\% - 20\%$ every time the number of selfish nodes increases by 10%. We show that with our protocol there is only a slight degradation of performance with the increase in the number of selfish nodes in the network. PDR degrades just by $1\% - 3\%$ every time the percentage of selfish nodes increase by 10%. $AEED$ and RO do not increase much with increase in the number of selfish nodes.

In the absence of attack also $RDVBS$ outperforms $DENG$ in terms of routing overhead. In this case, routing overhead of our algorithm is only slightly more than that of $AODV$ (this is natural to expect). Table 1.1 gives the salient contribution of the proposed scheme viz-a-viz other protocols.

1.2.2 Attacks due to Wormhole nodes

As the mobile devices use a wireless medium to transmit information, a malicious node can eavesdrop the packets, tunnel them to another location in the network and retransmit them at the other end. The tunnel so created forms a **wormhole**. The tunneling procedure generates an illusion that the two nodes more than one hop away are in the neighborhood of each other. We call the two nodes as the victim nodes. Since most of the route discovery

Protocol	Handles BHN	Handles SN	Handle Multiple $BHNs$	Handle Collaborative $BHNs$	Remarks
$DENG$ [DLA02]	Yes	No	No	No	Incurs routing overhead
$Tamilselvan$ [TS07]	Yes	No	Yes	No	May skip a single black hole with a non-zero probability; suffers with delay in establishing the path
Incentive based schemes ($Nuglets$) [BH00, BH01, BH03]	No	Yes	No	No	$Nuglets$ are difficult to implement
$CORE$(A COl-laborative REp-utation Mechanism) [MM02a]	No	Yes	No	No	Special Hardware required for Watchdog (WD) mechanism; switches the nodes between transmit/receive and promiscuous mode
$CONFIDANT$ (Co-operation of Nodes Fairness In Dynamic Ad-hoc Networks) [BB02a]	No	Yes	No	No	Special Hardware required for Watchdog (WD) mechanism; switches the nodes between transmit/receive and promiscuous mode
$RDVBS$	Yes	Yes	Yes	Yes	Simple to Implement with less routing overhead

Table 1.1: Table of Comparison between protocols handling blackhole and selfish nodes.

mechanisms maintain a neighborhood set at each node false information about a node's neighbor can severely affect the discovered route. If the routing protocol uses the number of hopcounts to compute the shortest path it prevents routes longer than three hops to be discovered between the victim nodes.

Related Work

Most of the existing approaches to mitigate wormhole attack are node-to-node [BC93, HPJ03a, CBH03, HE04, KBS05, NAT06, PL07, PM08, KBS08]. Such approaches either assumes a trust level between two neighboring nodes or carries out some sort of neighborhood validation on its own. The trust level is achieved using cryptographic schemes [PL07] or by neighborhood monitoring mechanisms. These schemes require the use of special hardware devices like directional or smart antennas [HE04], ultrasound [SSW03], special radio frequency devices [CBH03], global topology and connectivity information [MGD07, HCJ07, LS10] and synchronized clocks [HPJ03a] for neighborhood validation.

Hu and Evans [HE04] presented a solution in which a node validates its neighborhood set with the help of directional information shared between the nodes. Maheswari et al. [MGD07] proposed a graph theoretic approach in which the detection algorithm looks for a specific type of substructure in the connectivity graphs that should not be present in an attack free graph. In [KBS05, KBS08] Khalil et al. proposed the use of guard nodes to guard the traffic going in and out of its neighbors. It requires the node to listen to its neighbors in promiscuous mode. Hu et al. [HPJ03a] introduced the notion of packet leash as a general mechanism for detecting and thus defending against wormhole attacks. A leash is any information that is added to a packet designed to restrict the packet's maximum allowed transmission distance. The receiver detects the wormhole attack if it finds that a packet has traveled more than the allowed distance.

Node-to-node schemes incur a lot of computing and communication overhead as a lot of information needs to be exchanged to carry out verification at every node. Wang et al. [WBLW06] proposed a mechanism requiring only end to end trust. They require that the nodes know their positions and assume loosely synchronized clock. It computes the moving speed of a node by examining its position at various times. If the speed is found to be more than a certain threshold v, they declare a wormhole on the path. It requires $O(km)$ storage and $O(km^2)$ computation time where k is the number of hops on the path. To reduce the storage requirement and the computation time, they divide the network area into a number of cells and the time into equal time slots. For every node they store only one record for one (cell no, time slot) pair and hence compute the average moving speed for lesser number of pairs. They called the extended algorithm as 'Cell based Open Tunnel Avoidance ($COTA$)'. They achieve a reduction of $O(m)$ factor in storage requirement and the computation time. However their is a trade-off between the storage requirement and the number of false positives/detection capability and also between the computation time and the number of false positives/detection capability. Other end-to-end schemes [CL06, QSL07, THL$^+$07, SB07] handle only a specific type of wormhole which is launched by encapsulation.

Our Contribution

We propose an End-to-End scheme to secure ad hoc networks against Wormhole attacks (EEW). The proposed scheme requires that every node in the network is equipped with a GPS and that every node knows its location. The storage and the computation overhead is low as compared to Wang's algorithm. We do not store more than one packet at the destination, hence our protocol requires only $O(k)$ space and time. Comparison of our algorithm with Wang et al.'s end to end algorithms are summarized in Table 1.2.

Our scheme can be included in the route discovery process as well as used from time to time to examine the path for the presence of wormhole. It can be used as a plug-in for any existing routing protocol like DSR or $AODV$. It neither requires directional antennas nor tight synchronized clocks. The protocol requires trust only between the source and the destination which is achieved using symmetric key encryption. The detection is carried out only at the destination and hence there is no need for neighborhood validation. The protocol handles all types of wormholes and not just the one launched by encapsulation.

Our idea to handle wormhole attack in the network is simplistic. If d is the length

Protocol	Storage Overhead	Computation Overhead	Communication Overhead
Basic approach of Wang et al. [WBLW06]	$O(km)$	$O(km^2)$	$O(k)$
$COTA$	$O(c_1k)$	$O(c_2km)$	$O(k)$
EEW	$O(k)$	$O(k)$	$O(k)$

Table 1.2: Table of Comparison between protocols handling wormhole attack. k is the number of the hopcounts on the path, m is the number of wormhole detection packets examined, c_1 and c_2 are constants. c_1 and c_2 increase as sensitivity decreases. Large sensitivity leads to large number of false positives.

of a path between the source and the destination in terms of the distance traveled by a packet and r_{max} is the maximum communication range between any two nodes then the packet must travel at least $\lceil d/r_{max} \rceil$ hops. We show that if the length k of the path in terms of the number of hop-counts is less than $\lceil d/r_{max} \rceil$, then there is a wormhole on the path. Conversely, we show that if there is a wormhole on a path and the length of the tunnel is $\geq (\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$ then $k < \lceil d/r_{max} \rceil$ where $p = \frac{r_{max}}{r_{min}}$ and r_{min} is the minimum communication range between any two nodes. We assume that the maximum allowed communication range r_{max} is known to all the nodes in the network. It can be either incorporated statically at the time of setting up the network or flooded dynamically whenever a node with range greater than current r_{max} joins the network. Most of the times, however, the communication range is uniform and is known to every node. We check a malicious node from lying about its position by checking if two consecutive nodes on the path are in direct range of each other. All the checks are performed by the destination and intermediate nodes do not verify anything.

Through simulations, we show that our protocol is always able to detect the wormhole of length greater than the claimed bound ($(\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$). We also show that the protocol is able to detect and isolate wormholes of length much shorter than this most of the times. We also studied the effect of error (positive as well as negative) in the positions of the node on the wormhole detection capability and found that the effect was negligible.

1.2.3 Endangered nodes

Since most of the existing protocols do not handle all types of attacks it is imperative to find a solution that would reduce the impact of attacks on routing. **Endangered nodes** are defined as the nodes that are under the danger of attack. 'Minimum Exposed Path to Attack'($MEPA$) is the path which is farthest from the endangered nodes and hence is exposed minimum to the attacks. As threats on the Internet become increasingly sophisticated, we now recognize the value in controlling the routing of data in a manner that ensures security. However, few technical means for achieving this goal exist. In this paper we propose a method that allows users to specify nodes of the Internet they wish their data to avoid. Since the nodes in ad hoc networks are highly mobile, one would like to determine a path farthest from the prohibited area so that such a node does not even get around to the discovered path.

Our Contribution

To the best of our knowledge this problem has not been addressed earlier in the context of ad hoc networks. However, a related problem of computing a Maximal Breach Path in sensor networks has been addressed in [MKPS01, MLL03, HRS05]. We present an algorithm (henceforth referred as $MEPA$) to compute a $MEPA$ route in optimal $O(|P|)$ time where $|P|$ is the length of the $MEPA$ route.

We assume the existence of a mechanism that enables the nodes to detect the endangered nodes. In [ZL00, ZLH03, HL03] Lee et al. have used intrusion detection techniques to detect the presence of an intruder in ad hoc networks. Our algorithm works in two phases. Phase-I is a 'preprocessing' phase in which all the nodes compute their distances from the endangered nodes. Once the distances from the endangered nodes are computed, nodes can set up $MEPA$ routes in Phase-II. Due to the highly mobile nature of the nodes, distances may have to be updated dynamically as the nodes move. This is done in the 'maintenance' phase. The preprocessing step takes $O(D)$ time where D is the diameter of the network and then phase-II takes $(O|P|)$ time to compute the $MEPA$ route. When a node moves, maintenance phase takes $O(D)$ time to recompute the distances and then $(O|P|)$ time to recompute the $MEPA$ route. Assuming that the packets are received from the shortest path first, the algorithm computes a shortest $MEPA$ route.

We compared the performance of $MEPA$ with $RDVBS$ and $DENG$ for blackhole attack and, with EEW for wormhole attack. $MEPA$ outperforms all the three algorithms in terms of PDR and $AEED$. In terms of routing overhead, $MEPA$ performs better than $DENG$, $RDVBS$ and is comparable to EEW. Performance (packet delivery ratio and end to end delay) of our solution is comparable to that of $AODV$ in the absence of endangered nodes. Simulations also show that the $MEPA$ path is established in $O(|P|)$ time.

1.3 Thesis Outline

Remaining thesis is organized as follows: Chapter 2 discusses applications of ad hoc networks and challenges posed due to its characteristics. In Chapter 3 brief overview of various routing protocols in ad hoc networks are presented. Different types of attacks in ad hoc networks are presented in Chapter 4. In Chapter 5, we present our first protocol to mitigate the problem of attack by blackhole and selfish nodes . Chapter 6 presents the end-to-end protocol to secure ad hoc networks against wormhole attacks. Chapter 7 describes a linear time protocol to compute a $MEPA$ path. Conclusion of our work and directions for future work are presented in Chapter 8.

Chapter 2

Ad hoc Networks

2.1 Introduction

Mobile computing [Ch.94] has seen exponential growth in the past few decades. Mobile ad hoc networks consist of the devices that are autonomously self-organizing. In ad hoc networks the devices allow communication at low cost in a self-organized fashion and with easy deployment. The large degree of freedom and the self-organizing capabilities make mobile ad hoc networks different from other networking solutions. With new wireless applications added every few months, ad hoc networks become useful in many ways.

As wireless network devices and their applications become more familiar and available to a wider class of people, wireless networking applications will be seen independent of the availablity of Internet. For instance, people using laptop computers at a conference in a hotel might wish to communicate in a variety of ways without the mediation of routing through the global Internet. Wireless ad hoc networks allow mobile users with wireless devices to set up a possibly short lived network just for the communication needed at the moment. For example, a group of friends enjoying their holidays on a hill can share a piece of music through a network of their mobile phones.

Mobile applications are best suitable at places where necessary infrastructure is not available. In the absence of infrastructure, what is needed is that the wireless devices themselves take on the missing functions. Wireless computing devices should be able to communicate with each other even when no base station or Internet service provider can be found.

Making communications technology useful for people everywhere regardless of the nature or availability of backbone infrastructure is the aim of developing ad hoc networks. Requiring remote infrastructure for communication when people are within wireless range of each other is an unfortunate artifact of decades-old technology. Localized transmission technologies and local communication channel is the need of the hour. As this localized technology gains significant mindshare it is hoped that more spectrum will be allocated for local use given that the current ISM (industrial, scientific and medical) radio bands will not serve future needs.

As the technology promises to become increasingly present in everybody's life, it needs to be implemented across all wireless devices. Implementation of the technology is required across all the layers of the network stack. Thus, ad hoc networking forms an innovative and challenging area of wireless networking. These networks inherit the traditional problems of wireless and mobile communications, such as bandwidth optimization, energy consumption and transmission quality enhancement. In addition, the multihop nature and the lack of fixed infrastructure brings new challenges such as security, network configuration, device discovery and topology maintenance, as well as addressing and routing.

2.2 Commercial Applications of Ad hoc Networking

In this section we present some common applications of ad hoc networking as seen in a day-to-day life or otherwise.

Military Networks : Need for ad hoc networks was first realized for military networks. One of the original motivation for $MANET$s came from the U.S. Department of Defence (DoD). Air force networks between airplanes and navy networks between ships are examples of military ad hoc networks. The need for battlefield survivability and easy deployable solution in places like desert or jungles were the primary factors for the growth of ad hoc network. To survive under battlefield conditions, warriors and their mobile platforms must be able to move about freely without any restrictions of wired communications devices. Thus, wireless communication system for coordinating actions amongst the groups that operate in battlefield is required. In some regions, such as the desert or jungle, no terrestrial communication infrastructure is available. In other regions, access is unavailable because of destruction of local communications infrastructure. Thus a rapidly deployable, self-organizing infrastructure is required in such situations. $MANET$s provide a convenient solution for all these problems. Figure 2.1 depicts an army network set up on adhoc basis in a battlefield.

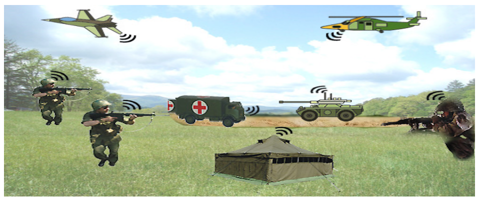

Figure 2.1: Army network set up on ad hoc basis in a battlefield

Emergency Services: Consider a scenario where the existing infrastructure is destroyed or is out of service due to natural calamity like floods and fire. Almost every year natural disaster wreak havoc with peoples' lives around the world. Tsunami in Indonesia in 2011, earthquake in Gujrat, India in 2001 and earthquake followed by tsunami in Japan 2011 are recent witness of these disasters. An ad hoc network needs to be set up to provide emergency services like food and medical aid under these conditions.

Conferencing : The support of business networks to the users of mobile devices who confer outside the normal office environment to collaborate on a project is often missing. In the current scenario where the projects involve people from broad range of industries their laptop devices need to communicate with each other to exchange ideas for discussion. An ad hoc network provides an ideal solution for the situation.

Personal Area Networks (PAN): Personal area network is a very localized network populated with some network nodes that are closely associated with a single per-son. For example a network consisting of wireless keyboard, mouse, monitor, CPU and printer. These nodes forming PAN may or may not need to have an attachment to the Internet but they will almost certainly need to communicate with each other. Moreover, mobility becomes important when interaction between several $PANs$ are needed.

Home Networking : In the modern days mobile devices like phones and laptops are becoming increasingly popular at home. Sometimes these devices need to communicate with each other; for example, one may wish to take a backup of one's mobile phone on his/her laptop. Playing multi-player game on laptops and mobile phones is another application of $MANET$s in a home environment.

Embedded Computing Applications: Intelligent solutions embedded in ubiquitous devices detect their environment, interact with each other and respond to changing environmental conditions. These devices are assumed to create future. For example Bluetooth short-range devices have been embedded in almost every wireless device these days. Ad hoc networking is likely to provide a flexible and convenient networking solution for interaction amongst these devices.

Sensor Dust : Suppose some hazardous chemicals were dispersed in an unknown manner because of an explosion or gas leakage. Instead of sending an emergency personnel it would be better to distribute sensors containing wireless transceivers. The sensors could then form an ad hoc network and cooperate together to collect and disseminate information about chemical concentration and identification.

Automotive PC Interaction : Another possible use of ad hoc networks is between the automotive computers, laptops or $PDAs$ that we may carry as we travel in our cars. These devices may be used to provide a link between our car and our home/office. For example a mobile phone may be used to switch on the air conditioner of our home before we enter. Alternatively when such a device comes in the range of home network, it may be used to program microwave timer, for say 15 minutes. A GPS (global positioning system) placed in our car may be used to find directions as we drive to a new city.

2.3 Challenges in Ad hoc Networks

Several concerns arise while establishing and maintaining $MANET$s. The advantages associated with an infrastructure based system are not available here. No part of network is dedicated to support any specific network functionality.

Lack of central controlling authority: Lack of central authority makes the system vulnerable to several threats. Distribution of keys by a central authority incorporates an element of security in infrastructure based systems. In ad hoc networks the solutions based on cryptographic techniques have to include methods to assign keys in a distributed manner. Decentralized decision making in $MANETs$ also needs cooperative participation of all nodes.

Limited Power Supply: As most of the times ad hoc networks are setup on fly, nodes rely on a battery for its power supply. Because of limited battery power nodes aim to minimize computation in order to save energy. Once a node's power supply drains out and it joins the network after recharge it needs to synchronize with other nodes.

Shared Medium : By the very nature of wireless communication, the channel is shared among several nodes in the network i.e. medium access control (MAC) is a fully distributed operation. This characteristic requires appropriate use of MAC protocols to allow secure and efficient use of the channel. The networks become prone to eavesdropping and the channel can easily be jammed or overused.

Scalability: Scalability is another issue which needs to be addressed while designing protocols for ad hoc networks. Nodes in ad hoc networks are often assumed to have IP addresses that are pre assigned in a way that is not directly related to their current position relative to the rest of the network topology. This differs from the way in which IP addresses are typically assigned to nodes in Internet. Routing within Internet depends on the ability to aggregate reachability information to IP nodes. This aggregation is based on assignment of IP addresses to nodes so that all the nodes on the same link share the same routing prefix. Ad hoc networks do not typically allow the kind of aggregation technique available to standard Internet routing protocols. Thus $MANET$s are vulnerable to scalability problems. In particular loss of aggregation leads to bigger routing tables.

Mobility and Fast changing topology : Mobility allows nodes to join and leave networks without any notice. Thus the network topology changes continuously and hence is unpredictable. It becomes difficult to have clear picture of the network membership at a given time. It is possible that a node leaves the path while it is being used to deliver data packets. A protocol must ensure that the packets so lost are delivered again. In case of wired networks the problem of broken link is not so frequent but in ad hoc networks where the nodes are highly mobile the problem needs attention. Also, the protocols which require knowledge of location may lead to more information exchange and hence would consume more bandwidth. Depending on the details of the algorithm, transmission of control messages may cause undesirable loads on the individual processing elements as well as on the available network bandwidth. For instance protocols that cause a recomputation of the entire network topology whenever a new routing update is received involve excessive computation on each node. In large networks such protocols may be subject to long convergence times whenever a node makes or breaks a link with its neighbors. The data in such a route update must be processed in far less time than the average time between network events caused by node mobility otherwise the network may never stabilize. By Murphy's Law such problems always arise at exactly the time when the communication is most critical.

Limited storage and computational capability : Most of the times the nodes in an ad hoc network are small in size thereby providing limited storage and computing power. Optimizing both storage and computation simultaneously poses a major challenge while designing services and protocols in ad hoc networks.

Protocol deployment and Incompatible standards : The need for standardization in routing protocols at the network layer is almost as great as the need at the lower protocol layers. When neighbors share the same physical medium and method of utilizing it (e.g., MAC layer using IEEE 802.11), communication within a local neighborhood is possible. Communication between nodes not in the same neighborhood (e.g., not sharing the same physical medium) will not be possible unless the nodes also agree on some higher-level protocol by which they can interchange connectivity information about links outside their respective neighborhoods. Like IEEE 802.11, some useful standards are needed for higher layers of ad hoc networks also, to ensure interoperability.

Wireless Data Rates : One can typically observe an order of magnitude difference in the speed of wired and wireless networks. For instance, while many enterprise users are accustomed to say 1 Gbps from the local Ethernet, wireless users must struggle to get a reliable 100 Mbps over the air. The wireless user has to be careful not to invoke applications that require a lot of bandwidth. As many of today's applications involve transactions over the Web, it may not be so easy for a user to avoid this. At any moment clicking hyperlink may attempt to load some beautiful dancing alphabetic letters on fire at great cost in bandwidth or at great cost in frustration as the user tries to figure out what went wrong.

Chapter 3

Routing in Ad hoc Networks

3.1 Introduction

Routing of packets is one of the basic functions performed in any network. Routing algorithms decide for a node whom to forward the packet. It uses a metric like delay, queue length and number of hops to take the decision. In wired networks, routing is performed by routers which are more powerful than a normal host. Router maintains a routing table in which it keeps the next hop for every destination on the basis of the best metric value. In ad hoc networks routing is performed by the node itself. Due to limited transmission range of the nodes, communication depends upon the co-operation amongst the neighbors. A routing protocol in $MANETs$ must be light weight as it executes on a node that is constrained in resources like storage, bandwidth and power supply. Since the nodes in $MANETs$ are highly mobile, a routing protocol must be able to maintain the routes quickly and with minimum overhead.

Several metrics are important to evaluate the performance of a routing protocol in $MANETs$. We list some of them here:

1. **Routing overhead**: Bandwidth consumed by the control messages is not available for data communication. Hence an efficient routing protocol is the one that does not generate too many control messages. In other words, routing overhead should be small.

2. **Computation overhead**: Algorithms that are complex require more processing cycles and hence consume more battery power. As the nodes in the network have limited battery life a protocol must be simple and lightweight.

3. **Central Controlling Authority**: Since the ad hoc networks are created on fly, most of the times there is no central controlling authority. Thus it is always recommended that a protocol be decentralized and distributed. That is the operations of a node are localized and rely only on the information collected by the node itself or by its 1-hop neighbors [L. 07].

4. **Topology dependent**: As the network topology is continuously changing, a routing protocol must be able to establish routes and maintain them quickly with the highly mobile nodes. The protocol should not be dependent upon the current network topology.

5. **Speed**: A route must be established quickly reducing the probability of a change in the network topology while the route is being established. It should reduce all the delays caused during route acquisition, buffering and processing at intermediate nodes and, retransmission delays at the medium access control (MAC) layer.

Following metrics have been used in the thesis to compare the performance of our solution with the existing approaches:

Definition 3.1.1 ***Packet Delivery Ratio****: The ratio between the number of packets originated by the CBR sources and the number of packets received by the CBR sink at the final destination.*

Definition 3.1.2 ***Average End-to-End Delay****: This is the average delay between sending the data packet by the CBR source and its receipt at the corresponding CBR receiver. This includes all the delays caused during route acquisition, buffering and processing at intermediate nodes, re transmission delays at the MAC layer, etc.*

Definition 3.1.3 ***Routing Overhead****: This is the ratio of the number of control packets generated to the number data packets transmitted.*

3.2 Routing Algorithms in Wired Networks

Some of the commonly used dynamic routing algorithms for wired networks are distance vector routing [R.E57], hierarchical routing [KK77], broadcast routing [DM78], link state routing [Per88] and multi cast routing [CRZ00, HCF$^+$01]. In this section we present distance vector routing (DVR) and link state routing (LSR), the most widely used algorithms amongst these. In the next section we will talk about some of the routing protocols for $MANETs$. Then we will discuss certain issues that need to be addressed in $MANET$s viz-a-viz routing in wired networks.

3.2.1 Distance Vector Routing (DVR) algorithm

Distance vector routing algorithm is an adaptive algorithm for wired networks in which the routing decisions change with a change in the topology of network. Each router x maintains a distance vector T indexed by a router in the subnet where i^{th} entry in the table stores the distance to the i^{th} router (from x) and the best next hop node (NHN) to reach i.

When a source router s wants to reach another router d, it searches its routing table for an entry for d. It forwards the packet to the neighbor k stored in the NHN field for destination d.

Distance vectors are exchanged periodically with neighbors to handle any change in topology. Good news travel fast but the bad news travel very slowly ('count-to-infinity' problem). The algorithm also causes formation of short-lived or long-lived routing loops thereby wasting lot of bandwidth. Routing loops are formed because next hop is chosen in a distributed fashion from the information exchanged which can be stale and therefore incorrect.

3.2.2 Link State Routing (LSR) algorithm

Like DVR, link state routing algorithm also maintains a routing table. Each router constructs a link state packet with the information about its neighbors and distance/delays from them. These packets are flooded in the network. Each router collects link state packets from all other routers in the subnet, constructs the subnet graph and executes Djikstra's best path algorithm to find out the next hop node on the shortest path for every destination in the network.

3.3 Routing Algorithms in Ad hoc Networks

In ad hoc networks each node behaves as a host as well as a router. Moreover due to small size and limited power supply a node has limited storage capacity and computing power. Thus a routing algorithm should be light in terms of processing and storage requirements. Moreover mobility of nodes in ad hoc networks makes the network topology unpredictable and continuously changing. It is possible that a node leaves the path while it is being used to deliver the data packets. A routing protocol must ensure that the packets so lost are delivered again. As packets are flooded in LSR and large routing tables are exchanged in DVR, they are not suitable for highly dynamic environment of ad hoc networks. Also, these protocols maintain and search a routing table with an entry for every node in the network. The size of this table increases the storage and processing requirement when the size of the network is large. These protocols also suffer from routing loops which result in wastage of bandwidth. Considering the issues of routing in ad hoc networks, these algorithms have been modified [PB94, BOT99, JHC$^+$01, BR02, BBO03, ZDF06] to suit the requirements of ad hoc networks. DVR has been modified with two types of routing protocols for ad hoc networks: table-driven protocol and on-demand routing protocol. Table driven protocols are proactive in nature in the sense that, they compute and maintain a path for every node in the network. On the other hand on-demand routing protocols establish path only when required. Though on-demand protocols incorporate a slight delay as compared to table-driven protocols they save other resources like bandwidth to respond better to changes in the topology. The on-demand routing protocols exchange routing information only when needed. The Destination Sequence Distance Vector $(DSDV)$ routing protocol is a table-driven protocol whereas Dynamic Source Routing (DSR) and Ad-hoc On demand Distance Vector $(AODV)$ are on-demand routing protocols.

3.3.1 Table Driven Routing

In this section we present 'Destination Sequence Distance Vector' the most commonly used table driven routing protocol in $MANETs$.

Destination Sequence Distance Vector$(DSDV)$ algorithm

To handle routing loops in DVR, Perkin and Bhagwat [PB94] proposed 'Destination Sequence Distance Vector' routing algorithm in which they introduced a field called *destination sequence number* in the routing table. A node initiating a packet generates a *sequence number* and includes it in the packet for others to know its *sequence number*. A node having an entry for another node say d as destination also stores its *sequence number* . It stores an even *sequence number* for a valid link and an odd number for a broken link. Thus, a node always generates an even *sequence number* for itself. *Sequence number* is then used to check the freshness of the information.

Consider the network in Figure 3.1. Let the link between the nodes B and C goes down at some point of time and B has not yet informed the node A about the breakage. Suppose node A transmits a message to C via B. In DVR, routing loop will be formed since node A will transmit the message to node B assuming that the link A-B-C is operational and is of lowest cost. Node B knows of the broken link and will try to reach node C via node A; thus, it will send the original message back to node A. Furthermore, when node A receives the message back from node B, it will consult its routing table. Since it has not been informed of the broken link, it will send the message back to B creating an infinite loop. In $DSDV$, when B sends packet back to A, it will send to A a packet not only with C as destination but it also sends C's sequence number i.e. say 47, which is odd. Hence A will come to know that link between B and C no longer exists and B is trying to send packet to C through it. Hence the loops will not be formed in $DSDV$. By avoiding the formation of routing loops $DSDV$ saves a lot of bandwidth. Further instead of sending the entire table $DSDV$ sends only the incremental updates thereby further reducing the bandwidth consumption. However, $DSDV$ still maintains routing table for all the nodes in the network and hence is not suitable for large networks. Also, $DSDV$ requires a regular update of its routing tables, which consumes battery power and a small amount of bandwidth even when the network is idle. Also, whenever the topology of the network changes, a new sequence number is necessary before the network re-converges.

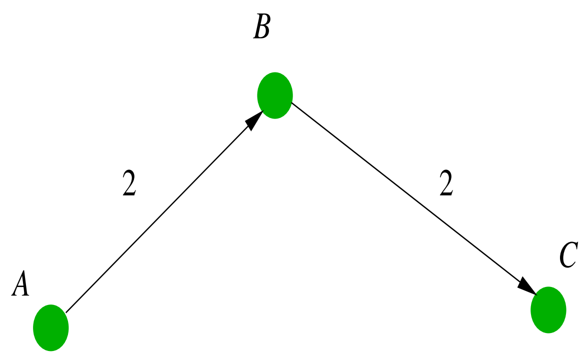

Figure 3.1: An example network to show routing loops in DVR

3.3.2 On-Demand Routing

In this section we present 'Dynamic Source Routing', and 'Ad hoc On-demand Distance Vector' the most commonly used on-demand routing protocols in $MANETs$. On demand routing protocols generate route request when a node wants to communicate with another node. The route request is broadcast to other nodes and a node having a path to destination replies with a route reply.

Dynamic Source Routing (DSR) Algorithm

The 'Dynamic Source Routing' protocol is an on demand routing protocol for ad hoc networks. It assumes that the links are uni-directional. It works in two phases: the $routediscovery$ phase and the $routemaintenance$ phase. Whenever a node wants to communicate with another node, it checks its $routecache$. If a path is available it sends the data packets on that path otherwise it initiates route discovery. In route discovery, it broadcasts $RouteRequest$ packet to its neighbors. Neighbors further broadcasts the request to their neighbors. Each node on the path appends its *id* to the request packet. If the node itself is mentioned in the list, it discards the packet thereby avoiding routing loops. Also, if an intermediate node has a path to destination, it sends reply to the source with $recordlist$ of complete path. Otherwise, when a node is neither mentioned in the list nor it has a path to the destination, it forwards the request to its neighbors until the request packet arrives at the destination. When the destination finally receives the $RouteRequest$ packet it checks its $routecache$, if it finds a path for the initiator node, it sends $RouteReply$ packet via this path copying the $recordlist$ of the path from the $RouteRequest$ packet. Otherwise, when it does not find a path in its $routecache$, it generates another route request for source and piggyback $RouteReply$ to the initiator in this $RouteRequest$. Path discovery completes when $RouteReply$ reaches the originator.

For path maintenance each node periodically broadcasts a $HELLO$ message to its neighbors. Though DSR saves bandwidth by establishing the route on request it wastes bandwidth in storing the entire path in the request packet and in the reply packet.

Ad hoc On-demand Distance Vector $(AODV)$ algorithm

As the name suggests $AODV$ is also an on-demand routing protocol. It assumes bidirectional links. It works in a similar manner as DSR except that the intermediate nodes do not append their id's to the request packet. Only a hop-count field is maintained in the routing tables and the request packet instead of the entire path. As in DSR an intermediate node upon receiving a $RouteRequest$, checks its routing table. If a fresh route is found it replies to its neighbors. It continues until either $RouteRequest$ reaches an intermediate node with a path to the destination or it reaches the destination itself. When the destination receives the $RouteRequest$, it replies with a $RouteReply$ packet. The path is established when the source node receives the $RouteReply$ packet. As the links are bidirectional in $AODV$ a reverse path for the source is created at every node as the $RouteRequest$ travels from the source to the destination. The reply packet (whether from the destination or from the intermediate node) follows the reverse path of the request packet. At each intermediate node on the reverse route of the route reply packet, a forward path to the destination is constructed or updated as the case may be.

As in $DSDV$, sequence number for each node is used to check the freshness of information hence routing loops are avoided. As the routes are established only on demand the size of routing tables are small which saves the search time. Since the routing tables are not exchanged it saves a lot of bandwidth. Thus $AODV$ is most widely used routing protocols in ad hoc networks.

3.4 Security Issues

Lack of central authority and wireless links make routing in $MANETs$ vulnerable to several threats in security.

1. Wireless medium : The medium of transmission in ad hoc networks is wireless. Unlike wired networks, the attacker does not have to physically break into a machine of the network or wiretap a cable. Once the intruder gets access to the network it can easily intercept the transmitted data without the sender even knowing. Furthermore, due to the limitations of the medium, communications can easily be perturbed; the intruder can perform this attack by keeping the medium busy sending its own messages or, just by jamming communications with noise.

2. Nodes serving as a router: Due to limited transmission range, nodes rely on their neighbors to route their messages to the destination. Thus nodes in $MANETs$ serve as hosts as well as routers and routing is performed in a multi-hop manner. Since the packets pass through a series of nodes, a malicious node can eavesdrop a packet or disrupt the normal routing operations of the network by modifying or dropping packets or replaying stale routing information in the absence of security measures.

3. Lack of Central Authority: As there is no central authority security measures like firewall cannot be installed in $MANETs$. Cryptographic schemes which are most successful in wired networks are hard to implement in $MANETs$ as they require a central authority to assign keys to the nodes in the network.

3.5 Security Goals

The ultimate goals of the security solutions for $MANETs$ are to provide security services, such as availability, confidentiality, integrity, authentication and nonrepudiation to its users. In order to achieve this goal, the security solution should provide complete protection spanning the entire protocol stack. **Availability** is concerned with the unauthorized holding of resources. For example, on the physical and medium access control layers, an adversary could employ jamming to interfere with communication on physical channel while on network layer it could disrupt the routing protocol and continuity of services of the network. **Confidentiality** ensures that certain information is only readable or accessible to the authorized party. Basically, it protects data from passive attacks. **Integrity** guarantees that only the authorized parties are only allowed to modify the information or messages. It also ensures that a message being transmitted is never corrupted. A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent with no duplication, insertion, modification, reordering, or replays. **Authentication** ensures that the access and supply of data is done only by the authorized parties. It is concerned with assuring that a communication is authentic. **Nonrepudiation** prevents the sender as well as the receiver from denying a transmitted message. Thus, when a message is sent the receiver can prove that the message was in fact sent by the alleged sender. When node A receives an erroneous message from node B, nonrepudiation allows A to accuse B using this message and to convince other nodes that B is compromised.

Chapter 4

Attacks on Routing Protocols in Ad hoc Networks

4.1 Introduction

As discussed in chapter 3, lack of infrastructure and wireless medium makes routing in ad hoc networks vulnerable to several types of attacks. In this chapter we will discuss some of these attacks. Attacks in $MANETs$ can be classified on the basis of their effects on the network, behavior of attacking node, the origin or, the method of performing attack using vulnerabilities and exposures in the network.

On the basis of effect on the network, attacks can be classified as active or passive. In passive attacks the attacker snoops the data exchanged in the network without altering it. This leads to attack on the confidentiality of the data. Detection of passive attacks is difficult as it does not tamper with the normal functions of the network. In active attacks the attacker is more active in disrupting the normal functioning of the network. An attacker may jam the network completely or partially making one part of the network unreachable from the other part. It may modify, spoof or replay packets.

An attacker may be classified as internal or external depending upon whether the attacker is a compromised node within the network or it does not belong to the network. Internal attacks are difficult to detect as the compromised nodes are the authorized nodes of the network and hence posses the secret keys required to authenticate the messages.

However, at the same time, internal attacks are difficult to launch. As external attacks are easier to launch they occur more frequently and hence they need attention.

Attacks have been classified in various ways on the basis of methods used by an attacker to perform the attack. Modification is a method in which an unauthorized party tampers an asset. For example a malicious node can redirect the network traffic and conduct denial-of-service (DoS) attack by modifying message fields or by forwarding routing message with false values. Spoofing or masquerading as some good node is another weapon used by malicious nodes to launch different types of attacks. Fabrication is a method in which an unauthorized party not only gains access but also inserts counterfeit objects into the system. In $MANETs$, fabrication is used to launch attacks by generating false routing messages.

An attack in a network may be at application layer, medium access layer (MAC), transport layer, network layer or physical layer. The most common physical layer attacks in $MANET$s are eavesdropping, interference and jamming. The wireless radio signal is easy to jam or intercept. Moreover an attacker can gain access to the wireless medium and overhear the transmission. MAC misbehaving is one of the common attacks at the MAC layer. In this attack a malicious node does not follow the rules of the MAC protocol. The MAC layer in ad hoc networks is typically based on $CSMA/CA$ (carrier sense multiple access/collision avoidance) protocol. In this protocol a node senses the channel, waits for sometime using back off exponential algorithm if the channel is busy and transmits only if the medium is free. A malicious node may start transmitting without waiting causing other nodes to continually back-off. In $MANETs$, the nodes also function as routers that discover and maintain routes to other nodes in the network. Establishing an optimal and efficient route between the communicating parties is the primary concern of routing protocols in $MANETs$. An attack on routing may disrupt the overall communication and hence paralyze the entire network. Network layer vulnerabilities fall into two categories: routing attacks and packet forwarding attacks [YLY$^+$04]. The family of routing attacks refers to any action of advertising routing updates that do not follow the specifications of the routing protocols. The specific attack behaviors are related to the routing protocol used by the $MANET$. These attacks are discussed in detail in the next section. Packet forwarding attack is related to disrupting and forwarding of data packets. The transport layer protocols provide end-to-end connection, reliable packet delivery, flow control and congestion control. Like TCP (transmission control protocol) in the Internet model nodes in a $MANET$ are also vulnerable to the SYN flooding and session hijacking attacks. Like other layers application layer is also vulnerable and attractive for an attacker to attack. The main attacks in application layer are malicious code attacks (like virus, worm etc) and repudiation attacks.

4.2 Attacks against Routing Layer

Most routing protocols in ad hoc networks rely on implicit trust-your neighbor relationship to route packets. This naive trust model allows attackers to paralyze the network in various ways. Malicious nodes attack by inserting erroneous routing updates, replaying old routing information, changing routing updates, or advertising incorrect routing information so that the network is not able to provide service properly. Attacks like reducing the amount of routing information available to other nodes, failing to advertise certain routes or discarding routing packets or parts of routing packets are due to selfish behavior of a node. Selfish nodes do not directly damage other nodes but their effect cannot be underestimated. Whatever the attacks are, an attacker exhibits its actions in the form of refusal to participate fully and correctly in routing protocols according to the principles of integrity, authenticity, confidentiality and cooperation. Several types of attacks against the routing layer in ad hoc networks have been discussed in literature. Some of these (blackhole or greyhole attack, rushing attack, wormhole attack) cripple the network by disrupting the route of the legitimate packets while others (flooding attack) inject too many extra packets in the system thereby consuming system resources like bandwidth, memory/computational power of nodes.

4.2.1 Types of Attacks

Functioning of network in routing layer may be disrupted due to **malicious behaviour** [Per88, YKL05] of an external/internal node or due to **selfishness** of a node whose aim is not to disrupt the network but rather it wishes to save power due to limited power supply. Depending upon how an attacker performs its malfunctioning behaviour in the network following attacks have been defined:

1. Flooding and Denial of Service attack

   In flooding attack an attacker attempts to consume battery life and computing power of the victim nodes by generating bogus packets to the victim node. The victim nodes are kept busy processing the bogus packets thereby preventing the nodes to perform the legitimate task of route discovery and packet forwarding. The attack can be against reactive (on-demand) as well as proactive routing protocols since both require broadcasting the control packets in the network. In proactive protocols control packets comprise of the routing table updates while in reactive protocols these are route request packets. In [PYY$^+$06] authors have defined two types of flooding attacks against on-demand routing protocols namely $RouteRequest$ flooding attack and $Data$ flooding attack. In $RouteRequest$ flooding, the attacker generates several route request packets for different destinations. For a destination, it either uses an IP address which is not used in the network or selects a random IP address depending upon the knowledge about the scope of IP addresses in the network. In $Data$ flooding attack, attacker clogs the network by injecting too many data packets through earlier established paths to various nodes. Authors have also proposed a mechanism to prevent the flooding attack in the $AODV$ protocol. In this approach, each node monitors and calculates the rate of its neighbors' $RouteRequest$. If the $RouteRequest$ rate of any neighbor exceeds a predefined threshold, it blacklists the neighbor. It then drops any future $RouteRequests$ from the blacklisted nodes. A drawback of this approach is that if a malicious node impersonates a legitimate node and broadcasts a large number of $RouteRequests$ on its behalf, other nodes will blacklist the legitimate node. In [DB05] also authors have proposed a similar scheme. However, in their scheme the threshold is adaptive based on statistical analysis. The advantage of this approach is that it can reduce the impact of the attack for varying flooding rates.

   Flooding attack against proactive protocols is also called $routing\ table\ overflow$

attack. Proactive routing algorithms keep routing information even before it is needed. In this case, an attacker can simply send excessive route advertisements for non existent nodes to the neighboring nodes in the network. The goal is to overwhelm the routing table with fake routes so as to prevent the new routes from being created. Yan et al. [YZV03] proposed a trust based mechanism which handles routing table overflow attack. According to the proposed protocol every node has a right to ignore or reject route serving or data receiving according to the trust and ability evaluation.

2. Replay Attack

   In replay attack a valid transmission is repeated or delayed. An attacker can perform a replay attack by recording old control messages and re-sending them later to make other nodes update their routing tables with stale routes. General authentication mechanisms cannot prevent replay attack. Sequence numbers were introduced in [PB94] to handle routing loops formed due to stale information in the routing table. A node can also spoof using replay attack: Suppose any mobile node A wants to prove its identity to B. B requests A's password as proof of identity, which A provides (possibly after some transformation like a hash function or using some digital signature); at the same time, C is eavesdropping the conversation and keeps the password. After the interchange is over, C connects to B presenting itself as A. When asked for a proof of identity, C sends A's password read from the last session which B accepts. Hence C starts communicating with B disguising itself as A. In $AODV$ replay attack can cause old route request packets to be replayed again which provokes unnecessary rounds of route discovery. Zhen et al. [ZS03] have proposed secure routing for preventing replay attack on $AODV$ protocol using technique based on strengthening the neighbor authentication. They extend $AODV$ protocol by introducing some control packets for attack discovery process. Winjum et al. [WMKS05] have proposed protocols to handle replay attack against optimized link state routing protocol ($OLSR$) in ad hoc networks.

3. Message tampering

If a message is not appropriately protected an attacker can modify the message originating from other nodes before relaying them. Message tampering attack can also be against proactive as well as against re-active routing protocols. In on-demand protocols attacker can include itself into the path by reducing the hop count value in the reply message. Ariadne [HPJ02], a secure on-demand routing protocol relies on symmetric cryptography to prevent attackers from tampering the reply messages. It uses a key management protocol called $TESLA$ that relies on synchronized clocks. $SEAD$ (Secure Efficient Ad hoc Distance Vector routing protocol) [HJP02] mitigates message tampering attack against $DSDV$ using one-way hash function. Song et al. [SWLK03] proposed to use a tamper resistant module (TRM) to protect routing module from both compromised and malicious users. They have defined TRM as hardware/software entity in which data and program can not be modified by the user.

4. Blackhole Attack

   Black hole attack is defined for on-demand routing protocol. In blackhole attack a node responds positively to a request for a shortest route even though it does not have a valid route to the destination node. Once included in the path, the attacker drops instead of forwarding the data packets making a blackhole there. The node is called the black node/blackhole node. Since a black node does not have to check its routing table it is the first one to respond to the route discovery request in most cases. Blackhole attack can be co-operative where several black nodes coordinate to launch the attacks with each other increasing the severity of the attack.

   Randomized route selection [TS07], Cross-validation of intermediate nodes [DLA02, YM06, KGA06] and Watchdog mechanisms [MM00, PM03] are some of the techniques used for handling blackhole attack in ad hoc networks. These approaches are discussed in more detail in chapter 5.

5. Greyhole Attack

   In contrast to blackhole attack where an attacker drops all the data packets, in grey hole attack the attacker selectively drops and forwards the data packets after it advertises itself as having the shortest path to the destination in response to a route request message from a source node. The attacker drops the intercepted packets with a certain probability. Grey hole attack can also be co-operative involving multiple nodes. Agarwal et al. [AGD08] and Banerjee [Ban08] have proposed mechanisms to detect greyhole attack. The algorithms are based on sending data in equal but small sized blocks instead of sending whole of data in one continuous stream. It checks the reliability of the path by taking a feedback between the transmission of two blocks of data.

6. Wormhole

   In wormhole attack, two nodes located distantly collaborate to create a fast tunnel [HPJ03a, KBS05]. The tunnel may be created using encapsulation or out-of-bound channel. The two nodes at the end of the tunnel get an illusion that they are only one hop away. Since route discovery in $MANET$s rely heavily upon the neighborhood set at each node, false information about a node's neighbor can severely affect the discovered route. A single node alone using high power transmitter can also establish this attack. If the routing protocol uses the number of hopcounts to compute the shortest path, it establishes a path through the wormhole. If the routing protocol uses the round trip delay to compute the shortest path and there exists a fast transmission path between the two ends of the wormhole, it prevents normal multi-hop routes to be discovered since the tunneled packets travel much faster through the wormhole than through the normal route. Hence in either case the route is established through the wormhole. Once a route has been established through malicious nodes it may drop or compromise packets. In [KW03] author presented significant impact of wormhole on both proactive and reactive ad hoc routing protocols. Several methods using directional antenna [HE04], graph theoretic approach [MGD07], packet leashes [HPJ03a] and others [WBLW06, GK08, KG08] have been proposed to handle wormhole attacks. These approaches are discussed in more detail in chapter 6.

7. Rushing Attack

In [HPJ03b] Hu et al. introduced rushing attack against on demand routing protocols which discover the fastest route to the destination. In most of the on-demand protocol, an intermediate node accepts and forwards only the first route request and discards others; the rushing attacker exploits this feature to implant the attack. In rushing attack, the attacker quickly forwards the route request packet, faster than any legitimate node can do. A legitimate node maintains a time difference between the received route request packet and the forwarded route request packet to avoid collision. However, a rushing attacker may rush a route request immediately without maintaining this time difference. As a result, the request from the attacker is the first one to reach the destination and hence the discovered route includes the rushing attacker. To mitigate the attack, Hu et al. proposed that instead of keeping the first route request and discarding the remaining, an intermediate node collects n route requests and, selects and forwards one of them randomly.

8. Spoofing or Impersonation

   In spoofing, an attacker assumes the identity of some good node in the network thus receiving messages directed to the node it fakes. This is done by misrepresenting an IP or medium access control (MAC) address. Usually this would be one of the first steps for an attacker to intrude into a network with the aim of carrying out further attacks to disrupt the normal operations of the network. The attacker could obstruct proper routing by injecting false routing packets into the network or by modifying routing information. Sometimes an attacker might find it advantageous to selectively forward packets. As described in [KW03], an intruder with this goal will most likely try to impersonate a node within the path of the data flow of interest. It could achieve this by modifying routing data or implying itself as a trustworthy communication partner to neighboring nodes in parallel. A compromised node may also have access to encryption keys and authentification information. Depending on the access level of the impersonated node, the intruder may even be able to reconfigure the network so that other attackers can (more) easily join or he could remove security measures to allow subsequent attempts of invasion. Exploiting the loopholes in the MAC layer protocol an attacker could place its node between two other nodes communicating with each other (man-in-the-middle attack).

Some attacks like sybil attack [Dou02, NSSP04, PSL06] use spoofing to implant the attack. In sybil attack, a node assumes multiple identities and pretends as several nodes. Stealth attack was introduced by Jakobsson et al. in [JWY06] in which an attacker can partition a network, reduce its goodput, hi-jack and filter traffic to and from a node and thereby eavesdrop and perform traffic analysis. Stealth attacker uses impersonation to partition the network at minimum cost and visibility thereby disrupting the normal routing function in the network. As the cost incured is low, it is easy to implant such an attack and since the visibility is poor, it is difficult to detect. Jakobsson et al. proposed ‘reputation based control’ to augment the existing routing protocols in order to immunize them against stealth attacks. Sinkhole attack is another example of an attack that is implanted by an attacker using impersonation. In sinkhole attack, the attacker assumes the identities of its neighbors to attract all the network traffic towards itself creating a sink there. Multipath [GL01] routing and probabilistic [YJWA02] routing have been proposed against handling sinkhole attack. However, by using good authentication algorithms, strong data encryption and secure routing protocols, the effects of impersonation can be reduced significantly [Bur03].

9. Attacks due to selfish nodes:

As mentioned earlier, battery power is an important resource in $MANET$s. A node which does not co-operate in packet forwarding to save its energy for its own purpose is known as a selfish node. Michiardi et al. in [MM02b] introduced three types of selfish nodes depending upon its extent of co-operation: those who forward only the control packets, those who forward only the data packets, those who take a decision depending upon its current energy levels. Though the purpose of a selfish node is not to cripple the network the performance of the network degrades significantly in presence of selfish nodes. Several reputation and incentive based solutions have been proposed in literature to detect and isolate selfish nodes. Reputation based schemes need a node to switch to promiscuous mode while incentive based schemes need a special hardware to implement gain and loss of incentives.

Chapter 5

Reliable Distance Vector routing algorithm to mitigate Blackhole and Selfish nodes

5.1 Introduction

In this chapter, we propose a co-operative security scheme which we call 'Reliable Distance Vector routing protocol to handle Blackhole and Selfish nodes ($RDVBS$)' to mitigate attacks by blackhole and selfish nodes. $RDVBS$ provides a foundation for secure operations with little impact on existing protocols and can be used in bandwidth constrained nodes. The existing approaches to assuage the impact of blackhole nodes do not handle selfish nodes and the ones to mitigate selfish nodes do not alleviate the blackhole nodes. We present first such approach that handles both black hole attack as well as selfish nodes in a single scheme. The proposed solution mitigates multiple black holes also. We use cross-verification but do not flood the verification packets. The protocol is based on $AODV$ protocol with the assumption that nodes cannot impersonate and all other network conditions are good. It behaves like $AODV$ in the absence of attack and, detects and isolates misbehaving nodes in presence of attack. The scheme allows the network to recover from the attack when a misbehaving node leaves the network or becomes good. It does not incur too much overhead as we do not flood the cross-verification packets nor does it require the nodes to listen in promiscuous mode. As the approach is deterministic attacks are detected with 100% success. The protocol does not require any fixed infrastructure as is required to implement virtual banks in incentive based schemes. We compare our algorithm with Deng et al.'s algorithm as it also uses cross-verification and show that our algorithm outperforms theirs in terms of routing overhead without affecting other parameters like end-to-end delay and packet delivery ratio.

For the ease of understanding we present our proposed solution in two steps. First we present reliable distance vector routing protocol to handle blackhole ($RDVB$) only. In the next section we extend this algorithm to reliable distance vector routing protocol ($RDVBS$) to mitigate blackhole and selfish nodes also.

5.2 Problem Statement

Blackhole attack is an active attack in which a node responds positively to a request for a shortest route even though it does not have a valid route to the destination node. The node is called **black node or blackhole node**. Since a blackhole node does not have to check its routing table it is the first one to respond to route discovery request in most cases. When data packets reach the black node it drops the packets rather than forwarding them to the destination creating a blackhole there. We call these nodes as blackhole nodes of **type 1**. Blackhole attack can be co-operative involving multiple nodes acting in coordination with each other.

Bharat Bhargava [Bha02, WLB03] defined **blackhole** attack as *false destination sequence* attack also. In this, blackhole node responds positively to a request for a shortest route with a very high sequence number of destination. Source considers this path as freshest path and starts sending data packet through it. Again blackhole node drops the packets rather than forwarding them to the destination. We call these nodes as blackhole nodes of **type 2**. Sometimes attacks like reducing the amount of routing information available to other nodes, failing to advertise certain routes or discarding routing packets or parts of routing packets are due to selfish behavior of a node. As the supply of power is limited, sometimes a node may wish to use its power supply for its own purposes and

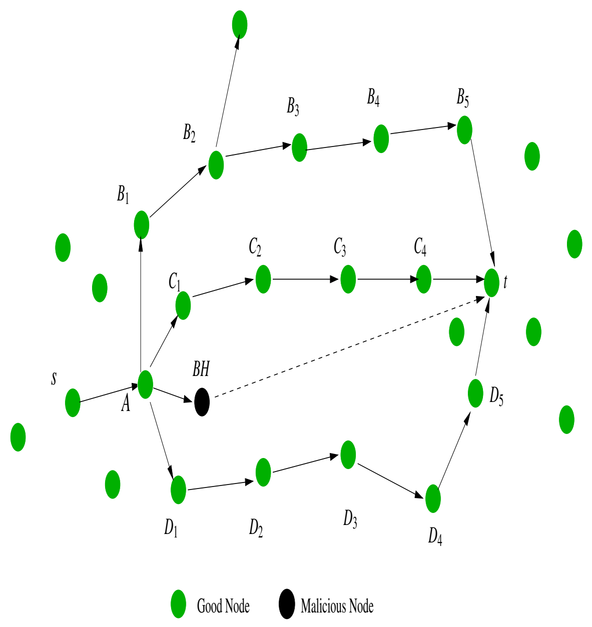

Figure 5.1: Blackhole node of type 1

hence does not participate in routing operations. Such nodes are called **selfish nodes**. Selfish nodes were first discussed in [HSSS04]. Here we define two types of selfish nodes depending upon their extent of non-cooperation in network operations.

1. Selfish node of Type 1 uses energy only for its communication and it forwards neither control packets nor data packets.

2. Selfish node of Type 2 forwards control packets but does not forward data packets. Here we make an assumption that once a node stops forwarding data packets, it does not involve itself into route establishment also. Let E be the initial maximum energy of a node. When the energy of the node falls within $(T1, E]$ the node behaves properly and executes both routing functions and packet forwarding. When energy falls in $(T2, T1]$ the node forwards control packets but disables data packet forwarding. Since now the node no longer wants to participate in data packet forwarding and its intention is not to disrupt the normal functioning of the network it is legitimate to assume that it will no longer participate in route establishment until its energy is restored. With in a limited time interval the node is recharged and its energy level is set back to the initial value.

5.3 Impact of blackhole and selfish nodes

Once a route is established through a blackhole node, it drops the data packets as it does not have a valid path to the destination. As a result the network throughput degrades considerably. Parsons et al. [PE09] showed the impact of various attacks on $AODV$. In particular they showed that Packet Loss Ratio (PLR) increases from $.13$ (in the absence of an attack) to more than $.5$ when a blackhole attacker is present. They also showed that the routing overhead increases significantly as the number of attackers increase. We also observe in our work that the packet delivery ratio (PDR) of $AODV$ falls from .97 (in the absence of attack) to about .24 when an attacker (blackhole node) is present. A blackhole attacker can also drop received routing messages instead of relaying them, as the protocol requires, thereby making the destination unreachable. The attacker can also store the data and perform traffic analysis.

Several authors [MM02b, KKSW04] have studied the impact of selfish nodes on dynamic source routing (DSR) algorithm. Michiardi et al. [MM02b] showed that the PDR of the algorithm drops by 60% when 50% of the nodes of the network are selfish. Further they pointed out that the PDR degrades by $10\% - 15\%$ every time the percentage of selfish nodes increases by 10%. They also showed that end-to-end delay increases linearly with the percentage of selfish nodes in the network. Kargal et al. [KKSW04] also showed that the performance (delivery ratio) of DSR degrades significantly as the number of selfish nodes increase in the network. We studied the impact of presence of selfish nodes on $AODV$(Figure 5.2). We show that the packet delivery ratio of $AODV$ drops by about 55% when 50% of nodes are selfish and it degrades by 10%-20% with increase of 10% in the number of selfish nodes.

5.4 Related Work

Approaches to assuage the impact of blackhole attacks either use cross-verification [DLA02, YM06, RFN05, TS08, Ban08, AGD08] or are based on watchdog mechanism [MM00, PM03]. In [DLA02] the source node verifies the authenticity of the intermediate node (IN) sending the *RouteReply* from its nexthop node (NHN). It does so by broadcast-

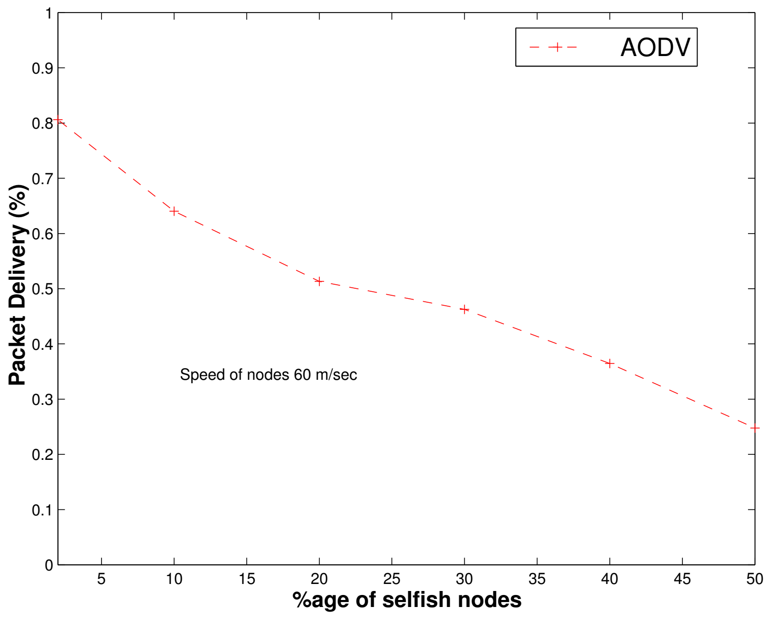

Figure 5.2: Packet Delivery Ratio of $AODV$ with varying percentage of selfish nodes

ing a *FurtherRequest* packet to the NHN to verify if it has a route to the destination. In [YM06] instead of the source node, the previous hop node broadcasts a verification packet to the NHN. Most of the approaches using cross-verification flood the verification packets and hence incur a lot of communication overhead. Watchdog mechanisms require the nodes to listen to their neighbor nodes in promiscuous mode. Switching the mode from promiscuous mode to transmit/receive mode is not easy and is error prone [KKSW04]. In some approaches [TS07, SYP04] the source node waits for some time, collects some paths and selects the one that shares at least one node with at least one more path. It is based on the hypothesis that if two paths share a node, it is unlikely that it is under attack. The approach suffers with the delay in establishing the route besides the fact that the probability of a blackhole node on the path is non-zero. Some researchers have also proposed Intrusion Detection System (IDS) and learning theory approaches to mitigate blackhole attack [KNK$^+$07, HL04, HFLY03, RFdAG08]. These approaches are compute-intensive and incur large storage and communication overhead as they collect and analyze large amount of data for anomaly detection. Sun et al. [SGCW03] proposed a mechanism to mitigate impersonation where an attacker impersonate as the destination to launch a blackhole attack. Some approaches have been proposed to handle blackhole attack launched by specifying false sequence numbers [Bha02, KNK+07]. They handle blackhole attack of type 2.

Most of the work to diminish the effect of selfish nodes either propose a reputation based trust system [WSST05, YZV03] or are based on providing economic incentives. Reputation based systems either rely on first hand information to build reputation or use second-hand information gathered by other nodes. Though using second-hand information results in building the reputation quickly, it suffers with the drawback of spreading rumors. To handle this CORE (a COllaborative REputation Mechanism) [MM02a] allows sharing of only positive behavior which makes it vulnerable to positive ratings by malicious nodes. By sharing only negative reputation CONFIDANT (Co-operation of Nodes Fairness In Dynamic Ad-hoc Networks) [BB02a] reduces the false praise but makes the system vulnerable to false accusations. Context-aware detection [PW02] accepts negative advertisement provided it is claimed by some threshold number of nodes else it is considered as misbehavior. It checks false accusations but at the same time also discourages legitimate reporting of misbehavior especially in sparse networks. DRBTS(Distributed Reputation-based Beacon Trust System) [STW06], CONFIDANT [BB02b], SORI (Secure and Objective Reputation-based Incentive scheme) [HWK04], RPA (Reputation Propagation and Agreement) [LY02] and RFSN (Reputation-based Framework for High Integrity Sensor Networks) [GS04] use both positive and negative information but use different weight functions to different type of information. OCEAN (Observation-based Cooperation Enforcement in Ad Hoc Networks) [BB03] and Pathrater [MM00] use only first hand information to check the rumors but it takes long for the reputation to fall. Whatever be the strategy, reputation based mechanisms either suffer with the danger of spreading rumors or positive ratings by co-operating malicious nodes or gaining high-reputation and trust by a malicious node and staying in the system. Moreover, most of these approaches require neighborhood monitoring in promiscuous mode. Refaei et al. [RSDE05] proposed a reputation based trust mechanism which does not depend upon the reputation information exchange but rather takes the feedback from the destination (e.g. by TCP acknowledgement) to raise the reputation index of its next hop neighbor on successful delivery of the packets. The approach suffers with the drawback that the presence of a selfish node down the path may lead to penalizing a good neighbor.

Incentive based schemes [BH00, BH01, BH03, JHB03, ZCY03] treat packet forwarding as a service that can be priced and introduce some form of virtual currency to encourage packet forwarding. [BH01] introduces 'Incentives to co-operate' scheme which uses a virtual currency called $Nuglets$ in every communication. $Nuglets$ serve as a per-hop payment for every packet forwarding. They are incremented when a node forwards for others and decremented when it sends packets for themselves. Thus a node exhibiting selfish behavior is penalized appropriately. Authors propose two conceptual models for charging the packet forwarding service. In the first one, called Packet Purse Model (PPM) the source of the packet is charged, whereas in the second one, called Packet Trade Model (PTM), the destination is charged. A hybrid solution is the one in which both source and destination are charged according to the requirement. In PPM, the source node loads the packet with a number of nuglets sufficient to reach the destination. Each forwarding node acquires some nuglets from the packet that covers its forwarding costs. The exact number of nuglets charged by the forwarding nodes may depend upon many things including the amount of energy used for the forwarding operation, the current battery status of the forwarding node, and its current number of nuglets. If a packet does not have enough nuglets to be forwarded then it is discarded. In Packet Trade Model, the packet does not carry nuglets, but it is traded for nuglets by intermediate nodes. Each intermediary buys it from the previous one for some nuglets and sells it to the next one for more nuglets. These schemes (incentive based) require tamper-proof hardware so that the correct amount of credit is added or deducted from a node [BH00] or require virtual banks [BH03, JHB03]. There are arguments that tamper-resistant devices in general might be next to impossible to be realized [AK96, AK97]. Approaches requiring virtual banks need a fixed communication infrastructure to implement the incentive schemes which is not applicable for a pure ad hoc network. Zhong et al. [ZCY03] propose a Simple, Cheat-Proof, Credit based ($Sprite$) mechanism which does not require tamper-proof hardware but requires an infrastructure (Credit Clearance System) to implement credits.

Most of the approaches to alleviate blacknhole/selfish nodes do not mitigate collaborative/multiple attacks. The existing solutions to handle multiple/collaborative attacks are either based on Intrusion Detection System (IDS) [BdOZI09, KNK+07, HL04, HFLY03, RFdAG08] or are recursive application [RFN05] of the approach proposed for a single attack. These approaches handle only a single type of attack. To the best of our knowledge, no algorithm handles more than one type of attack in a single scheme. Handling more than one type of attack in a single scheme is a major challenge for researchers. Bhargava [BdOZI09] in their work have suggested a scheme to classify the attacks on the basis of observed behavior and then take corrective measures accordingly.

5.5 $RDVB$: Reliable Distance Vector routing protocol to handle Blackhole attack

In this section we present an algorithm that assuages only blackhole node. Reliable distance vector routing protocol to handle blackhole ($RDVB$) is based on $AODV$ routing protocol. After a path has been discovered in $AODV$, instead of immediately sending out data packets, we check the reliability of the path by sending a verification packet on the discovered path, the reply to which can be generated only by the destination. If there is a blackhole on the discovered path the verification packet will not reach the destination as the blackhole node does not have a path to the destination. When the source node does not receive a reply within a fixed amount of time, it discards the route. Path discovery in $RDVB$ can be thought of as consisting of two phases. Phase I is a slight modification of path-discovery in $AODV$. In phase-II, we use two control packets called Reliable Route Discovery Unit ($RRDU$) and $RRDU$ reply ($RRDU\_REP$) to check the reliability of path.

5.5.1 Phase-I of Algorithm

When a node wishes to communicate with another node it looks for a route from its table. If a valid entry is found for the destination it uses that path to send data packets else it broadcasts a $RouteRequest$ packet ($RREQ$) to its neighbors with hopcount set to 1. Neighbors check their routing tables for a fresh entry to the destination. If it is found, it replies with a $RouteReply$ ($RREP$) packet else forwards $RREQ$ to its neighbors with hopcount incremented by 1. The process continues until either the destination or an intermediate node with a fresh route to the destination is located. At each intermediate node a reverse path is created for the source. When the $RREQ$ packet reaches the destination it also replies with an $RREP$ packet. Processing of $RREQ$ at an intermediate node and destination are explained in Figures 5.16 and 5.17 respectively.

Since intermediate nodes as well as destination send $RREPs$ in response to $RREQ$ packets, source node as well as intermediate nodes may receive multiple $RREPs$ in the process. In $AODV$ source or intermediate node receiving multiple $RREPs$ selects the one that arrives first and others are discarded. Hence, one unique path is established between the source and the destination. However, in $RDVB$, a node receiving multiple $RREPs$ maintains a list of next hops in its routing table. When an intermediate node receives an $RREP$ it appends the nexthop (NH) node to the next hop list (NHL). NHL is used to discover a new path free from a malicious node in phase-II. In $AODV$, when the source node receives $RREP$ packet, route is established whereas $RDVB$ enters phase-II. Processing of $RREP$ at an intermediate node and at the source are explained in Figures 5.18 and 5.19 respectively. Algorithm 1 summarizes phase-I of algorithm.

The format of Routing Table entry in $RDVB$ is almost the same as that of $AODV$ except for the NH entry . Next Hop entry in the table is now a list of next hops. Formats of $AODV$ routing table entry, $RDVB$ routing table entry, $RREQ$ and $RREP$ are shown below:

$AODV$ **Routing Table Entry Format**							
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	Hopcount	Next Hop	Precursors	Life Time

$RDVB$ **Routing Table Entry Format**							
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	Hopcount	Next Hop List	Precursors	Life Time

$RREQ$ **Message Format**						
Type	RREQ id	Dest id	Dest Seq Number	Src id	Src Seq Number	Hopcount

$RREP$ **Message Format**					
Type	Dest id	Dest Seq Number	Src id	Hopcount	Life Time

1em 2ex

```
1.1 Source broadcasts an $RREQ$ packet.
1.2 When an intermediate node ($IN$) receives the $RREQ$ packet, it checks its routing
    table:
    if ( the node has a fresh route to destination) then
        it replies with an $RREP$ packet.
    else
        it broadcasts $RREQ$ further to its neighbors.
    end
1.3 When the destination receives an $RREQ$, it also replies with an $RREP$ packet.
1.4 When an intermediate node receives the $RREP$, it appends the next hop to its $NHL$
    and forwards the $RREP$ packet on the reverse route.
1.5 When the source node receives the $RREP$ packet it enters phase -II.
```

Algorithm 1: Phase-I of $RDVB$

5.5.2 Phase-II of Algorithm

$AODV$ has been extended to $RDVB$ by adding two types of control packets: Reliable Route Discovery Unit ($RRDU$) and $RRDU$ reply ($RRDU\_REP$). $RRDU$ messages are control packets sent by the source node and $RRDU\_REP$ message is the response of $RRDU$ by the destination to the source node. $RRDU\_REP$ can only be generated by the destination. We assume that there is no impersonation i.e. no node other than the destination can generate $RRDU\_REP$ on behalf of the destination. In phase-II when the source node receives an $RREP$, it sends an $RRDU$ packet with hopcount set to 1 on the path to check its reliability. If the source node receives multiple $RREPs$, it sends out an $RRDU$ packet to each of the node from which it receives the $RREP$ packet. The path from which it receives the reply to $RRDU$ is finally established as a reliable path.

When an intermediate node receives an $RRDU$ packet, it forwards $RRDU$ to all the nodes in its NHL with hopcount incremented by one. It also keeps a copy of $RRDU$ for the future $RREPs$. It keeps on sending $RRDUs$ to the nodes from which it receives $RREPs$ until it receives an $RRDU\_REP$ packet. For example, in Figure 5.1, suppose A receives the first $RREP$ from BH and forwards to s. After this, it receives $RREP$

from B_1 and adds it to NHL. When it receives $RRDU$ from s, it sends $RRDU$ to nodes in NHL i.e to BH and B_1. It also keeps a copy of $RRDU$ packet. Later when it receives $RREP$ from C_1, it adds this to NHL and forwards the copy of $RRDU$ to it. Destination may also receive multiple $RRDUs$; it responds with an $RRDU\_REP$ packet (with hopcount set to 1) to the one that arrives first and discards future $RRDU$ packets as duplicates. Thus each node including the source node receives a unique $RRDU\_REP$ and a secure path is established. Each intermediate node keeps only the node from which it receives the $RRDU\_REP$ in the NHL and discards all other entries from the list. It copies the hopcount from the $RRDU\_REP$ packet in the routing table entry for the destination, increments the hopcount in the packet by 1 and forwards the packet on the reverse route. Algorithm 2 summarizes phase-II of $RDVB$. 1em 2ex

```
2.1 When the source node receives an $RREP$ packet it sends out an $RRDU$ packet with
    hopcount set to 1 to check the reliability of the path.
2.2 When an intermediate node receives the $RRDU$ packet it increments the hopcount in
    the packet by one, forwards it to all the nodes in its $NHL$ and keep a copy for future
    $RREPs$.
     /* Notice here that if we did not keep this list and
    forwarded the $RRDU$ packet only to the node from which it
    received the first $RREP$ and that path had a blackhole
    node, we had no way to discover a path free from the
    malicious node.  We would have known that the path
    discovered was under attack but would not have been able to
    discover an alternate reliable path.  */
2.3 When the destination receives the $RRDU$ packet it replies with an $RRDU\_REP$
    packet, hopcount set 1, to the first $RRDU$ it receives. It discards the $RRDU$ packets it
    receives in future as duplicates.
2.4 $RRDU\_REP$ travels a reliable path back to the source node and the path is established.
```

Algorithm 2: Phase-II of $RDVB$

Processing of $RRDU$ and $RRDU\_REP$ are explained in Figures 5.20, 5.21 and 5.22. Formats of $RRDU$ and $RRDU\_REP$ messages are shown below:

$RRDU$ **Message Format**							
Type	RRDU id	Dest id	Dest Seq Number	Src id	Src Seq Number	Hopcount	Life Time

$RRDU\_REP$ **Message Format**					
Type	Dest id	Dest Seq Number	Src id	Hopcount	Life Time

As in $AODV$, $RDVB$ uses $RERR$ and $HELLO$ messages for route maintenance.

5.5.3 Security Analysis: handling blackhole attack

In this section we will show that our scheme discovers a path free from blackhole node. See Figure 5.1. If BH is a malicious node then it may send $RREP$ without having a route to the destination declaring that it has a fresh route to the destination. In case of $AODV$, if A receives the first $RREP$ from BH, it keeps the path through BH and discards others if the hopcount of others is more. Hence a path through BH is set up between the source and the destination. In $RDVB$, we send $RRDU$ on this path to check the reliability of the path. Suppose A receives $RREPs$ first from BH, then subsequently from B_1 (an intermediate node with path), and then from C_1 ($RREP$ received through path $C_1 - C_2 - C_3 - C_4 - t$). It forwards the first $RREP$ to the source. Later when A receives $RREPs$ from B_1 and C_1, it stores their *id*s in NHL and discards the $RREP$ packets. When the source receives the $RREP$ packet, it sends $RRDU$ to A. A forwards $RRDU$ to BH, B_1 and C_1. However, since no node other than the destination can generate a reply to $RRDU$, A does not receive $RRDU\_REP$ from BH. Suppose t receives $RRDU$ first from C_4 as it is on a shorter route and then from B_5. It sends $RRDU\_REP$ to C_4 and discards the $RRDU$ from B_5. Thus A receives $RRDU\_REP$ from C_1 via $C_1 - C_2 - C_3 - C_4 - t$, it sets C_1 as next hop on the path to t and forwards $RRDU\_REP$ to s. When s receives $RRDU\_REP$ a secure reliable path, free from blackhole, is established.

5.6 $RDVBS$: Reliable Distance Vector routing protocol to handle attacks due to Blackhole and Selfish nodes

Consider a selfish node of type 1 i.e. a selfish node that forwards neither the control packets nor the data packets. Such a node will be isolated in $RDVB$ as it will not forward $RRDU$ packet and hence $RRDU\_REP$ will not be received through it. However, if there is a selfish node of type 2 i.e. a node forwards all control packets including $RRDU$ and $RRDU\_REP$ but does not cooperate in forwarding data packets, $RDVB$ will not be able to avoid it and hence the above algorithm in its current form will not be able to isolate such a node. Also, a node on the discovered path may co-operate in forwarding data packets for some time and may become selfish after some time due to its reduced energy levels. In order to identify such a behavior we modify $RDVB$ and call it as Reliable Distance Vector routing algorithm for Handling Blackhole and Selfish nodes ($RDVBS$). Path discovery in $RDVBS$ is same as that in $RDVB$. However once a path free from black node has been discovered, $RRDUs$ are sent periodically to maintain the reliability of the path, i.e. to detect if any misbehaving selfish node has crept into the path. We call this as phase-III of the algorithm.

5.6.1 Phase-III of Algorithm

To maintain the reliability of the path we introduce a field called Forward Data Packet Count ($FDPC$) in the routing table (RT) entry as well as in the $RRDU\_REP$ packet and a field called Reliability Flag (RF) in the $RRDU\_REP$ packet. Initially RF is set to 1 by the destination and it is cleared when a selfish node is detected on the path. $FDPC$ in the routing table entry keeps a count of the number of data packets forwarded by the node. For every data packet received and forwarded by a node, $FDPC$ in RT entry of the node is incremented. This $FDPC$ is copied by the node, on return, in the $RRDU\_REP$ packet to tell its previous neighbor as to how many data packets it has forwarded. The neighbor uses this count to detect whether the node has forwarded all the packets or not. If a node discovers that its next hop neighbor has not forwarded all the packets, it informs the sender by clearing the reliability flag in the $RRDU\_REP$ packet. Since the selfish node of type 2 participate in forwarding all control packets exept the ones used for discovery of path ($RREQs$ and $RREPs$) it forwards the $RRDU\_REP$ packet and since it does not intend to disrupt the normal functioning of the system, it does not lie. In case a selfish node is detected on the discovered path, a fresh route discovery is initiated by the source. Since we assume that once a node starts dropping the data packets, it does not participate in route establishment until its energy is restored, the selfish node is isolated when fresh route discovery is initiated.

A node keeps an entry for each destination in its routing table. In $RDVB$, the routing table entry for destination t does not depend upon which source is trying to communicate with t. When two source nodes say s_1 and s_2 try to communicate with t, the only thing an intermediate node remembers is the next hop required to reach t. But now the node needs to remember how many data packets it has forwarded for each communication. Thus, a list called Reliability List (RL) is added in the routing table entry of each node. An entry in the RL has source address, Forwarded Data Packet Count $(FDPC)$ and $RRDU\ id$, i.e. the triplet $(Sourceaddress, FDPC, RRDU\ id)$. The triplet entry keeps a count of the number of data packets forwarded by the node from source s to the destination t since the last $RRDU$. $RRDU\ id$ is incremented every time a new $RRDU$ packet is sent by the source. The triplet is initialized when the first $RRDU$ is processed in phase-II and, it is used and updated when periodic $RRDU$s are processed in phase-III. Assuming that not too many nodes will be communicating with a given node at a given time, the size of RL is not expected to be big. Algorithm 3 summarizes phase-III of the algorithm.

Processing of $RRDU$ and $RRDU\_REP$ in phase-II and phase-III of $RDVBS$ is shown in Figures 5.23, 5.24, 5.25 and 5.26. The format of $RDVBS$ routing table entry is same as that of $RDVB$ except for the additional RL field and the format of $RRDU\_REP$ is modified to include the reliability flag field. The format of $RDVBS$ routing table entry and modified $RRDU\_REP$ are shown below:

$RDVBS$ **Routing Table Entry Format**								
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	Hopcount	Next Hop List	Precursors	RL	Life Time

$RRDU\_REP$ **Message Format modified for** $RDVBS$							
Type	Dest id	Dest Seq Number	Src id	Hopcount	FDPC	Reliability Flag	Life Time

1em 2ex

```
Initialization :$RF \leftarrow 1$, $FDPC \leftarrow 0$
3.1 $RRDUs$ are sent periodically to maintain the reliability of the path.
3.2 When an intermediate node receives a periodic $RRDU\_REP$ from its next hop
    neighbor it checks its routing table:
      if ( next hop neighbor has not forwarded all the packets) then
          it clears the reliability flag in $RRDU\_REP$ packet and forwards the packet on
          reverse route.
      else
          it copies $FDPC$ from the $RL$ entry of routing table in the $RRDU\_REP$ packet
          and forwards the packet on reverse route.
      end
3.3 When source receives $RRDU\_REP$ it checks $RF$ in the $RRDU\_REP$ packet:
      if (RF is set to $1$) then
          path is considered to be reliable; it sends more data packets, if any.
      else
          it initiates route discovery process again.
      end
```

Algorithm 3: Phase-III of $RDVBS$

Note: We can improve the performance of the algorithm slightly by checking if the $RREP$ packet is from the destination itself. If so, it need not send the $RRDU$ packet and it can start sending the data packets on this path without waiting for $RRDU\_REP$ from the destination.

5.6.2 Security Analysis: handling selfish nodes

Suppose that the path discovered in Figure 5.1, is $s - A - C_1 - C_2 - C_3 - C_4 - t$ and let that a node (say C_2) on this path becomes selfish. Let C_2 be a selfish node of type 2 i.e. it forwards control packets but does not forward the data packets. Suppose s sends n data packets to t before sending next $RRDU$. Then A and C_1 forward all the n data packets to the successor. Let C_2 forwards only p out of the n packets. Then C_3 and C_4 also forward

p packets and, the destination receives only p packets. After some time, s sends another $RRDU$ and t sends back the count of the received packets in $RRDU\_REP$. $FDPC$ field in the $RRDU\_REP$ is set to p by the destination. At every node x on the reverse path from the destination to the source, $FDPC$ in $RRDU\_REP$ is set to the number of data packets forwarded by x on the forward path. Hence, C_4, C_3, and C_2 set $FDPC$ in $RRDU\_REP$ to p whereas C_1 and A set it to n. When C_1 sees that it had forwarded n packets to C_2 but C_2 forwarded only p $(< n)$ out of them it comes to know that C_2 is selfish and it clears the Reliability Flag in the $RRDU\_REP$ packet to 0. When the source receives this $RRDU\_REP$ packet with RF set to zero it knows that something is wrong on this path and it initiates a fresh route discovery. Here we make an assumption that once a node stops forwarding data packets, it does not involve itself into route establishment also. Thus, C_2 discards any $RREQ$ packet received from C_1 and a path ignoring C_2 is established. The previous set of data packets are sent again.

5.7 Simulation Study

We simulated our protocol using Network Simulator [NS2]. To study the performance of $RDVBS$, packet delivery ratio, average end-to end delay and routing overhead were studied.

5.7.1 Simulation Design

Simulation results were obtained for 50 nodes located over $1000m$ by $1000m$ region. The traffic sources are CBR (constant bit rate), 512-byte as data packet, sending rate is 1 pkt/sec and with maximum load of 300 packets for one transaction. The node movement speed is varied from 0 to 80 which will be closer to real applications. The mobility are done with pause time 100 second. Script was executed for 300 seconds.

5.7.2 Simulation Results

We compared the performance of our protocol with that of $AODV$ and $DENG$ in presence of blackhole attack.

Comparison with DENG: Both $DENG$ and $RDVBS$ are able to detect and isolate blackhole node. Average End to End Delay ($AEED$) and Packet Delivery Ratio (PDR) of $RDVBS$ and $DENG$ are comparable as shown in Figure 5.3 and 5.4. Our protocol out performs $DENG$ in terms of Routing Overhead (RO) (see Figure 5.5). $DENG$'s route discovery phase comprises of broadcasting route request twice, once for destination and another for intermediate node (for feedback). This leads to the generation of a much more number of $RREQ$ packets than the number of control packets generated in $RDVBS$.

Comparison with AODV: As expected, Packet Delivery Ratio (PDR) of $AODV$ drops significantly as compared (Figures 5.4) to $RDVBS$ and $DENG$ in presence of a blackhole node. In the absence of mobility, PDR of $AODV$ is zero whereas that of $RDVBS$ is 1. As the nodes start moving sometimes the blacknode falls on the path and sometimes not. Since $RDVBS$ isolates the blackhole node and $AODV$ does not, PDR of $RDVBS$ remains better than that of $AODV$ whereas the RO of $RDVBS$ is only slightly more than that of $AODV$ which is natural to expect.

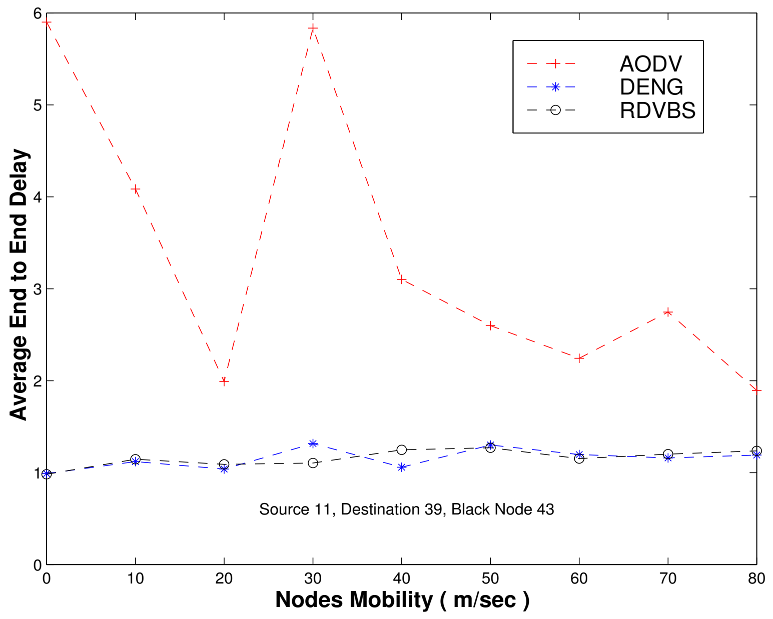

Figure 5.3: Comparison of Average End to End Delay in presence of blackhole node

We also compared the performance of our protocol with that of $AODV$ in the presence of selfish nodes.

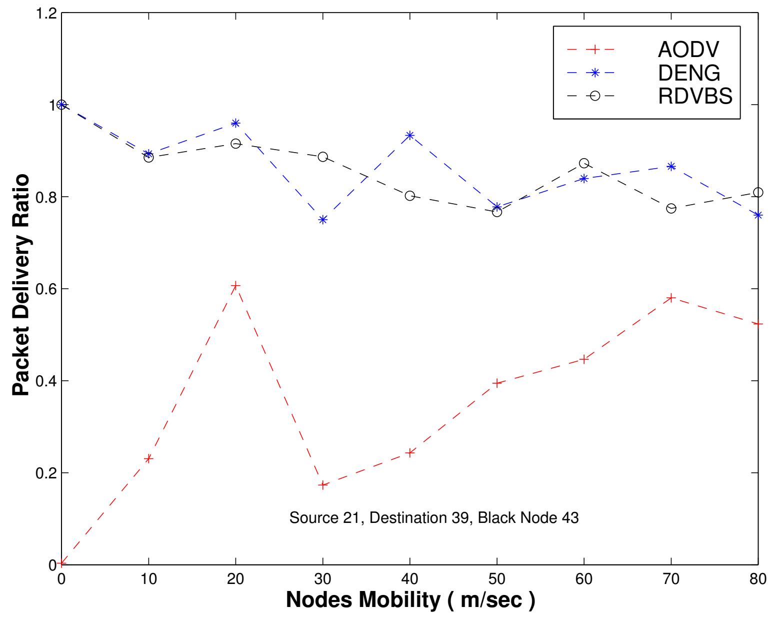

Figure 5.4: Comparison of Packet Delivery Ratio in presence of blackhole node

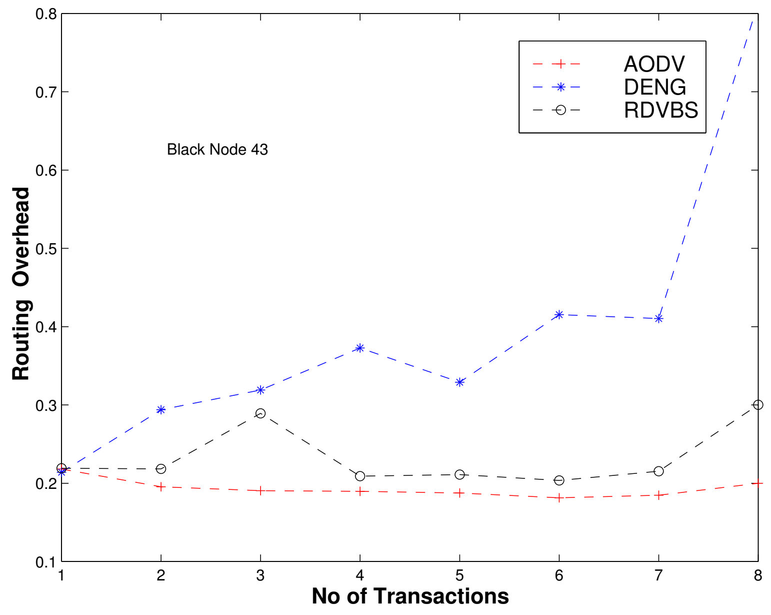

Figure 5.5: Comparison of Routing Overhead in presence of blackhole node

Comparison with AODV in presence of selfish nodes: As shown in Figure 5.2, PDR of $AODV$ decrements by 10% - 20% with every 10% increase in the percentage of selfish nodes in network. The figure also shows that when 50% of the nodes of the network are selfish PDR degrades by more than 55%. In $RDVBS$, as shown in Figure 5.6, PDR

degrades just by $1\% - 3\%$ every time the percentage of selfish nodes increases by 10%. On the other hand, $AEED$ and RO (Figures 5.7 and 5.8) do not increase much with increase in the number of selfish nodes. Figure 5.9 shows the impact of mobility on PDR in the presence of selfish nodes. For $RDVBS$, it is observed that this effect diminishes significantly.

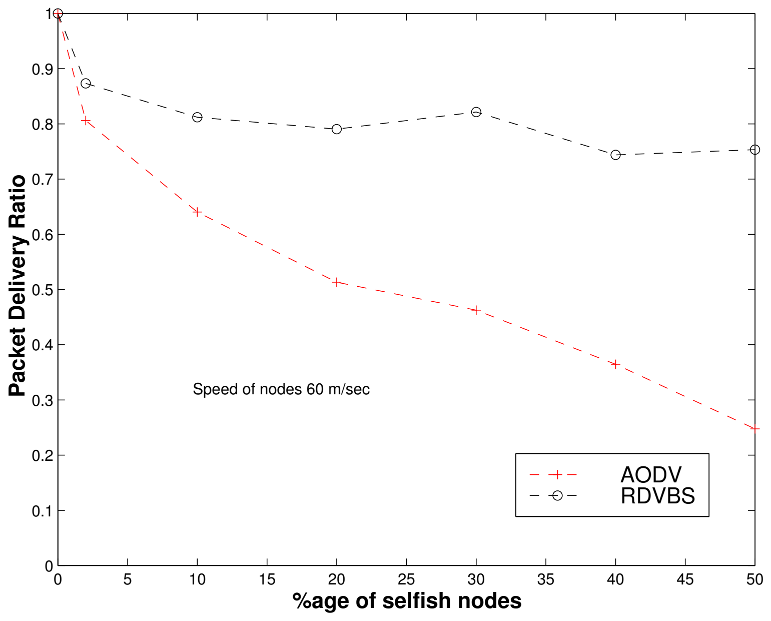

Figure 5.6: Comparison of Packet Delivery Ratio in presence of selfish nodes

Comparison with AODV and DENG in the absence of attack: We also compared our protocol with $AODV$ and $DENG$ in the absence of black node and selfish nodes. When there are no black nodes, verification step of the route discovery phase of $RDVBS$ and $DENG$ lead to some routing overhead (Figure 5.10). Here also $RDVBS$ outperforms $DENG$ whereas RO of $RDVBS$ is only slightly more than that of $AODV$. Packet delivery ratio and average end-to-end delay of all the three protocols are comparable, see Figure 5.11 and 5.12.

5.8 Handling Multiple and Co-operative Blackhole nodes

Our protocol also handles multiple black hole attacks. Consider the scenario in Figure 5.13. If several $RREPs$ are received from all the black nodes present in the network,

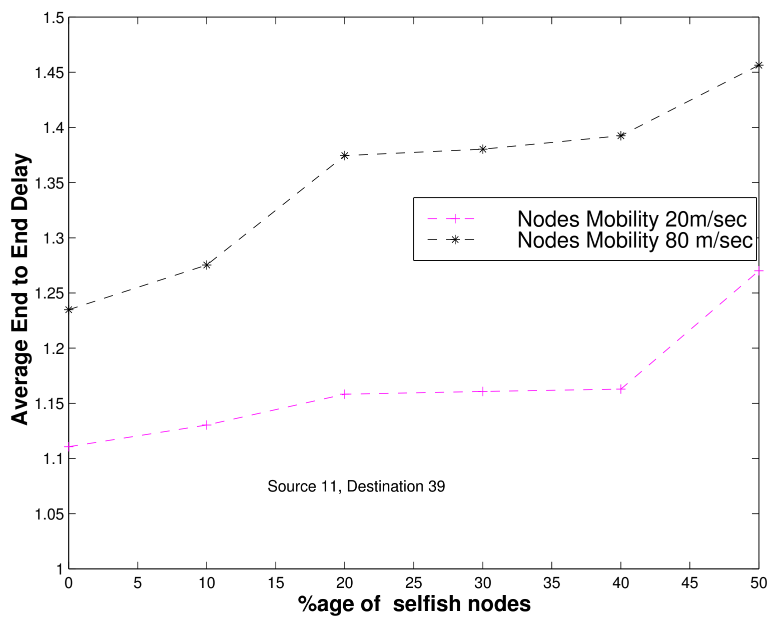

Figure 5.7: Average End to End Delay of $RDVBS$ in presence of selfish nodes

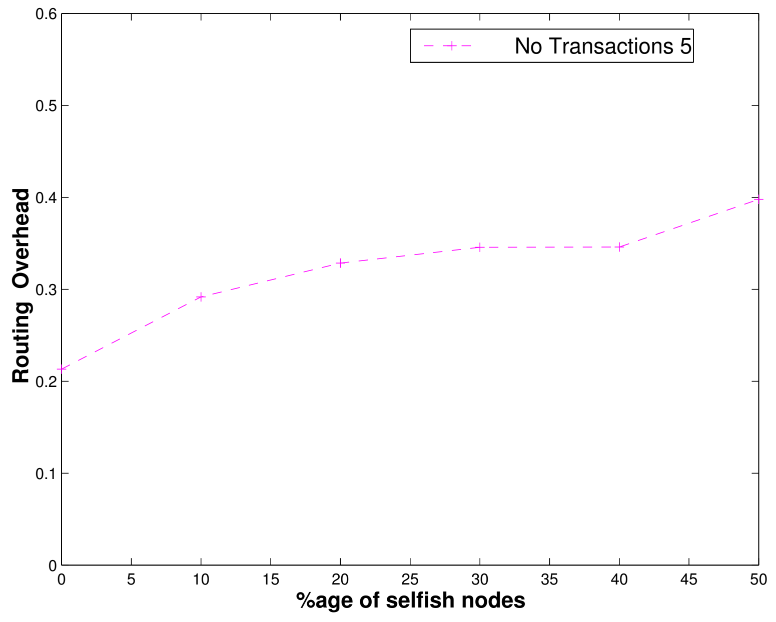

Figure 5.8: Routing Overhead of $RDVBS$ in presence of selfish nodes

our protocol sends an $RRDU$ packet to all of them. However, no $RRDU\_REP$ will be received from any black node and hence all the black nodes will be isolated. Also consider the scenario of Figure 5.14 where the black nodes co-operate with each other to launch the attack. Such attacks are also detected and isolated by our protocol in a similar way as

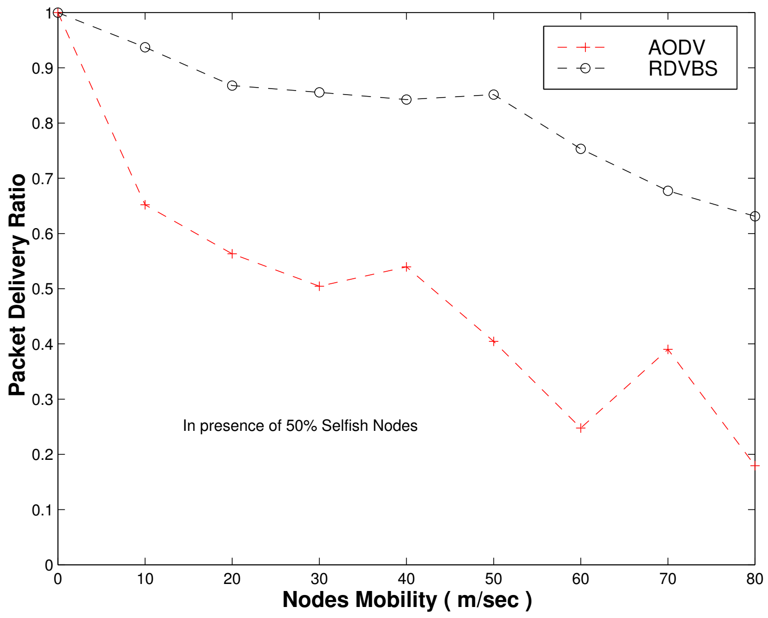

Figure 5.9: Comparison of Impact of mobility on Packet Delivery Ratio in presence of selfish nodes

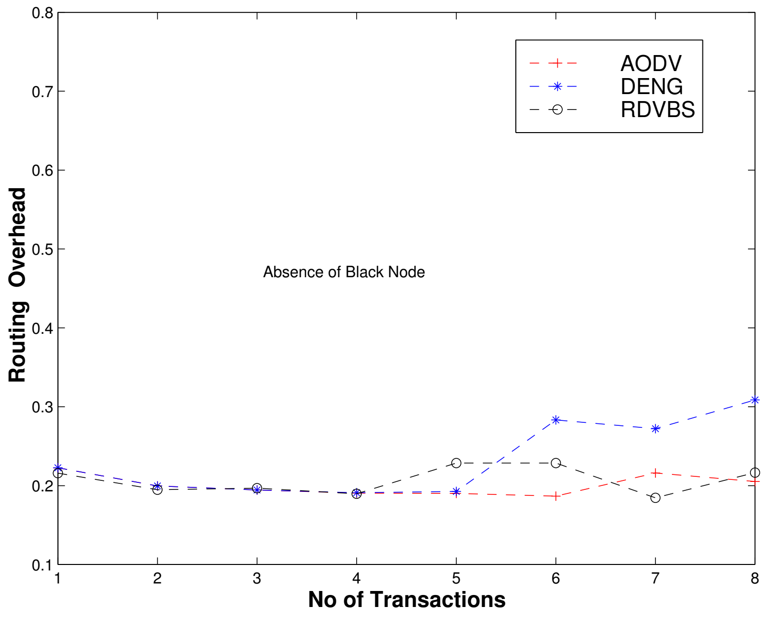

Figure 5.10: Comparison of Routing Overhead in absence of blackhole node

no $RRDU\_REP$ will be received from such a path. $DENG$ will be able to detect multiple attacks of Figure 5.13 by recursive application thereby incurring a lot of overhead. Also, it will not be able to detect the co-operative attack of Figure 5.14. In [TS07], as the number of co-operating black nodes sending $RREPs$ with the same NHN increases, the

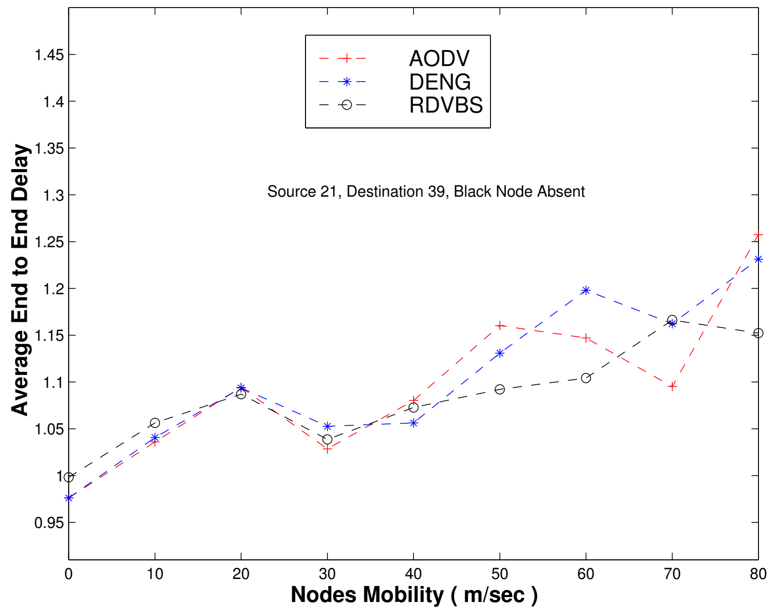

Figure 5.11: Comparison of Average End to End Delay in absence of blackhole node

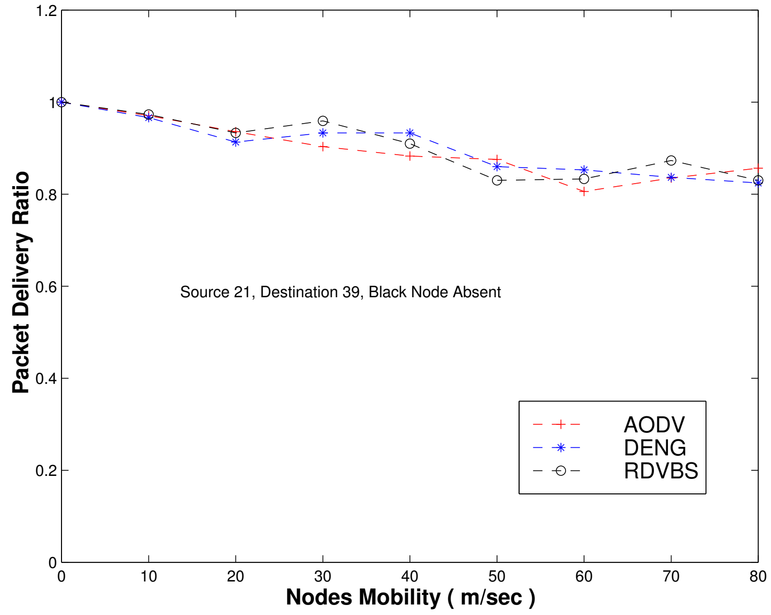

Figure 5.12: Comparison of Packet Delivery Ratio in absence of blackhole node

chances of establishing a path through a black node increases.

Comparison with $AODV$ in presence of multiple black nodes: Figure 5.15 shows that the decrease in PDR with the increase in the number of black nodes is much less in case of $RDVBS$ as compared to that for $AODV$.

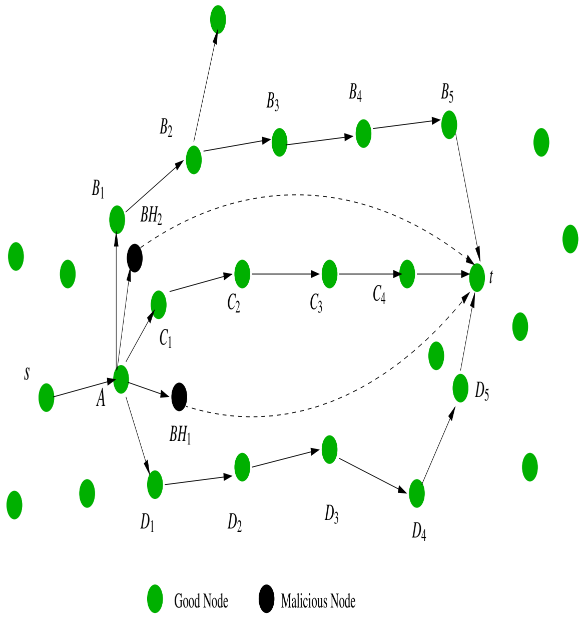

Figure 5.13: Example of multiple blackhole nodes

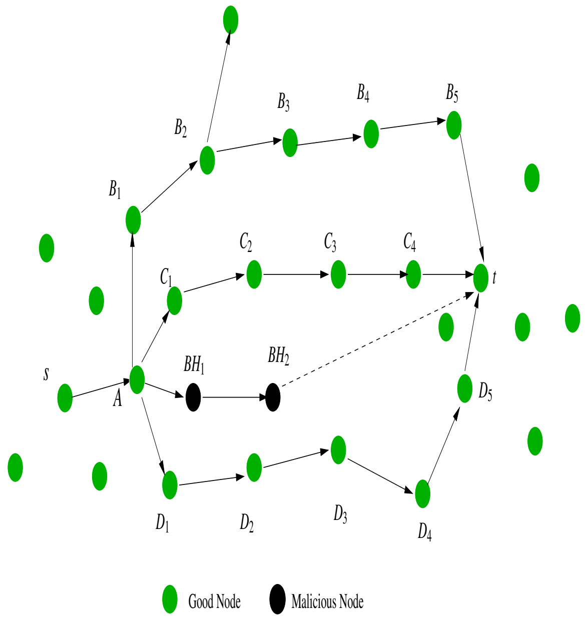

Figure 5.14: Example of co-operative blackhole nodes

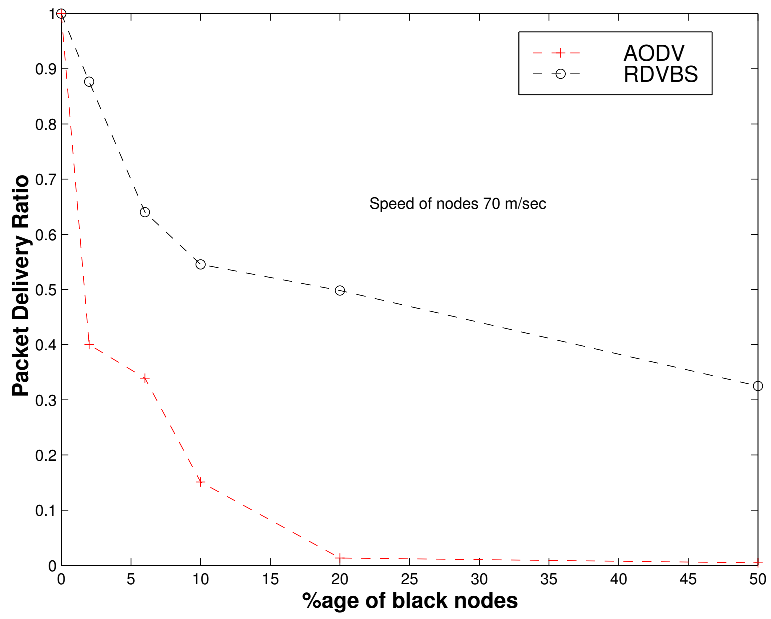

Figure 5.15: Impact of multiple blackhole nodes on Packet Delivery Ratio of $RDVBS$ and $AODV$

When an intermediate node receives an $RREQ$ packet from source s, it does the following steps:

1. if it has a fresh route to the destination, it replies to the source with $RREP$ else it broadcasts (forwards) the $RREQ$ packet to its neighbors with hopcount incremented by 1. If additional copies of the same $RREQ$ are later received, they are discarded as duplicates.

2. it sets up a reverse path for the reply message.

   (a) if it has an entry in its routing table for s but it is not fresh, it refreshes it.

   (b) if there is no entry for s in its routing table it creates an entry for s by copying the hopcount and the source *id* from the $RREQ$ packet and, setting the NH field to the address of the neighbor from which the first copy of the broadcast packet is received.

Figure 5.16: Processing of $RREQ$ at an intermediate node in $RDVB$

When the destination receives an $RREQ$ packet from source s, it does the following steps:

1. if it has an entry in its routing table for s but it is not fresh, it refreshes it.
2. if there is no entry for s in its routing table it creates an entry for s by copying the hopcount and the source *id* from the $RREQ$ packet and, setting the NH field to the address of the neighbor from which the first copy of the broadcast packet is received. It creates an $RREP$ packet and unicasts $RREP$ to the next hop on the reverse path.
3. if additional copies of the same $RREQ$ are later received, they are discarded as duplicates.

Figure 5.17: Processing of $RREQ$ at the destination in $RDVB$

When an intermediate node receives an $RREP$ message, it does the following steps:

1. if it has an entry in its routing table for the destination but it is not fresh, it refreshes it.
2. if it does not have an entry for the destination, it creates an entry for it and sets the NH field to the address of the neighbor from which the packet is received. It forwards it to the next hop on the reverse path.
3. if it already has an entry for the destination (in case of multiple $RREPs$) in its routing table, it appends the next hop from which it received the $RREP$ in the NHL entry of the routing table and discards the $RREP$ packet. This is required to establish a secure path from blacknode in phase-II of the algorithm.

Figure 5.18: Processing of $RREP$ at an intermediate node in $RDVB$

When the source node receives an $RREP$ packet, it does the following:

1. if it has an entry in its routing table for the destination but it is not fresh, it refreshes it.
2. if there is no entry for the destination in its routing table it creates an entry for it and sets the NH field to the address of the neighbor from which the packet is received, as the next hop.
3. if it already has a fresh entry for the destination in its routing table (in case of multiple $RREPs$), it appends the next hop from which it received $RREP$ in NHL entry of the routing table.
4. the node sends an $RRDU$ packet with hopcount set to 1 to the node from which it received the $RREP$ packet and phase-II of the algorithm starts.

Figure 5.19: Processing of $RREP$ at the source in $RDVB$

When an intermediate node receives an $RRDU$ packet, it does the following:

1. if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as it is done on seeing $RREQ$ (this case may arise when an intermediate node n_1 replies to $RREQ$ with an $RREP$ packet and n_2 is a node on the path from n_1 to t).
2. each node on the path of $RRDU$ must be having a table entry for the destination. It increments the hopcount in the $RRDU$ packet by 1 and forwards it to all the nodes in NHL.
3. it keeps a copy of $RRDU$ packet for subsequent $RREPs$.

Figure 5.20: Processing of $RRDU$ at an intermediate node in $RDVB$

When the destination receives the $RRDU$ packet, it does the following steps:

1. if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as it is done on seeing $RREQ$.
2. it creates an $RRDU\_REP$ packet with hopcount set to 1 and replies to the $RRDU$ which arrives first. It discards the copies of $RRDU$ it receives in future, as duplicates.

Figure 5.21: Processing of $RRDU$ at the destination in $RDVB$

When an intermediate node receives an $RRDU\_REP$ packet, it does the following:

1. the node must be having a table entry for the source. It finds next hop on the path from the table entry, and forwards $RRDU\_REP$ to it with hopcount incremented by 1.
2. in a table entry for the destination, it keeps only one entry in the NHL, the one from which it received the $RRDU\_REP$ and deletes others. It copies hop count from the packet in the routing table entry for the destination.

Since no intermediate node can generate $RRDU\_REP$, source node receives a unique $RRDU\_REP$ and a secure path is established. Source starts sending data packets on this path.

Figure 5.22: Processing of $RRDU\_REP$ in $RDVB$

Phase-II of $RDVBS$

When an intermediate node receives the fist $RRDU$ packet it does the following:

1. if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as is done on seeing $RREQ$.
2. for the first $RRDU$ packet it receives from source s, it appends an entry $(s, 0, 0)$ in RL (i.e. id of the source is copied from the originator field of the $RRDU$ packet and, $FDPC$ and $RRDU\ id$ are set to zero). Each node on the path of $RRDU$ must be having a table entry for the destination. It forwards $RRDU$ with hopcount incremented by 1 to all the nodes in NHL.
3. it keeps a copy of $RRDU$ packet for subsequent $RREPs$ as in $RDVB$.

Phase-III of $RDVBS$

If an intermediate node receives a periodic $RRDU$ packet it does the following:

1. the node must be having a table entry for the destination. It updates the (source, $FDPC$, $RRDU\ id$) triplet i.e. the $RRDU\ id$ is copied from the $RRDU$ packet and the $FDPC$ count is reset to 0. It finds the next hop on the path from the table entry and forwards $RRDU$ to it (at this time NHL contains a unique NH as a unique path had already been established).

Figure 5.23: Processing of $RRDU$ at an intermediate node in $RDVBS$

Phase-II of $RDVBS$

When the destination receives the first $RRDU$ packet it does the following:

1. if there is no reverse path entry for s, it creates an entry for s in its routing table in the same manner as it does on seeing $RREQ$.
2. for the first $RRDU$ packet it receives from source s, it appends an entry $(s, 0, 0)$ in RL (i.e. id of source is copied from the originator field of the $RRDU$ packet and $FDPC$, $RRDU\ id$ are set to zero) and it discards the $RRDU$ packet.
3. it creates an $RRDU\_REP$ packet with hopcount set to 1 and replies to the node from which the first $RRDU$ is received.

Phase-III of $RDVBS$

When the destination receives a periodic $RRDU$ packet it does the following:

1. it creates an $RRDU\_REP$ packet and copies $FDPC$ from the RL list entry to $RRDU\_REP$. It finds the next hop for s on the path from table entry and sends $RRDU\_REP$ packet to it.

Figure 5.24: Processing of $RRDU$ at the destination in $RDVBS$

Phase-II of $RDVBS$

When an intermediate node receives the fist $RRDU\_REP$ packet, it finds the next hop for the source from the table entry (on the reverse path) and forwards $RRDU\_REP$ with hopcount incremented by 1 to it.

Phase-III of $RDVBS$

When an intermediate node receives a periodic $RRDU\_REP$ packet, it compares $FDPC$ stored in the routing table entry with $FDPC$ in $RRDU\_REP$ packet; if they are same, it forwards $RRDU\_REP$ to the next hop on the reverse path else it copies the $FDPC$ value stored in its routing table entry in the $RRDU\_REP$ packet, clears the reliability flag in the $RRDU\_REP$ packet and forwards the $RRDU\_REP$ packet to the next hop on the reverse path.

Figure 5.25: Processing of $RRDU\_REP$ at an intermediate node in $RDVBS$

Phase-II of $RDVBS$

When the source node receives the fist $RRDU\_REP$ packet, a secure path is discovered and the path is established; it starts sending the data packets on the path.

Phase-III of $RDVBS$

When the source node receives a periodic $RRDU\_REP$ packet, it checks the RF flag in the packet. If it is set, path is considered to be reliable and it continues sending data packet on that path else it realizes that there is a selfish node on the path and it initiates the route discovery process again.

Figure 5.26: Processing of $RRDU\_REP$ at the source in $RDVBS$

Chapter 6

EEW: End-to-End scheme against Wormhole attacks

6.1 Introduction

In this chapter, we address the problem of detecting wormhole attacks in ad hoc networks. Since the mobile devices use a wireless medium to transmit information, the malicious nodes can eavesdrop the packets, tunnel them to another location in the network and re-transmit them at the other end. Attackers may use out of band channel, high power transmission, packet relay or encapsulation technique to tunnel packets to colluding nodes. The tunnel so created forms a wormhole. The tunneling procedure generates an illusion that the two nodes more than one hop away are in the neighborhood of each other. We call the two nodes as the victim nodes. Since most of the route discovery mechanisms maintain a neighborhood set at each node, false information about a node's neighbor can severely affect the discovered route. If a routing protocol uses the number of hopcounts to compute the shortest path, it prevents the routes longer than three hops to be discovered between the victim nodes. If the routing protocol uses the round trip delay to compute the shortest path and there exists a fast transmission path (out of band channel) between the two ends of the wormhole, it prevents normal multi-hop routes to be discovered since the tunneled packets travel much faster through the wormhole than through the normal route. Hence in either case the route is established through the wormhole. Having high speed or long range transmission can be beneficial for the system throughput if used in a friendly way. However, after gaining full control over the traffic, the malicious node can drop or compromise packets reducing the network throughput, instead of improving. Since the path through the wormhole appears to be the shortest, most of the traffic from one side of the wormhole to the other side is routed through the wormhole leading to network partitioning. The attacker can also store the data and perform traffic analysis which can be used to gain partial or full control over the network. The attacker can also decide to drop the packets selectively at times crucial for the secure functioning of the network. Similarly, the attacker can selectively switch off the wormhole tunnel. The communicating parties using the wormhole link will have to find new routes. The network will thus be flooded with lot of route request thereby reducing the performance of the network greatly.

Most of the existing approaches to alleviate wormhole attacks are node-to-node which require a trust level between two neighboring nodes. In contrast, end-to-end schemes require trust only between the source and the destination, and the wormhole detection is carried out either only by the source or by the destination. Wang et al. [WBLW06] proposed a first mechanism requiring only end to end trust. It requires $O(km)$ storage and $O(km^2)$ computation time where k is the number of hopcount for the path. We present an end-to-end scheme [KG08, GK08, KG10] which improves upon Wang et al.'s scheme in terms of storage and computation overhead. Our protocol requires only $O(k)$ space and time. Wang et al. also achieved a reduction of $O(m)$ in storage and computation requirement in their extended algorithm 'Cell based Open Tunnel Avoidance ($COTA$)'. However their is a trade-off between the storage requirement and the number of false positives/detection capability and also between the computation time and the number of false positives/detection capability. Other end-to-end schemes [CL06, QSL07, THL$^+$07, SB07] assuage only a specific type of wormhole which is launched by encapsulation.

Our protocol requires that every node in the network is equipped with a global positioning system (GPS) and that every node knows its location. We assume that nodes are equipped with secret keys which provide secrecy and authenticity of message between the source and the destination. The protocol does not require clock synchronization. We provide a lower bound on the minimum number of hops on a good route. Any path showing lesser hopcounts is shown to be under attack. The idea works well for closed wormholes where nodes do not lie about their positions. However, in open or half-open wormhole a malicious node may show a large hopcount, big enough to escape the test or may lie about its position. Our protocol checks a node from lying too much about its position by checking if two consecutive nodes on the path are in direct range of each other. To detect a malicious node lying about the hopcount every intermediate node attaches its *id* to the packet, recomputes the Message Authentication Code (MAC) code using a secret shared key between itself and the destination. If a malicious node lies about the hopcount, it will have to generate and attach a THL (traversed hop list) to each packet. Though the node may be able to generate a fake list of *id*s, it will not be able to generate their MAC code as it neither has their keys nor enough computational power. All the checks are performed by the destination and intermediate nodes do not verify anything.

Our scheme can be included in the route discovery process as well as used after a data path has been established to examine the path for the presence of wormhole from time to time. It can be used as a plug-in for any existing routing protocol like DSR (Dynamic Source Routing) or $AODV$ (Ad hoc On-demand Distance Vector).

6.2 Problem Statement

The wormhole attack in wireless networks was independently introduced in [SDL$^+$02], [PH02] and [HPJ03a]. In [KBS05] authors have described different types of wormholes depending upon the techniques used to tunnel the packets between the colluding nodes: wormhole using encapsulation, wormhole using out-of-band channel, wormhole with high power transmission, and wormhole using packet relay.

1. Wormhole using encapsulation: The source node broadcasts a route request packet, received by the malicious node $M1$, which encapsulates it and forwards it to $M2$ via good nodes. $M2$ demarshalls the packet and broadcasts it further to the destination. Note that due to the packet encapsulation, the hopcount does not increase during the traversal through the good nodes thus the nodes near $M2$ think that the nodes

near $M1$ are one-hop away. See Figure 6.1.

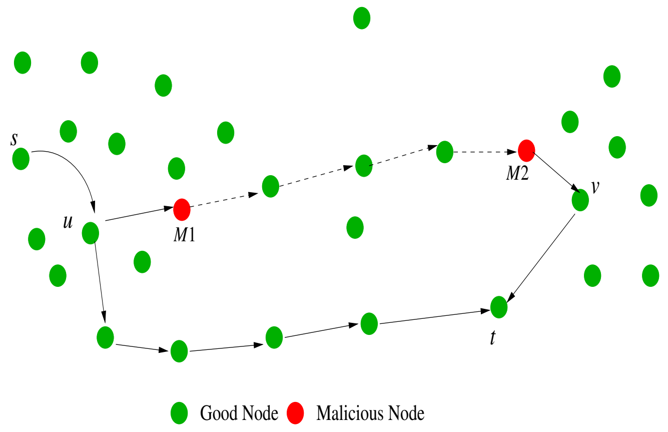

Figure 6.1: Wormhole using encapsulation

2. Wormhole using out-of-band channel: The two colluding nodes communicate directly via an out-of-band (use different radio frequency compared to the frequency bands used by the other good nodes of the network) high-bandwidth channel using a long-range directional wireless link or a direct wired link. See Figure 6.2.

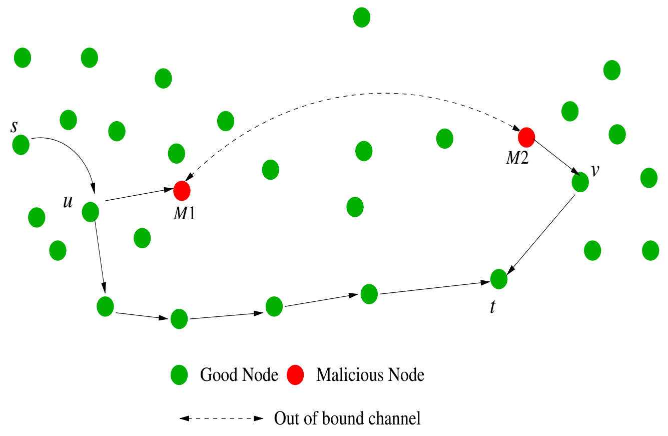

Figure 6.2: Wormhole using out-of-band channel

3. Wormhole using high power transmission: the attackers use the same radio channel used by the good nodes in the network but increase their transmission range by transmitting at the highest possible power. Hence distant nodes receive the route request packet faster from the malicious nodes than through the normal multi-hop

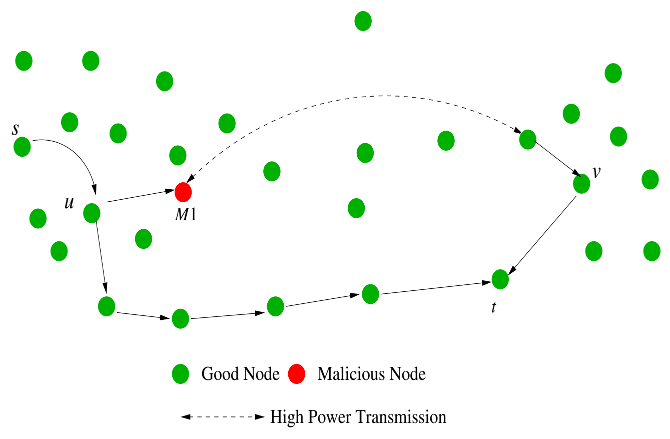

Figure 6.3: Wormhole using high power transmission

route increasing the chance of malicious nodes to get inserted in the route. See Figure 6.3.

4. Wormhole using packet relay: A malicious node relays packets between two non-neighbor nodes creating an illusion that they are neighbors.

Wang et al. have classified wormholes depending upon whether one, both or none of the two colluding nodes at the end of the tunnel are visible to the good nodes (the victim nodes). See Figures 6.4, 6.5 and 6.6. They describe the wormhole as closed if none of two ends of the wormhole tunnel is visible; u and v get the illusion that they are direct neighbors of each other. It is called half-open, if one of the malicious node say, near u is visible but the other end is not visible (to v); two hops path is established between u and v. Finally, they call the wormhole to be open if both the ends are visible to u and v; three hops path is established between them through the wormhole in this case.

Now consider a node s trying to establish a route to destination t. It is possible to establish a path between s and t through the wormhole as shown in Figure 6.7. Once a path through a wormhole is established malicious nodes may drop or compromise the data packets.

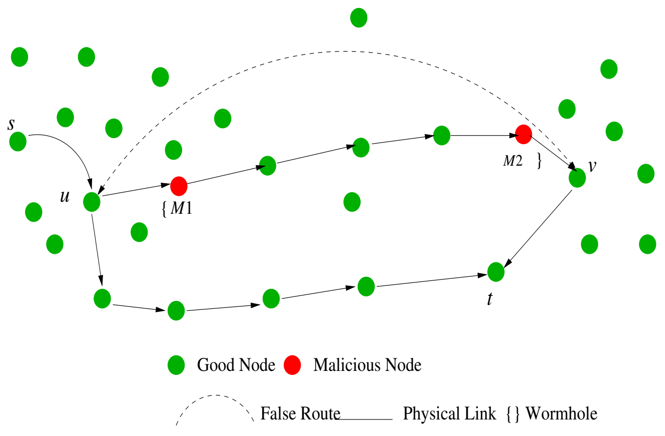

Figure 6.4: Closed Wormhole

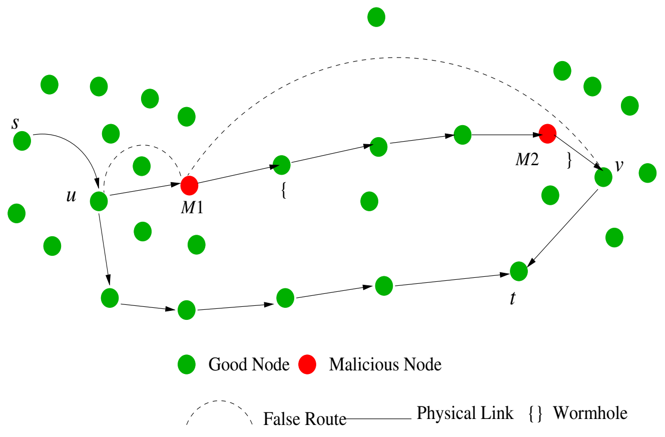

Figure 6.5: Half Open Wormhole

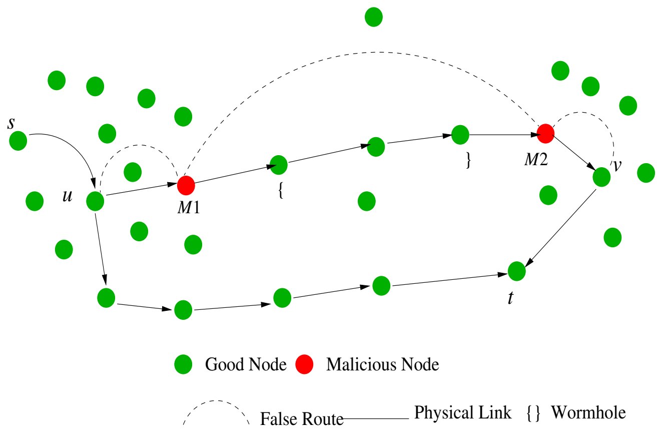

Figure 6.6: Open Wormhole

6.3 Related Work

We classify the existing approaches to mitigate wormhole attack into two categories:

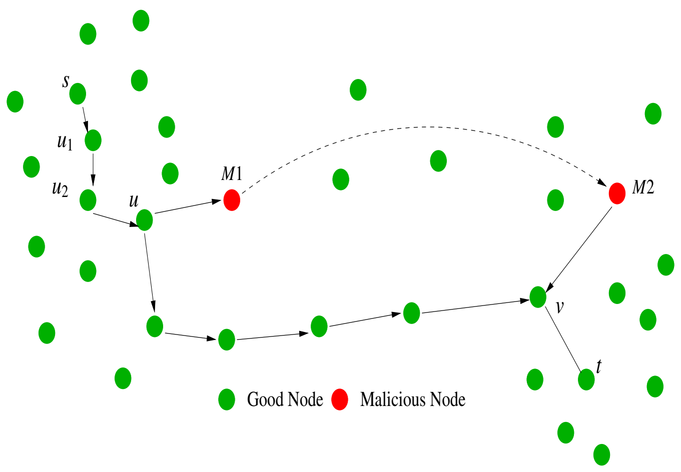

Figure 6.7: Path through wormhole

node-to-node mechanisms [HPJ03a, CBH03, BC93, HE04, KBS05, NAT06, PL07, PM08, KBS08] which require a trust level between two neighboring nodes and end-to-end mechanisms [WW07, THL$^+$07, VKSM08] that require trust only between the source and the destination.

6.3.1 Node-to-Node Mechanisms

In these approaches the detection mechanism is carried out at each node using local information gathered from the neighbors. Such an approach either assumes a trust level between two neighboring nodes or carries out some sort of neighborhood validation on its own. The trust level is achieved using cryptographic schemes or by neighborhood monitoring mechanism. In contrast, end-to-end schemes require only the trust between the sender and the receiver and the wormhole detection is carried out either by the source or by the destination only.

Hu et al. [HPJ03a] introduced the notion of a packet leash as a general mechanism for detecting and defending wormhole attacks. A leash is any information that is added to a packet designed to restrict the packet's maximum allowed transmission distance. They describe two types of leashes: geographical leash and temporal leash. A geographical leash ensures that the recipient of the packet is within a certain distance from the sender. They require the sender to include its location and the time of sending in the packet.

The receiver uses these values in addition to its own location and the time at which it receives the packet along with the maximum velocity of a node, to compute an upper bound on the distance to the sender. The protocol requires loosely synchronized clocks. In temporal leash the sender includes the time at which the packet is sent in the packet and the receiver uses this time and the time at which it receives the packet along with speed of light to bound the maximum travel distance. An implicit assumption is that packet processing, sending, and receiving delays are negligible. It requires tightly synchronized clocks.

In [BC93, CBH03] authors used response time to estimate the distance between the two nodes. The protocol is based on a secure request-response mechanism and requires accurate time measurements. The two nodes generate a series of fast bit exchanges. Starting from a seed, a node generates a number and sends its generated number (request) to the other node. The other node uses the received number to generate the next number (response) and send it back and the series continues for a fixed number of times. The time delay between sending the request and receiving the reply is then used to estimate an upper bound on the distance of the other node and hence determine whether it is in the neighborhood or not. Since the response is dependent on the request, it prevents the adversary from sending the response too quickly to project itself too close to the node and hence make it believe that it is in its neighborhood. The check is done by both the nodes. These schemes require special hardware for accurate time measurement and fast switching between the sender and the receiver.

Hu and Evans [HE04] used directional antenna to prevent the wormhole attack. They presented a solution which works by keeping an authentic set of neighbors at every node. Each node shares a secret key with every other node. They require that the antenna of the two nodes must be aligned (which is difficult to achieve in practice) in order for the nodes to communicate with each other. To discover its neighbors, a node, called the announcer, uses its directional antenna to broadcast a $HELLO$ message in every direction. Before the announcer adds the responder to its neighbor list, it verifies that it heard the message in the opposite direction. For every packet which a node receives, it checks if it is in the neighborhood set of the node, it accepts the message else discards it.

There are several approaches [KBS05, PM08, KBS08] based on maintaining a trust level of the nodes in the network. A node serves as a guard node for another node (or a link) if it is in the neighborhood of that node (or in the neighborhood of the nodes at the ends of the link). A guard node listens to the traffic going in and out of the node(/s) it is guarding and raises or reduces the trust level of the node (or of the link) depending upon the behavior of the node (/s). Pirzada et al. [PM08] has been designed to handle the wormhole attack against the DSR protocol. Liteworp works only for static networks. All these protocols require special hardware for a node to be able to listen to its neighbors in promiscuous mode.

In [NAT06] authors handled wormhole attack against Optimized Link State Routing ($OLSR$) protocol for ad hoc networks. In this method, a node detects spurious links by sending hello packets to each of its neighbors in neighbor discovery phase. This is done by assuming that wormhole attacks will cause longer packet latency compared to a normal single hop wireless latency. This happens for wormhole established using encapsulation. Once a link is suspected presence of wormhole is verified by exchanging encrypted probing packets between the two ends of the link. If the two nodes are actually connected through a wormhole then the response takes longer than a legal link and the link is isolated.

In [MGD07, ZMB08] authors presented topology based approaches to mitigate wormhole attacks. Each node maintains a local topology in its neighborhood. In [ZMB08], the approach is based on the hypothesis that in a dense bidirectional network each neighbor should be reachable from more than one node and thus cyclic structures must be present in the network. For each edge in its local topology, a node determines the number of cycles to which the edge belongs. If this number turns out to be zero for any edge a wormhole is suspected and the id of the suspected node is broadcast to inform the other nodes in the network. If the number of broadcasts for a node increases beyond a certain threshold the node is isolated. In [MGD07] the key notion is to determine the minimum number of nodes that can be packed without being connected in a fixed region. If a node detects a situation where this is violated in its neighborhood, it detects the presence of a wormhole. These algorithms detect wormhole with tunnel length of k-hop. Moreover they work only for static and dense topology.

In [PL07] Poovendran and Lazos used cryptography to encrypt the broadcast messages to one-hop neighbors so that the attacker cannot decrypt and relay old messages and, authenticate them too. They proposed one centralized approach using symmetric key distribution and another decentralized approach with a more expensive asymmetric mechanism for local key distribution.

6.3.2 End-to-End Mechanisms

Wang et al. [WBLW06] proposed a mechanism requiring only end to end trust. They require that the nodes know their positions and assume loosely synchronized clock. Each node attaches a (P, t) pair where P is the location of the node at time t. The destination checks if there is a conflict in the information sent by various nodes. It computes the moving speed of a node by examining its position at various times. If the speed is found to be more than a certain threshold v, they declare a wormhole on the path. To reduce the storage requirement and the computation time they modified the algorithm to Cell based Open Tunnel Avoidance ($COTA$) algorithm in which they divide the network area into a number of cells and the time into equal time slots. For every node they store only one record for one (cell no, time slot) pair. In this way some pairs belonging to the same node or cell are not compared. $COTA$ may miss the detection of some wormholes.

Chiu et al. [CL06] proposed a detection method called Delay Per Hop Indication ($DelPHI$). The source node compares the delay and the hopcount information of some disjoint paths between the source and the destination. If the delay per hop along any path is unusually longer than the delay on other paths, it detects a wormhole on that path. This approach is also based on the assumption that the delay along a wormhole link is much higher than a normal link which is true only for the wormholes launched by encapsulation.

Several other groups [WW07, QSL07, THL$^+$07, SB07, VKSM08] were also working on the problem simultaneously to our work. Wang and Wong [WW07] gave an end to end approach in which the source estimates the number of hopcount on a shortest path to the destination. They assume that the nodes are uniformly distributed in the network (which may not be true always). They use node-density and the straight line distance between the source and the destination to estimate the number of hopcounts on a shortest path between the source and the destination. If a path shows less hopcount, a wormhole is suspected on the path. They confirm the presence of wormhole using $TRACING$ procedure to locate the ends of wormhole on the path assuming the fact that there exist multiple paths between the source and the destination including the wormhole link.

Qian et al. [QSL07] proposed a technique to detect wormhole for multi-path routed *MANETs*. They proposed statistical analysis of multi-path collected during route establishment for a most frequent and second most frequent link. The idea is based on the fact that the statistic used will have higher value for a network under wormhole.

In [THL+07] wormhole detection is based on the round trip time between a node (on the path) and the destination. This approach is also based on the hypothesis that the delay along a wormhole link is much higher than a normal link which is true only for the wormholes launched by encapsulation. The higher delay could also be due to other reasons like congestion and almost full buffer. Authors propose no way to validate that the high delay on a path is due to the presence of a wormhole link. $WORMEROS$ [VKSM08] proposes a method to verify that a link is a wormhole link by using frequency hopping.

In [SB07] Su and Boppana presented the impact of in-band wormhole attack against secure routing protocols like $Ariadane$ [HPJ02], SRP [PH02] and $endairA$ [ABV06]. The attack is launched by the exchange of authentication information between the two colluding nodes. They propose two solutions to mitigate against in-band wormhole attack. One is to prioritize the processing time and transmitting time of route request so that the differences between the delays of falsified requests/replies and legitimate requests/replies is large. They use this difference to detect the presence of a wormhole on a path. In the second approach, they have proposed a distributed and adaptive statistical profiling technique to filter $RREQs$ (by destination) or $RREPs$ (by source) that have excessively large delays. Since different $RREQs$ take varying number of hops, it gives an upper bound on the per hop time of $RREQ/RREP$ packets so that most normal packets are retained and most falsified packets are filtered. The approach handles only in-band (launched by encapsulation) wormholes.

6.3.3 Wormhole detection for sensor networks

Some protocols to mitigate wormhole attacks in sensor networks have also been proposed. In [ZMB08], a central authority suspects a wormhole if it detects a sudden increase in the number of neighbors of any node or a sudden increase in the number of short paths between any two nodes. It uses estimation model to estimate the number of short paths between any two nodes. The base station is responsible for detecting and isolating the malicious nodes.

Approach of Madria et al. [MJ08] is based on neighborhood monitoring. Each node monitors its parent for data packet forwarding. If a node detects that a data packet has been dropped or tampered with, it considers its parent or a remote neighbor connected by a wormhole.

In [HCJ07] and [LS10] also authors proposed wormhole detection mechanisms for a sensor network. Both papers are based on the idea that if wormhole is present in the network it significantly increases the one-hop neighbors of a node. They count the number of nodes in the 1-hop/2-hop neighborhood of a node and suspects the presence of a wormhole if there is a sudden increase in this number.

6.4 Assumptions and Notations

6.4.1 Network Assumptions

We assume that the authentication of keys can be performed by pairwise secret keys or digital signatures. Researchers have proposed different algorithms [ZXSJ03, LN03, DDHV03, CPS03, MJ08] for the distribution of pairwise shared secret keys using various approaches. A Message Authentication code (MAC) generating algorithm [KBB97] takes a message and a secret key shared between the sender and the receiver to generate a secret code. The receiver also uses the shared key to generate the MAC code and authenticates the received message by matching it with the received MAC code. We also assume that the network drops packets only due to wormhole. We do not assume any clock synchronization.

6.4.2 Node Assumptions

Every node is equipped with a global positioning system (GPS) so that it knows its geographic location. Though the computation power of the nodes is limited, it is enough to carry out the computations required by security mechanism such as calculation and verification of digital signatures and, calculation of the MAC code.

6.4.3 Model of Attacker

The attackers do not have the capability to acquire the secret keys nor the computation power to compute MAC codes. The attacker may use encapsulation, out-of-band channel, high power transmission or packet relay to tunnel the packets through long distances without interfering with the signals sent by the good nodes. The attacker has a total control over the wormhole.

6.4.4 Notations

If pairwise keys are used to encrypt a message, K_{AB} denote the symmetric shared key between the nodes A and B. $MAC_{K_{AB}}(M)$ represents the encrypted MAC code on the message M using the key K_{AB}.

Geographic location of a node A is denoted by P_A. The maximum error in location is denoted by δ. If a packet is forwarded by a node A at recorded location P_A and it arrives at node B at recorded location P_B then the real distance d_{AB} traveled by the packet between A and B lies between $||P_A - P_B|| - 2\delta$ and $||P_A - P_B|| + 2\delta$.

6.5 Proposed solution to mitigate the effect of wormhole attack

We propose an End-to-End scheme to secure a network against Wormhole attack (EEW). The proposed solution assumes trust only between the source and the destination. The assumption holds in most of the conditions. Once a route has been established, existence of wormholes is examined several times during the lifetime of the route. The detection packets may be sent separately or the information may be attached to the routing packets or the data packets.

6.5.1 Algorithm

For the ease of presentation we assume that a path has been established using a routing protocol like $AODV$ or DSR and the source sends a wormhole detection packet (WDP) to check the existence of wormhole on the path. However in simulations we attach the same information in $RREQ$ packets and examine the paths while it is being established. When the source node sends a wormhole detection packet each node on the path attaches its location before forwarding the packet. The distance traveled by the packet is calculated at the destination by adding the distance traveled in each hop. Let d denote this distance. Let r_{max} be the maximum communication range between any two nodes. The protocol is based on a simplistic idea that any packet from source to destination must travel at least $\lceil d/r_{max} \rceil$ hops. For example, if $d = 9m$ and $r_{max} = 2m$ then Figure 6.8 shows that a packet from the node s to t must travel through the nodes $n_1, n_2 \dots n_4$ resulting in a hopcount of 5.

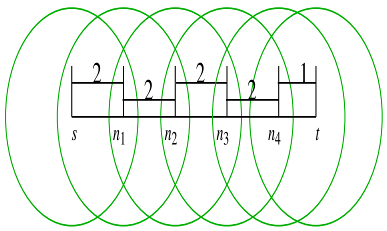

Figure 6.8: Example to illustrate the lower bound on the length of the wormhole tunnel

Note that if straight line distance is used (as is done in Wang and Wong [WW07]) instead of the traveled distance, some paths under attack may go undetected. For example consider Figure 6.9, distance traveled is 28 units whereas the straight line distance is 20 units. Thus a path showing hopcount less than 10 is declared to be under attack by Wang and Wong whereas in our approach a path showing hopcount less than 14 will be declared under attack. Hence a path with hopcount 11 (as shown in the Figure 6.9) will go undetected in their approach whereas it will be detected by our scheme.

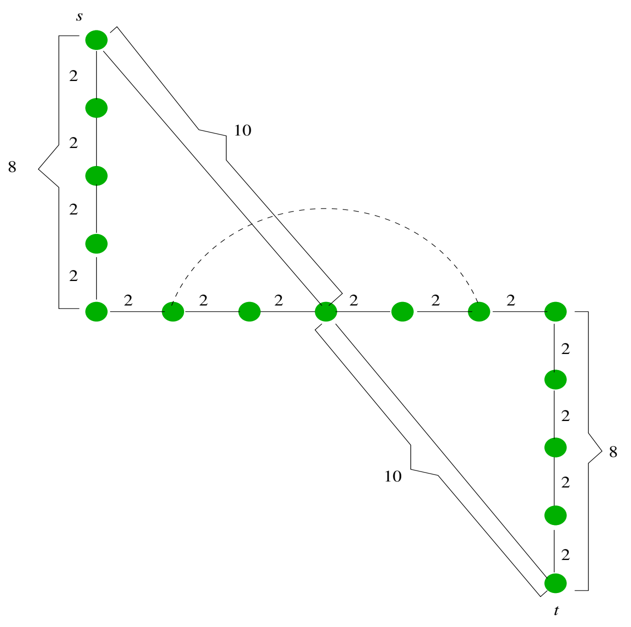

Figure 6.9: Counter example for Wang and Wong's mechanism

Let k be number of hops along a path P between the source s and the destination t and d be the distance traveled by a packet along P. We prove the following two theorems one of which provides an upper bound on the length of the wormhole tunnel. The wormhole is detected if the length of its tunnel is greater than this bound.

Theorem 6.1 *If $k < \lceil d/r_{max} \rceil$ then there is a wormhole on the path.*

Proof We will prove the result by proving that on a normal good path, number of hops is at least $\lceil d/r_{max} \rceil$. See Figure 6.8. Clearly, the number of nodes on P is minimum when the nodes are placed as far apart as possible and P lies along a straight line between s and t. Since two consecutive nodes on P cannot be placed farther than r_{max}, distance traveled by a packet along P is $\leq kr_{max}$ or $k \geq \lceil d/r_{max} \rceil$. Thus, if $k < \lceil d/r_{max} \rceil$ there must be a wormhole tunnel of length greater than r_{max} on P. ■

However, the converse of the above theorem is not true in general. That is, there may be a wormhole on a path and $k \geq \lceil d/r_{max} \rceil$. There may be lots of closely placed nodes between s and another node u and then there is a long tunnel between u and t. For example in figure 6.7, if $r_{max} = 2m, d = 10m$ so that $\lceil d/r_{max} \rceil = 5$, but $k = 7$. Let $dist(S, u_1) = 1m$, $dist(u_1, u_2) = (1 + \epsilon)m$, $dist(u_2, u) = 1m$, $dist(u, M1) = (1 + \epsilon)m$, $dist(M2, v) = 1m$, and $dist(v, t) = (1 + \epsilon)m$ then $dist(M1, M2) = (4 - 3\epsilon)m$. The reason is that there are 6 nodes covering a distance of $(6+3\epsilon)m$ with a long tunnel of length $(4-3\epsilon)m$. Thus there is a wormhole on the path but $k > \lceil (d/r_{max}) \rceil$.

In the following lemma we bound the number of good nodes that may occur on a good path spanning some distance. The idea is if n_1, n_2, n_3 are on some path then n_3 must be outside the range of n_1. This property is satisfied in most of the routing protocols like $AODV$ and DSR. In $AODV$, Suppose n_1 broadcasts a route request packet. If both n_3 and n_2 are in the range of n_1, both of them will receive the packet. If n_3 is in the range of n_2 also, it will later receive the packet from n_2 but will discard it as a duplicate. Hence no path will be setup through n_1, n_2, n_3.

Lemma 6.2 *Let S_i denote the interval $(ir_{min}, (i+1)r_{min}]$ and $d \in S_i$ for some integer i then $k \leq 2i+1$.*

Proof We prove the claim by induction on i.

For $i = 0, d \leq r_{min}$, clearly then t is neighbor of s and $k = 1$. Let the result holds for $i \leq x$. That is for any node D_i whose distance d_i from s along P satisfies $ir_{min} < d_i \leq (i+1)r_{min}$, the number of hops k_i from s to D_i satisfies $k_i \leq 2i+1$ for all $i \leq x$. Let D_{x+1} be a node whose distance d_{x+1} from s along P satisfies $(x+1)r_{min} < d_{x+1} \leq (x+2)r_{min}$. Consider the part Q of P between s and D_{x+1}. Let D_l be the neighbor of D_{x+1} on Q, then either $d_l \in S_i$ for some $i \leq x$ or $d_l \in S_{x+1}$ In the first case, induction applies and and hence $k_l \leq 2i+1$. Then $k_{x+1} = k_l + 1 \leq 2i+2 \leq 2x+2 \leq 2(x+1)+1$. In the second case, we cannot apply induction. In this case, let $D_{l'}$ be the neighbor of D_l on Q. Then, $d_{l'} \in S_i$ for some $i \leq x$ and hence $k_{l'} \leq 2i+1$. l' cannot be in S_{x+1} for else D_{x+1} would be in the range of $D_{l'}$ and hence they would be neighbors. Thus $k_{x+1} = k_{l'} + 2 \leq 2i+3 \leq 2x+3 = 2(x+1)+1$. ■

From the above lemma it follows that $k < 2d/r_{min} + 1$ or $d > (k-1)r_{min}/2$. In the following theorem, we show that the converse of Theorem 6.1 holds if the tunnel is long enough.

Theorem 6.3 *If there is a wormhole on a path and the length of the tunnel is* $\geq (\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$ *then* $k < \lceil d/r_{max} \rceil$, *where* $p = r_{max}/r_{min}$.

Proof Suppose there is a wormhole on a path $S = u_1, u_2, \ldots u_{k+1} = D$. Since there is a wormhole, there exists a pair of vertices u_i, u_{i+1} which form a wormhole. Also, the distance between u_i and u_{i+1} is $\geq (\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$ by assumption. Then,

$$d = dist(S, u_i) + dist(u_i, u_{i+1}) + dist(u_{i+1}, t)$$
$$> \tfrac{(i-1-1)}{2}r_{min} + (\tfrac{2p-1}{2p}k + \tfrac{2}{p})r_{max} + \tfrac{(k-i-1)}{2}r_{min}$$
$$= \tfrac{(i-1-1)}{2}\tfrac{r_{max}}{p} + (\tfrac{2p-1}{2p}k + \tfrac{2}{p})r_{max} + \tfrac{(k-i-1)}{2}\tfrac{r_{max}}{p}$$
$$= \tfrac{2pk+1}{2p}r_{max} > kr_{max}.$$
$$\Rightarrow k < d/r_{max} \leq \lceil d/r_{max} \rceil. \qquad \blacksquare$$

Theorem 6.1 shows that if $k < \lceil d/r_{max} \rceil$ then we are sure that there is a wormhole on the path. Theorems 6.1 and 6.3 can be combined to give the following algorithm: discard a path if $k < \lceil d/r_{max} \rceil$. Theorem 6.1 guarantees that no good path is discarded while Theorem 6.3 guarantees that long wormholes are always detected. With simulation, we will show that we are able to detect wormholes of length much smaller than $(\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$ also.

When the source sends a wormhole detection packet (WDP), it includes the source *id*, the destination *id*, the WDP packet *id*, message if any, its location, and hopcount field set to 1, in the packet and, encrypt it with say MAC code using K_{st}, the shared key between the source and the destination. Each intermediate node generates a MAC code of the wormhole detection packet after appending its id and location and, appends it to the wormhole detection packet. When the destination receives WDP packet, it checks the authenticity of the received packet using its shared key with the intermediate nodes. It also calculates the distance traveled by the packet using the location information and checks the hopcount announced by the path. If it is less than $\lceil d/r_{max} \rceil$ it detects a wormhole on the path and broadcasts a WDP reply packet (WDP_REP) with wormhole detection flag (WDF) set to 1, to inform the source. The delivery of wormhole detection packet is shown in Figure 6.10. Algorithm 4 summarizes the EEW algorithm.

$S : HC = 1$

$h_s = MAC_{K_{st}}(s, t, M, id, P_s, HC)$

$S- > A : (s, t, M, id, P_s, HC, h_s)$

$A : HC + +$

$A : h_A = MAC_{K_{At}}(ReceivedPacket, A, P_A, HC)$

$A- > B : (ReceivedPacket, A, P_A, HC, h_A)$

$B : HC + +$

$B : h_B = MAC_{K_{Bt}}(ReceivedPacket, B, P_B, HC)$

$B- > t : (ReceivedPacket, B, P_B, HC, h_B)$

Figure 6.10: Delivery of wormhole detection packets, HC is the hopcount, M is the message, $MAC_{K_{At}}(M)$ is MAC code on M using the key K_{At} and P_A is the location of a node A

0.6em 2ex

```
    Assumption: We assume that a path has been established using a routing protocol like
4.1 $AODV$ or $DSR$. Source unicasts a $WDP$ packet on the path established by a routing
    protocol.
4.2 When an intermediate node receives the $WDP$ packet, it does the following steps:
      it increases the hopcount by one,
      it appends its id, location and $MAC$ code of its $WDP$ to the received packet and
      forwards the packet to the next hop node on the path.
4.3 When the destination receives a $WDP$ packet, it does the following steps:
      it verifies that all the $MAC$ codes have been computed correctly,
      it verifies that all pairs of consecutive nodes are in direct range of communication with
      each other,
      it verifies the hopcount by traversing the $THL$,
      it extracts the locations of all the nodes from the packet and computes $d$ by adding the
      distance traveled by the packet per hop.
      if (hop-count in the detection packet $< \lceil d/r_{max} \rceil$) then
          it sets the wormhole detection flag ($WDF$) in the $WDF\_REP$ packet to 1.
      else
          it clears the $WDF$.
      end
    It broadcasts the $WDF\_REP$ packet to inform the source.
```

Algorithm 4: EEW: End-to-End scheme against Wormhole attacks.

Format of WDP and WDP_REP are shown below:

WDP **Format**								
Type	Src id	Dest id	Message	WDP id	THL	Loc List	Hopcount	List of MAC codes of WDPs

WDP_REP **Format**				
Type	Src id	Dest id	WDF	MAC code of WDP_REP

Since the solution relies on the location and hopcount information provided by the nodes on the path, an attacker may insert false information in the detection packet. In the next section we present how to check the attacker from lying too much. In section 6.5.3, we will show that with the GPS accurate upto 15 feet, the effect of error in the location information does not affect the detection capability of our protocol. However, the error may sometimes lead to false positives.

6.5.2 Check the attacker from lying

The above scheme requires that each node attaches its location information in the detection packet. The scheme works fine in closed wormhole where the two end points of the tunnel are not visible to the other good nodes on the path. That is, in the packets examined by the destination no node has lied about its position. However, in half-open or open wormhole an attacker (or colluding attackers) may lie about its (their) position(s). To check an attacker from lying too much about its location, destination also verifies whether two consecutive nodes are with in r_{max} range of each other. Consider figure 6.6, to announce that the distance between $M1$ and $M2$ is small, one or both of $M1$ and $M2$ may lie about their position. In either case, at least one of them will go out of the range of its good neighbor and hence the wormhole will be detected.

An attacker may also lie about its hopcount from s. It may put a large value in the hopcount of the detection packet. Let $d = 20m$ and $r_{max} = 2m$. Since $M1$ and $M2$ are colluding $M1$ may have an idea of the location of $M2$. Let d_{M1M2} denote the distance between $M1$ and $M2$. Let $d_{M1M2} = 16m$. Then $M2$ may increment the hopcount by $d_{M1M2}/r_{max} = 8$. The destination will then get the packet with the right hopcount value 10, and hence the wormhole will go unnoticed. To detect such wormholes, we use the

THL in the detection packet. The attacker may be able to generate a fake list of *id*s, but it will not be able to generate their MAC code as it neither has the keys nor enough computational power. Hence by examining the THL, wormhole will be detected. Note that this may affect the tunnel length mentioned in Theorem 6.3 at most by $2r_{max}$.

6.5.3 Effect of error in the location information

Every node is equipped with a GPS so that it knows its geographic location. We show that the effect of error in location information on the detection capability of the algorithm is negligible. In a very few cases some good short paths may remain undiscovered.

Let P_i and P_{i+1} denote the recorded location of two consecutive nodes u_i and u_{i+1} on the path and let P'_i and P'_{i+1} be their real positions. Then $||P_i - P_{i+1}||$ the recorded distance traveled by the packet lies between $||P'_i - P'_{i+1}|| - 2\delta$ and $||P'_i - P'_{i+1}|| + 2\delta$, where δ is the maximum error in the location information of any node. By summing it over all the hops we see that the computed distance d lies between $d' - 2k\delta$ and $d' + 2k\delta$ where d' is the actual distance traveled by the packet.

If $d = d' - 2k\delta$, then we are putting a looser lower bound on the number of hops of a good path. A wormhole may go undetected if the hopcount is greater than $\lceil d/r_{max} \rceil = \lceil (d' - 2k\delta)/r \rceil$ but less than $\lceil d'/r_{max} \rceil$ even if its tunnel is long. However, in a practical scenario, with very accurate GPS, the value of $2k\delta/r_{max}$ is a very small quantity and its effect is negligible. For example, if the real distance is $1250m$, $r_{max} = 250m$ and $\delta = 5m$ (> 15 feet). Let $k = 10$, then the recorded distance could be $1150m$. We rightly discard the paths with hopcounts less than $5 = \lceil 1150/250 \rceil$.

If $d = d' + 2k\delta$, then we are putting a tighter lower bound on the number of hops of a good path. Hence it will not affect the wormhole detection capability of the algorithm but we may have some false positives. That is, we may miss some good short paths. For example, if in the above scenario, the recorded distance is 1350m then it discards all paths of length less than $6 = \lceil 1350/250 \rceil$ and hence good paths of length 5 are also discarded.

6.5.4 Security Analysis

Our protocol is able to detect closed wormholes as well as open and half-open wormholes. Most of the algorithms designed to defend the ad hoc networks against various types of attacks suffer from false positives, (i.e. a good path is suspected to be under attack and is discarded) and false negatives (i.e. a path under attack escapes detection). Theorem 6.1 guarantees that in the absence of any error in the location, our algorithm does not give false alarms. In the previous section we showed that even in presence of error, wormhole detection capability of the protocol is not affected, however in a very few cases there may be some false alarms. Some wormholes of relatively short length ($< (\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$) may escape detection.

6.6 Overhead

In this section we present the overhead due to storage, communication and computation incurred by our protocol and compare it with other algorithms.

6.6.1 Storage and Communication Overhead

If there are k nodes on a path then the size of a detection packet is $O(k)$. Hence the communication time per packet per hop is $O(k)$ which is same as that of end-to-end mechanism of Wang et al. Storage used is only $O(k)$ at the destination. This is much less than $O(km)$ space used by end-to-end mechanism of Wang et al., where m is the total number of packets examined by their scheme. In $COTA$, they store only c_1 number of packets instead of all the m packets, where c_1 is a constant. The storage space used by $COTA$ is then $O(c_1k)$. However, the constant c_1 decreases as a factor, called *sensitivity*, increases by more than a linear rate. That is, c_1 and hence the amount of storage space can be made arbitrarily small by making *sensitivity* large. However, large sensitivity leads to large number of false positives. Thus there is a trade-off between the storage space and the number of false positives/detection capability (small *sensitivity* leads to missing the detection of some real wormholes).

6.6.2 Computation Overhead

Computing the MAC code at the intermediate nodes and verifying them at the destination does not take much time. Checking whether consecutive nodes are in direct range or not involves only $O(k)$ pairs. Hence this step takes $O(k)$ time. Similarly, computing the distance between consecutive nodes and adding them to compute the distance between the source and the destination requires only $O(k)$ computation time. Examining the THL of length $O(k)$ also takes $O(k)$ time. This is much less than the $O(km^2)$ time of end-to-end mechanism and $O(c_2km)$ time of $COTA$. Again the constant c_2 decreases as the *sensitivity* increases, by more than a linear rate. That is, c_2 and hence the computation time can be made arbitrarily small by making *sensitivity* large. Thus there is also a trade-off between the computation time and the number of false positives/detection capability. Table 1.2 summarizes the comparison of our EEW algorithm with the basic approach of Wang et al. and $COTA$.

6.7 Simulation Study

We tested the performance of our wormhole detection algorithm through simulation. Network topology was generated randomly in NS2. We studied several connection pairs. For each pair we studied the length of the wormhole tunnel beyond which our protocol was always able to detect the wormhole. We also studied the effect of error in the positions of the node on the wormhole detection capability. Out of several scenarios studied, only in one case a good path was deleted and only in two cases, the length of wormhole tunnel increased slightly due to error, but the length never crossed the bound claimed in Theorem 6.3.

6.7.1 Simulation Design

Parameters used for simulation are listed in Table 6.1. $AODV$ was used as the routing protocol and was modified to combine with the detection mechanism. Source initiates the route request ($RREQ$) with attached location information. When the intermediate node forwards the route request it also attaches its location to the request packet. When

Simulation Duration	1000 seconds
Simulation area	$2000m * 2000m$
Number of mobile nodes	50
Transmission Range	$150 - 350m$
Movement model	Random waypoint
Traffic type	$CBR(UDP)$
Data payload	512 bytes

Table 6.1: Simulation Parameters

the destination receives the request, it computes the distance (d) traveled by the $RREQ$ packet using location information attached by each node on the path. Finally, it calculates $\lceil d/r_{max} \rceil$ and compares it with the received hopcount in $RREQ$ packet. If $\lceil d/r_{max} \rceil$ is less than or equal to the received hopcount, it sends $RREP$ otherwise it discards $RREQ$ packets. Hence the source node receives an $RREP$ from a secure path and not from the path under attack. Hence a secure path avoiding the wormhole is established.

6.7.2 Simulation Results

For a connection scenario between nodes s and t, wormholes of varying tunnel length were created. It was found that whenever the tunnel length was greater than $(\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$, the wormhole was always detected. In fact, we were able to detect and isolate wormholes of length much shorter than this. For example for the connection pair (node 1, node 11), $r_{max} = 300$, $r_{min} = 150$, $p = 2, k = 7, (\frac{2p-1}{2p}k + \frac{2}{p})r_{max} = 1895$, we were able to detect wormholes of length $1300m$. We studied more connection scenarios and similar results were obtained. Table 6.2 summarizes our results.

6.7.3 Simulation Results to study the effect of error

To study the effect of error for a connection scenario between nodes s and t, we introduced maximum error $\delta = 15$ feet for each node. We studied the effect of error in two parts. First, we added $2k\delta$ to the computed distance at the destination. We studied 20 connection

Conn. between	k	h	L	Length of Wormhole studied	Length of Wormhole undetected	Length of Wormhole detected
$(8, 39)$	5	6	1372	$250 - 1400$	< 800	≥ 800
$(1, 11)$	7	8	1895	$350 - 1900$	< 1300	≥ 1300
$(6, 49)$	5	6	1645	$350 - 1650$	< 1150	≥ 1150
$(0, 47)$	6	7	1590	$350 - 1650$	< 950	≥ 950
$(21, 9)$	5	6	1525	$350 - 1550$	< 850	≥ 850

Table 6.2: Summary of Simulation Results of EEW. $L = (\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$, k = received hopcount, h= computed hopcount$\lceil d/r_{max} \rceil$

pairs with nodes of variable range and r_{max} = 350 for the first case. We could not find even a single case in which a good path was deleted because of error. We reduced r_{max} of the network to 150. With this maximum range, no path was established with the parameters stated in Table 6.1. We reduced the network area to 1000m by 1000m. In this scenario, 30 s and t pairs were studied and it was found that there was only one case ($s = 0$, $d = 40$ with received hopcount = 1 calculated hopcount = 2) in which a good path of length 1 was deleted. Secondly, we subtracted $2k\delta$ from the computed distance at the destination. 5 connection pairs were studied to observe the effect of subtracting $2k\delta$ from the distance. As earlier, on each path wormholes of varying tunnel length were studied. It was found that only in 2 cases wormhole tunnel length beyond which the wormhole was detected increased by a small amount. Table 6.3 summaries our results for this case. Note that even the increased length is with in the claimed bound.

Conn. between	k	$h1$	L	Length of Wormhole studied	Length of Wormhole detected without error	Length of Wormhole detected
$(8, 39)$	5	5	1372	$250 - 1400$	< 850	≥ 850
$(1, 11)$	7	8	1895	$350 - 1900$	< 1300	≥ 1300
$(6, 49)$	5	5	1645	$350 - 1650$	< 1200	≥ 1200
$(0, 47)$	6	7	1590	$350 - 1650$	< 950	≥ 950
$(21, 9)$	5	6	1525	$350 - 1550$	< 850	≥ 850

Table 6.3: Effect of Error on the performance of EEW. $L = (\frac{2p-1}{2p}k + \frac{2}{p})r_{max}$, k= received hopcount, h1= computed hopcount $\lceil d/r_{max} \rceil$ with error

Chapter 7

Computing Minimum Exposed Path to Attack ($MEPA$)

7.1 Introduction

None of the existing protocols handles all types of attacks. Hence it is imperative to find a solution that would reduce the impact of attacks on routing. In this chapter, we present an algorithm to find a path between a pair of nodes such that the path is at maximum distance from the nodes which are in danger of attack. We call such paths as Minimum Exposed Path to Attack ($MEPA$). To the best of our knowledge the problem has not been addressed earlier for ad hoc networks. The related problem in the context of sensor networks is 'maximal breach path' problem [MKPS01, MLL03, LWF03, LDR03, HRS05, BAS06, LWY09]. Maximal breach path is defined as the path which is as far away as possible from the sensors i.e the minimum distance from sensors is maximized on this path. The path finds the area of low observability from the sensor nodes. A solution to 'maximal breach path' problem can be used to compare two given configurations of sensors in a battlefield or, to add new sensors to a network to increase the observability or the coverage area.

Our algorithm (henceforth referred to as $MEPA$) consists of a preprocessing phase where all the nodes compute their distances from the endangered nodes. Once the initial distances are computed, $MEPA$ routes are discovered in a manner similar to that in

$AODV$ in the route establishment phase. Whenever a route with greater distance from the endangered node is found, the previous one is discarded and the new one is kept. We show, theoretically as well as through simulations, that the algorithm converges in $(O|P|)$ time after a preprocessing step where $|P|$ is the length of the $MEPA$ route. The preprocessing step takes $O(D)$ time where D is the diameter of the network. When a node moves, maintenance phase takes $O(D)$ time to recompute the distances and the new $MEPA$ route is computed in $(O|P|)$ time. Assuming that the packets are received from the shortest path first, the algorithm computes a shortest $MEPA$ route. As in $AODV$ a path with minimum number of hops is shortest.

We simulated our protocol in $NS2$. $MEPA$ was compared with $AODV$ in the absence of endangered nodes and the performance was found to be comparable. We also compared $MEPA$ with $RDVBS$ and $DENG$ to address the blackhole attack and with EEW to mitigate wormhole attack. $MEPA$ out performs all the three algorithms in presence of blackhole/wormhole/both attacks in terms of packet delivery ratio and average end to end delay. Routing overhead of $MEPA$ is slightly higher due to pre-processing phase when the number of communicating pairs is less. However as the number of connections increase, the time taken in the path establishment phase starts dominating and $MEPA$ performs better than $DENG$ and $RDVBS$ and, is comparable to EEW in terms of routing overhead.

7.2 Problem statement and Notations

Endangered nodes are defined as the nodes in a network which are under the danger of attack. 'Minimum Exposed Path to Attack ($MEPA$)' is a path which is farthest from the endangered nodes, i.e. a path whose minimum distance from the endangered nodes is maximized.

Definition 7.2.1 *Let E be the known subset of nodes that are in danger of attack or jamming. For any node u in the network,*

1. *let $\delta(u)$ denote the minimum distance of node u from the set E i.e. $\delta(u) = min_{e \in E}\{dist(u, e)\}$, where $dist(u, e)$ is the distance of u from e in terms of number of*

hops.

2. *let $Nb(u)$ denote the set of neighbors of u. Define $pr(u)$ to be the node, in the neighborhood of u, through which the distance of u from E is minimum i.e. $pr(u) = u_i$ where $u_i = argmin_{v \in Nb(u)}\{\delta(v)\}$. For example, in Figure 7.1, let $E = \{e_1, e_2, e_3\}$. $\delta(u_1) = 1, \delta(u_2) = 1, \delta(u_3) = 1, \delta(u_4) = 2, \delta(u_5) = 3, \delta(u_6) = 2, \delta(u_7) = 2, \delta(u_8) = 2$, $pr(u_4) = u_1, pr(u_5) = u_4, pr(u_6) = u_3$, $pr(u_7) = u_2$ and $pr(u_8) = u_2$.*

3. *For a set S, $Nb(S) = \cup_{u \in S}\{Nb(u)\}$. For example, $Nb(E)$ in Figure 7.1 is $\{u_1, u_2, u_3\}$.*

These terms are used in the pre processing phase of our algorithm.

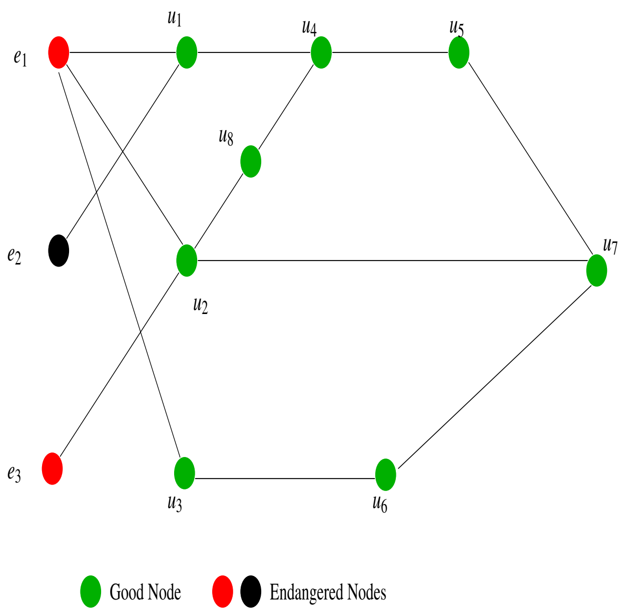

Figure 7.1: Computation of $\delta(u)$ and $pr(u)$.

Definition 7.2.2 *For a path P between s and t, define $dist(P, E) = min_{u \in P : u \neq s,t}\{\delta(u)\}$ then $MEPA(s, t) = argmax_P\{dist(P, E)\}$.*

Definition 7.2.3 *Let a node s wants to establish a $MEPA$ route to another node t. Let $MEPA(s, v)$ denote the 'Minimum Exposed Path to Attack' from s to a node v . Let*

$DEN(s,v)$ denote the distance of the $MEPA$ route $MEPA(s,v)$ from the endangered nodes. Then,

1. $DEN(s,v) = min\{max_{v_i \in Nb(v)}\{DEN(s,v_i)\}, \delta(v)\}$ *for* $v \neq s,t$,

   $DEN(s,t) = max_{v \in N(t)}\{DEN(s,v)\}$ *and* $DEN(s,s) = \infty$.

2. $p(s,v) = argmax_{v_i \in Nb(v)}\{DEN(s,v_i)\} for\ v \neq s$, *and* $p(s,s) = Null$.

$p(s,v)$ denotes the next hop for v on the reverse route of $MEPA$, we call it the precursor of v on the $MEPA$ route. These terms are used in the route-establishment phase of the algorithm.

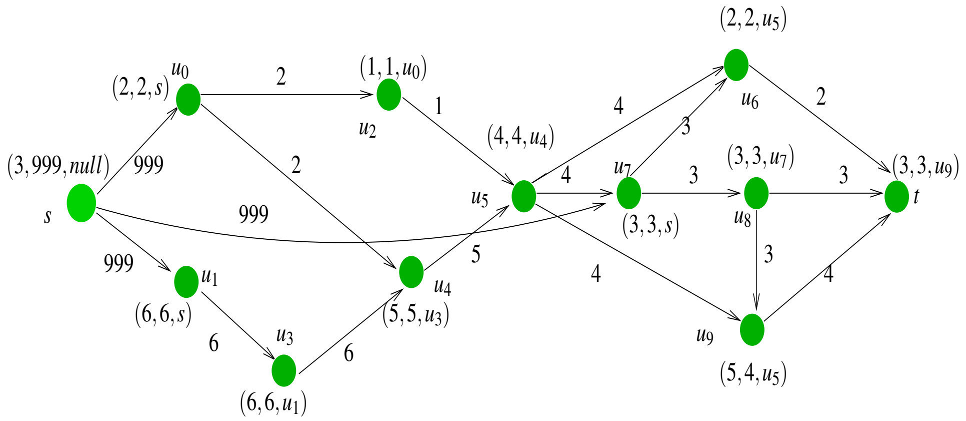

Figure 7.2: Computation of $DEN(s,u)$. Each node u is labeled with the triplet $(\delta(u)$ $, DEN(s,u), p(s,u))$ and edge (u,v) is labeled with $DEN(s,u)$.

Consider Figure 7.2. Each node u is labeled with the triplet ($\delta(u)$, $DEN(s,u)$, $p(s,u)$) and edge (u, v) is labeled with $DEN(s,u)$. Consider node u_5. Three routes from s to u_5 have 'distance from E' as $1, 2$ and 5, maximum being 5. $DEN(s,u_5) = min\{5, \delta(u_5)\} = min\{5,4\} = 4$. Since distance 5 was received from the node u_4 therefore $p(s,u_5) = u_4$. As can be seen from the figure that this indeed is the next hop at u_5 on the reverse $MEPA$ route from t to s. $DEN(s,s)$ is set to ∞ so that $DEN(s,u)$ is set appropriately for all $u \in Nb(s)$. Note that the distances of s and t from the endangered nodes should not be considered while computing the $MEPA$ distances . See Figure 7.3, if we will include $\delta(t)$ or $\delta(s)$ in the computation of $MEPA$ route, routes like P_2 will never be discovered if P_1 has already been established.

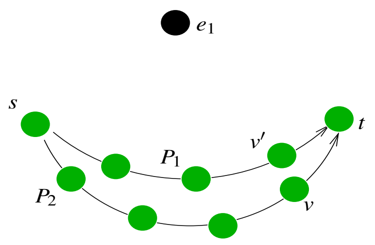

Figure 7.3: Effect of $\delta(t)$ and $\delta(s)$ on the computation of $MEPA$ route where e_1 is an endangered node.

7.3 Algorithm to compute a $MEPA$ route

We assume the existence of a mechanism that enables the nodes to detect the endangered nodes. In [ZL00, HL03, ZLH03] Lee et al have used intrusion detection techniques to detect the presence of an intruder in wireless ad hoc networks.

Our algorithm works in two phases. Phase-I is a 'preprocessing phase' in which all the nodes compute their distances from the endangered nodes. Once the distances from the endangered nodes have been computed, nodes can set up $MEPA$ routes in Phase-II. Due to the highly mobile nature of the nodes, the distances may have to be updated dynamically as the nodes move. This is done in the 'maintenance' phase. For the sake of simplicity and better understanding, we have presented the phases separately but in actual implementation 'maintenance' phase and the 'route-establishment' phase occur simultaneously.

7.3.1 Phase-I : The Preprocessing Phase

Let $in\_nbhd(u)$ be a flag that denotes whether u is in the neighborhood of E or not. Initially the flag $in\_nbhd(u)$ is off and the value of $\delta(u)$ is ∞ for all the nodes u in the network. Nodes compute their distances from E as follows:

1. Any node $u \in Nb(E)$ knows that it is in $Nb(E)$ when it hears from an endangered node. It sets its $in\_nbhd(u)$ flag to 1, $\delta(u)$ to 1 and broadcasts its distance to all its neighbors.

2. Let u be a node not in $Nb(E)$. When u receives $\delta(v)$ from its neighbor v, it updates its $\delta(u)$ as follows: if the distance of u from E is shorter through v than its current

value (that is, if $\delta(v) + 1 < \delta(u)$) then it updates $\delta(u)$ to $\delta(v) + 1$ and sets $pr(u)$ to v.

Whenever a node updates its distance from E, it broadcasts the updated distance to all its neighbors except to the one through which the new distance was computed. Algorithm 5 summarizes phase-I of $MEPA$.

Assuming that packets are received from the shortest path first, no updation is done in the preprocessing phase and the distances are established in $O(D)$ time where D is the diameter of the network. We will see in section 7.3.3 that, as the nodes move, distances are updated in the maintenance phase.

1em 2ex

```
    Initialization: $in\_nbhd(u) \leftarrow 0$, $\delta(u) \leftarrow \infty$
    The Preprocessing Phase: Computing distances from Endangered nodes
5.1 a node $x \in Nb(E)$ hears from an endandgered node:
    $in\_nbhd(x) \leftarrow 1,\ \delta(x) \leftarrow 1,\ pr(x) \leftarrow NULL.$
    it broadcasts its distance to all its neighbors.
5.2 when a node $u$ receives $\delta(v)$ from its neighbor $v$:
    if ( $\delta(v) + 1 < \delta(u)$ ) then
        $\delta(u) \leftarrow \delta(v) + 1$, $pr(u) \leftarrow v$, it forwards its $\delta(u)$ to all its neighbors.
    else
        it discards the packet.
    end
```

Algorithm 5: Phase-I of $MEPA$

We have used a packet called 'Distance Unit' (DU) to broadcast the 'distance from E'. '$\delta elta$' field in this packet holds the distance of the node from the endangered nodes. Format of the DU packet is

Distance Unit(DU)				
Type	DU id	Src id	Src Seq Number	$\delta elta$

7.3.2 Phase-II : Establish a $MEPA$ route

Let a node s wants to establish a $MEPA$ route to another node t. The algorithm works in a breadth first manner starting from s. $DEN(s, u)$ is initialized to 0 for all $u \neq s$ and

$DEN(s, s)$ is set to ∞. s broadcasts its $DEN(s, s)$ (included in route request packet) to its neighbors. Let $u \in Nb(s),\ u$ updates its $DEN(s, u)$ and $p(s, u)$ (explained below) if required and, broadcasts its $DEN(s, u)$ (included in route request packet) to its neighbors if updated. $u's$ neighbors then update their DENs and precursors if required and so on. In the process, a node may receive DENs more than once. Each time it receives a new DEN, it updates its information if required and broadcasts further (all this happens while processing the route request packets).

Suppose an intermediate node u receives $DEN(s, v)$ from a node v then u updates $DEN(s, u)$ and $p(s, u)$ as follows: If $DEN(s, v) > DEN(s, u)$ then $DEN(s, u)$ will be updated to $min\{DEN(s, v), \delta(u)\}$ and $p(s, u)$ will be updated to v. This case is exhibited in Figure 7.2 at node u_4. Suppose node u_4 receives $DEN(s, u_0) = 2$. Since $DEN(s, u_4)$ was initialized to 0 therefore $DEN(s, u_0) > DEN(s, u_4)$ and hence $DEN(s, u_4)$ is updated to $min\{\ DEN(s, u_0), \delta(u_4)\} = min\{2, 5\} = 2$ and $p(s, u_4) = u_0$. Later when it receives $DEN(s, u_3) = 6$, then since $DEN(s, u_3) > DEN(s, u_4)$, $DEN(s, u_4)$ is updated to $min\{DEN\ (s, u_3), \delta(u_4)\} = min\{6, 5\} = 5$ and $p(s, u_4)$ is updated to u_3.

When the destination t receives $DEN(s, v)$ from a node v then t updates $DEN(s, t)$ and $p(s, t)$ as follows: If $DEN(s, v) > DEN(s, t)$ then it updates $DEN(s, t)$ to $DEN(s, v)$ and $p(s, t)$ to v. t then sends a reply packet to v. Note that the destination may reply back to several nodes in case it receives a path with a higher DEN value later. The destination copies the value of $DEN(s, t)$ in the reply packet.

When an intermediate node v receives the reply packet, it makes an entry for the destination t, sets the next hop on the $MEPA$ path and copies $DEN(s, t)$ in its routing table. It also forwards the packet to its precursor $p(s, v)$ on the reverse path. In case a node receives multiple replies, it updates its routing table entries and forwards the reply packet only if received DEN is higher than the previous one. When the source node s receives the reply packet, it starts sending the data packets on that path. In case the source node receives multiple replies, it updates its $MEPA$ path only if the received DEN is higher than the previous one.

Notice that we need to store $DEN(s, t)$ at every intermediate node u so that when destination sends multiple replies with the updated DENs, this updation takes effect on

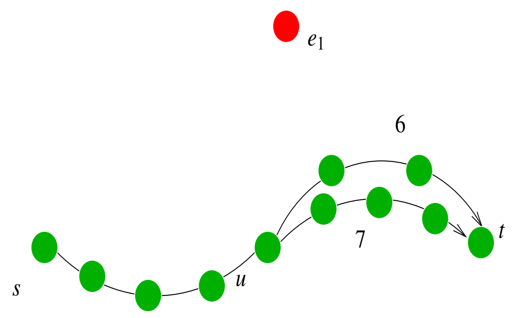

Figure 7.4: Updating $DEN(s, t)$ and the corresponding path at an intermediate node u

u as well. Consider the scenario in Figure 7.4. Suppose t sends a reply with DEN set to 6 first and then set to 7. u needs to store $DEN(s, t)$ to compare the stored DEN value and the received DEN value so as to be able to take a decision whether to update its next hop or not. If u receives reply with DEN value 6 before the one with DEN value 7 then it should update its table entries but if it receives DEN value 7 before 6 than it should not.

Next, we will show that this decision does not depend upon s and hence $DEN(s, t)$ can be stored in the entry for t in the routing table for u. Consider a path P for which u contains a value Δ for $DEN(s, t)$. Suppose while computing a $MEPA$ path between another source s' and t, t sends a route reply with $DEN(s', t)$ equal to Δ' through u. u updates its next hop for t, only if $\Delta' > \Delta$. Let us see the impact of doing this on P. See Figure 7.5, $MEPA$ path between s and t would be of the form $P1$ -$P2$ or $P1$-$P3$ or $P1$-$P4$. Note that due to s'-t communication it might change from one of these to another but would still remain $MEPA$ route i.e. the next hop in the entry for t in the routing table of u might change but the new route would still be a $MEPA$ route between s and t.

Algorithm 6 summarizes phase-II of $MEPA$. Processing of $MEPA\_REQ$ and $MEPA\_REP$ is shown in Figures 7.20, 7.21, 7.22 and 7.23. We replaced the hopcount field in the $RREQ$ and $RREP$ packets of $AODV$ by the DEN field. They are now called $MEPA\_REQ$ and $MEPA\_REP$ respectively. Similarly hopcount field in the routing table entry of $AODV$ was replaced with DEN. Formats of routing table entry of $MEPA$, $MEPA\_REQ$ and $MEPA\_REP$ packets are shown below:

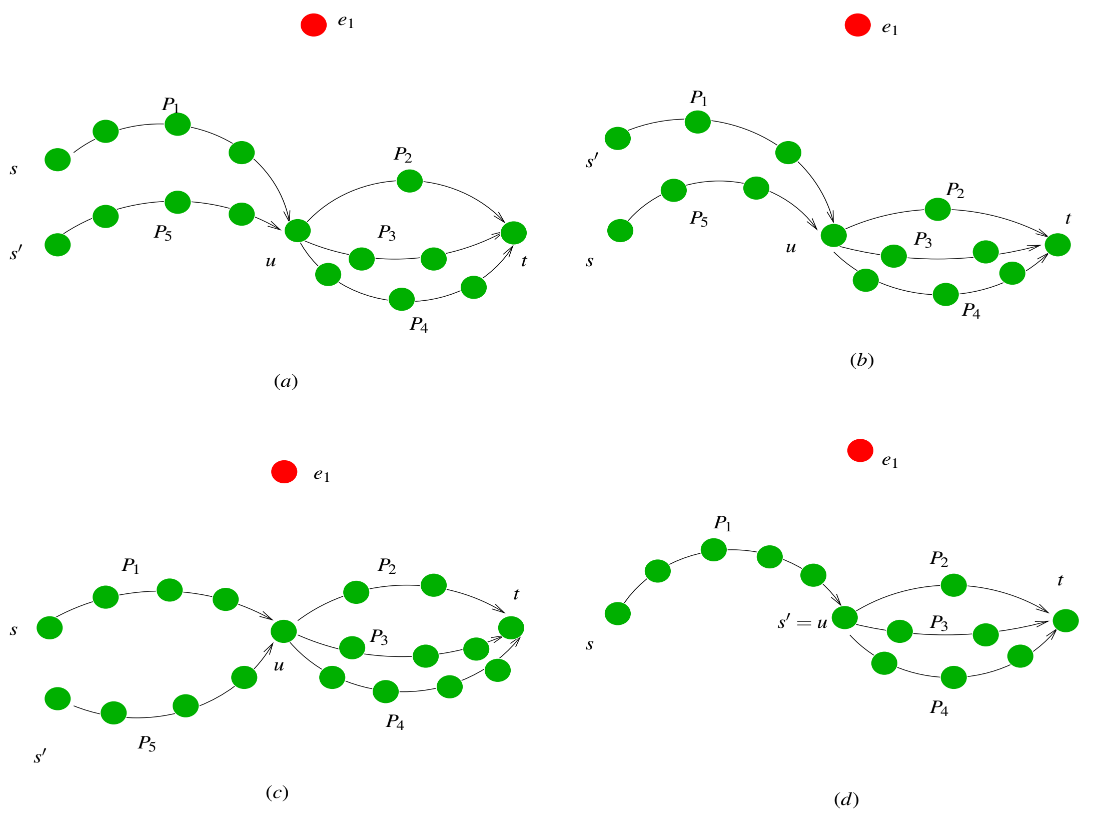

Figure 7.5: Impact of another communication on $DEN(s,t)$ and the corresponding path at an intermediate node u. (a) $DEN(P_1) < DEN(P_2) < DEN(P_5) < DEN(P_3) < DEN(P_4)$ (b) $DEN(P_1) < DEN(P_5) < DEN(P_2) < DEN(P_3) < DEN(P_4)$ (c) $DEN(P_1) < DEN(P_2) < DEN(P_3) < DEN(P_4) < DEN(P_5)$ (d) $s' = u$, $DEN(P_1) < DEN(P_2) < DEN(P_3) < DEN(P_4)$.

$MEPA$ **Routing Table Entry Format**							
Dest id	Seq Number	Valid Seq Number Flag	Other Flags	DEN	Next Hop	Precursors	Life Time

MEPA_REQ						
Type	MEPA_REQ id	Dest id	Dest Seq Number	Src id	Src Seq Number	DEN

MEPA_REP					
Type	Dest id	Dest Seq Number	Src id	DEN	Life Time

.1em 2.0ex

```
Initialization: $DEN(s,s) \leftarrow \infty$, $DEN(s,u) \leftarrow 0$, $DEN(s,t) \leftarrow 0$
Phase-II: Establish a $MEPA$ route
6.1 Source node broadcasts a route request containing $DEN(s,s)$.
6.2 When an intermediate node $u$ receives a route request from another node say $v$ containing $DEN(s,v)$ it checks its
    routing table:
    if ( $DEN(s,v) > DEN(s,u)$) then
        $DEN(s,u) \leftarrow min\{DEN(s,v), \delta(u)\}$.
        $p(s,u) \leftarrow v$.
        it broadcasts the route request containing $DEN(s,u)$ further to its neighbors.
    else
        it discards the packet.
    end
6.3 When destination receives a route request from a node $v$ containing $DEN(s,v)$ it checks its routing table:
    if ( $DEN(s,v) > DEN(s,t)$) then
        $DEN(s,t) \leftarrow DEN(s,v)$.
        $p(s,t) \leftarrow v$.
        it sends route reply packet containing $DEN(s,t)$ to $v$.
    else
        It discards the packet.
    end
6.4 When an intermediate node receives a route reply packet containing $DEN(s,t)$ from a node say $u$ it checks its routing
    table:
    if (stored $DEN(s,t) <$ received $DEN(s,t)$ ) then
        it updates stored $DEN(s,t)$ with the received $DEN(s,t)$.
        it sets the next hop for $t$ to $u$.
        it forwards reply packet to next hop on reverse path.
    else
        It discards the packet.
    end
6.5 When source node receives a $MEPA\_REP$ packet it checks its routing table:
    if (stored $DEN(s,t) <$ received $DEN(s,t)$ ) then
        it updates stored $DEN(s,t)$ with the received $DEN(s,t)$.
        it sets the next hop for $t$ to $u$.
        it starts sending data packets on the discovered path.
    else
        It discards the packet.
    end
```

Algorithm 6: Phase-II of $MEPA$

In the next theorem, we show that as the updated $DENs$ propagate from s to t, the entire algorithm converges in $O(|P|)$ time, where $|P|$ is the length of the $MEPA$ route. Note that we do not need to know the length of the $MEPA$ path for the result to hold.

Theorem 7.1 *Suppose a $MEPA$ route is to be established between the nodes s and t. Then at any node u on the $MEPA$ route between s and t, final value of $DEN(s,u)$ will* *be set in l hopcounts, where l is the length of the $MEPA$ route from s to u. In particular t sets final value of its $DEN(s, t)$, in $|P|$ hopcounts where $|P|$ is the length of the $MEPA$ route.*

Proof Consider a $MEPA$ route $s, n_1, \dots n_l, \dots t$. Let $n_l \neq t$ be the l^{th} node on the $MEPA$ route. Then, n_l receives $MEPA\_REQ$ originated by s after l hopcounts via $MEPA$ route. In the worst case, n_l receives $l-1$ $MEPA\_REQs$ from other paths before getting $MEPA\_REQ$ from its precursor node on the $MEPA$ route. Thus, n_l updates $DEN(s, n_l)$ at most $(l - 1)$ times; once it gets a $MEPA\_REQ$ along its $MEPA$ route no further updation to $DEN(s, n_l)$ takes place in future. To see this consider See Figure 7.4 :

Let the part of the $MEPA$ route from s to n_l be denoted by P_1. That is the $MEPA$ route from s to t is $P_1\ n_l \dots t$. Suppose n_l receives a $MEPA\_REQ$ from a node v' along P_1 first and then it receives $MEPA\_REQ$ with higher DEN from a node v via another path P_2 in future. There are following cases:

case1(a): See Figure 7.6(a). $DEN(s, v') < \delta(n_l)$, $DEN(s, v) < \delta(n_l)$. Such cases cannot arise because in these cases $MEPA$ route from s to t would be $P_2 n_l \dots t$ and not $P_1 n_l \dots t$.

case1(b): See Figure 7.6(b). $DEN(s, v') < \delta(n_l)$, $DEN(s, v) > \delta(n_l)$. Such cases cannot arise because in these cases $MEPA$ route from s to t would be $P_2 n_l \dots t$ and not $P_1 n_l \dots t$.

case2: See Figure 7.6 (c). $DEN(s, v') > \delta(n_l)$, $DEN(s, v) > \delta(n_l)$. In this case n_l receives a higher $DEN(s, v)$ from v on P_2. Then, as per step 1 of Section 7.21, $DEN(s, n_l)$ is not updated as $min\{DEN(s, v), \delta(n_l)\} = \delta(n_l)$ and $DEN(s, n_l)$ is already set to $\delta(n_l)$. However $p(s, n_l)$ is updated in this case.

Thus, once n_l gets a $MEPA\_REQ$ along its $MEPA$ route, $DEN(s, n_l)$ is not updated in future and the final value of $DEN(s, n_l)$ is set in l hopcounts.

Step 1 of Figure 7.21 is not applicable when $n_l = t$ (instead, step 1 of Figure 7.22 is applicable) but these cases do not exist for t as in that case P_2 would be the $MEPA$ route instead of P_1. Thus, t also sets final value of its $DEN(s, t)$ in $|P|$ hopcounts where $|P|$ is the length of the $MEPA$ route. ■

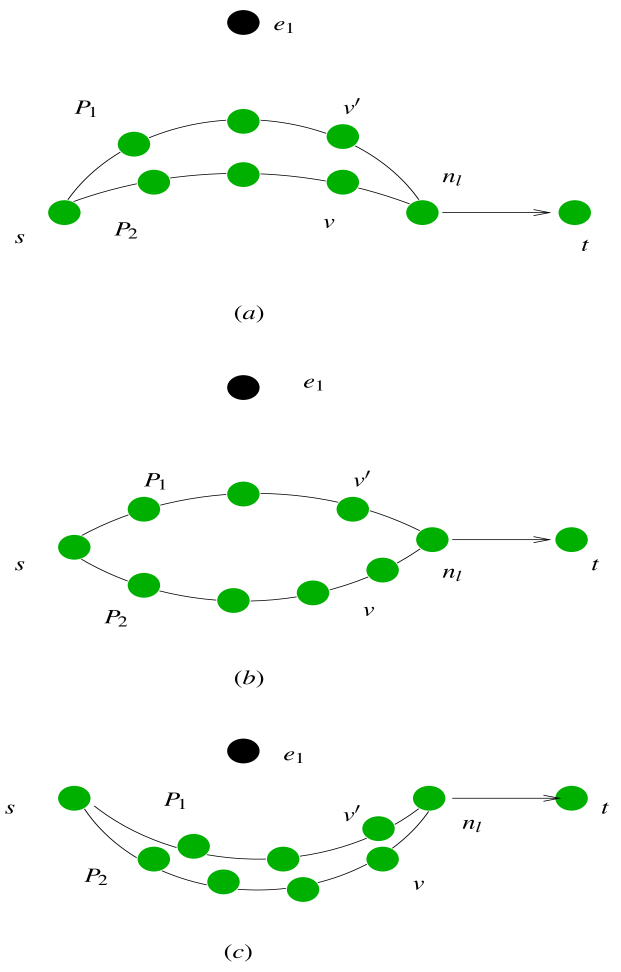

Figure 7.6: e_1 is an endangered node (a) $DEN(s, v') < \delta(n_l)$, $DEN(s, v) < \delta(n_l)$ (b) $DEN(s, v') < \delta(n_l)$, $DEN(s, v) > \delta(n_l)$ (c) $DEN(s, v') > \delta(n_l)$, $DEN(s, v) > \delta(n_l)$

Gujrat

Theorem 7.2 *In the absence of endangered nodes $MEPA$ route is the shortest route .*

Proof Let s be the source node that wants to establish $MEPA$ route to destination t. In the absence of endangered node, for any node u, $\delta(u) = \infty$. Let v be a node on $MEPA$ path from s to t then initially $DEN(s, v) = 0$. Initially s will broadcast $DEN(s, s) = \infty$ to its neighbors. For node $v_i \in Nb(s)$, $DEN(s, v_i) = min\{max\{DEN(s, s), DEN(s, v_i)\}, \delta(v_i)\} = min\{max\{\infty, 0\}, \infty\} = \infty$. Then for node $v \notin Nb(s)$ on $MEPA$ route, $DEN(s, v) = min\{max_{v_i \in Nb(v)}\{DEN(s, v_i)\}, \delta(v)\} = min\{max_{v_i \in Nb(v)}\{\infty, \infty\}\} = \infty$.

Thus the $MEPA\_REQ$ packets are treated the same way as $RREQ$ packets in

$AODV$ except that an intermediate node does not reply to a $MEPA$ route request. There is no effect of $\delta's$ and $DENs$ on the discovered route.

Note that for an intermediate node v, once $DEN(s, v)$ is set to ∞ it will never change. So v will broadcast $DEN(s, v) = \infty$ to it neighbors and so on. When destination receives $DEN(s, v)$ from any node v first time it will also set $DEN(s, t) = max_{v \in N(t)}\{DEN(s, v)\} = max_{v \in N(t)}\{\infty\} = \infty$. For destination also $DEN(s, t)$ is set to ∞ , it is never changed since it cannot receive DENs higher than ∞. So in the absence of endangered node, once $MEPA$ path is set it never changes. Assuming that the request which reaches the destination first follows the shortest path, $MEPA$ route is the shortest route.

■

Note: By keeping an additional hopcount field and storing the path with the smallest hopcount when $DENs$ are equal, we get $MEPA$ route which is shortest in terms of hopcount as well.

7.3.3 Maintenance of distances

As the nodes in an ad hoc network are highly mobile, distances must be updated as the nodes move or the links break or new links are established. Our algorithm updates the distances as described below:

Consider a case when a node moves out of the range of another node.

1. Let $u \in Nb(E)$. As long as, it is in the range of at least one endangered node, it does nothing. As soon as it stops hearing from all the endangered nodes (for example, when an endangered node has moved or u has moved), it switches off its in_nbhd flag. Now, distances of all those nodes v (including u), whose minimum distance from E was through u, need to be updated. This is done as follows: u broadcasts a request for 'distance from E' to its neighbors, which in turn pass on the request to their neighbors. The process continues till the request is received by some other neighbor of E.

   Now, consider a node v. It may receive the request from its $pr(v)$ or from a node which is not $pr(v)$. That is, it may receive the request from a node through which v had the shortest path to E or it may not be from such a node. (For example, in Figure 7.1, the node u_7 may receive the request from $pr(u_7)$ i.e. u_2 when both the nodes e_1 and e_3 have moved out of the range of u_2 or from u_6 when e_1 has moved out of the range of u_3). In the former case, $\delta(v)$ needs to be recomputed. So it resets $\delta(v)$ to ∞ on seeing the request and broadcasts it to its neighbors. In the latter case, $\delta(v)$ need not be recomputed (one can see that $\delta(u_7)$ is not affected in case e_1 has moved out of range of u_3) so, it simply broadcasts the request without resetting $\delta(v)$.

When other neighbors of E receive the request they do not broadcast it further and reply to the request by broadcasting their 'distance from E' . Distances of all the nodes are re-computed as it is done in phase-I. For example in Figure 7.1, in case u_2 receives the request of u_3 (when e_1 has moved out of the range of u_3) via u_7, it broadcasts its 'distance from E' ; it is received by u_7 and u_8. Next u_4 receives the distance broadcast from u_2 via u_8. Since $\delta(u_8) + 1 = 3 > 2 = \delta(u_4)$ (when u_4 received the request of u_3 from u_7 earlier then, since $u_7 \neq pr(u_4)$, $\delta(u_4)$ was not reset to ∞) therefore $\delta(u_4)$ does not change. Same is the case of u_7. But when u_6 receives $\delta(u_7) + 1$ then, since $\delta(u_6)$ was set to ∞, therefore it will be updated to 3. If we had set $\delta(u_4)$ and all nodes receiving the request of u_3 to ∞, $\delta(u_4)$ would have been updated to 3 which is wrong. Also, if $\delta(u_6)$ was not set to ∞ then it would not have been updated from 2 to 3.

2. Let u be a node not in $Nb(E)$ and $pr(u)$ moves out of the range of u or u moves out of range of $pr(u)$. In this case u will broadcast the request for 'distance from E' to its neighbors and the procedure explained above is repeated.

Next, consider a scenario in which a node u has moved into the range of another node v or v has moved into the range of u. If u is an endangered node, then v switches its $in\_nbhd(v)$ flag on, sets $\delta(v)$ to 1 and broadcasts its distance to its neighbors. The distances of all other nodes not in $Nb(E)$ are updated as explained earlier. If u is not an endangered node, its distance from E may have changed. Thus it broadcasts the request for 'distance from E' to its neighbors and the distances of all the nodes including u and v are updated according to the procedure explained above. A packet called 'Distance Discovery Unit' (DDU) is used to request for 'distance from E'. Algorithm 7 summarizes the maintenance phase of $MEPA$. It clearly takes $O(D)$ time for a request of 'distance from E' to reach a node in $Nb(E)$ and come back. The format of DDU packet is

Distance Discovery Unit(DDU)			
Type	DDU id	Src id	Source Seq Number

7.4 Simulation Study

We simulated our protocol using $NS2$. Performance of $MEPA$ was studied in the presence of multiple attacks (blackhole and wormhole attacks) and compared with $RDVBS$, $DENG$ and EEW. The proposed scheme was also compared with $AODV$ in the absence of endangered nodes.

7.4.1 Simulation Design

Simulation results were obtained for 50 nodes located over $1000m$ by $1000m$ region. The traffic sources are CBR (constant bit rate), 512-byte data packet, sending rate 1 pkt/sec and with maximum load of 300 packets for one transaction. The node movement speed is set from 0 to 80, which will be closer to real applications. The mobility are done with pause time 100 second. Script was executed for 300 seconds.

1em 2ex

```
Initialization: $in\_nbhd(u) \leftarrow 1$, $\delta(u) \leftarrow x$
Let $u$ be a node in the network.
7.1 if ( $u \in Nb(E)$) then
        if ( $u$ stops hearing from all the endangered nodes ) then
            $in\_nbhd(u) \leftarrow 0$.
            it broadcasts a $DDU$ packet.
        else
            it does nothing.
        end
    end
7.2 Suppose a node $u$ moves out of the range of another node $v$
    if ($pr(u)$ moves out of the range of $u$ or $u$ moves out of the range of $pr(u)$) then
        $u$ broadcasts a $DDU$ packet.
    end
7.3 Suppose a node $u$ moves into the range of another node $v$ or $v$ moves into the range of $u$
    if ( $u$ is an endangered node ) then
        $\delta(v) \leftarrow 1$.
        $v$ broadcasts $DU$ packet to its neighbors.
    else
        distance of $u$ from $E$ may have changed; it broadcasts a $DDU$ packet.
    end
7.4 When a node $v$ receives a $DDU$ packet
      if ($v \notin Nb(E)$) then
          if ( the $DDU$ packet is from pr(v)) then
              $\delta(v) \leftarrow \infty$.
              it broadcasts the $DDU$ packet to its neighbors.
          else
              It broadcasts the $DDU$ packet to its neighbors.
          end
      else
          It replies back with a $DU$ packet.
      end
7.5 When a node receives a $DU$ packet
      it updates its distance from $E$, if required, as it is done in phase-I and broadcasts the packet to its neighbors if the
      distance is updated.
```

Algorithm 7: Maintenance Phase of $MEPA$

7.4.2 Simulation Results

To study the performance of $MEPA$ in presence of attack, a node was selected randomly to behave like a blackhole node and a pair of nodes were selected randomly to implement a wormhole channel. The performance of $MEPA$ was compared with $RDVBS$, $DENG$ and EEW. It was observed that no packet was delivered in case of $RDVBS$ and

$DENG$ as a path through wormhole was established and packets were dropped there after. Similarly no packet was delivered in case of EEW as a path was established through blackhole node.

In the absence of mobility, packet delivery ratio of $MEPA$ was 1. $AEED$ is not defined (ND) for $DENG$, EEW and $RDVBS$ as no packet was delivered. These results are summarized in Table 7.1. Routing overhead of $MEPA$ is a little more initially when the number of connections is less as the advantage of pre-processing phase is less in that case. As the number of connections increase, the time taken in the path establishment phase starts dominating and $MEPA$ performs better than $DENG$ and $RDVBS$ in terms of routing overhead. Routing overhead of $MEPA$ is comparable with EEW. This is shown in Figure 7.7. Even in presence of mobility, $MEPA$ outperforms the other three algorithms as shown in Figures 7.8 and 7.9.

We also compared the performance of $MEPA$ with $DENG$ and $RDVBS$ in presence of **blackhole node only**. Packet delivery ratio and average end to end delay of $MEPA$ and $RDVBS$ were found to be better than that of $DENG$ (Figures 7.10 and 7.11) and routing overhead of $MEPA$ was better than that of $RDVBS$ and $DENG$ (Figure 7.12). Comparison of performance of $MEPA$ and EEW in presence of **wormhole node only** revealed that the packet delivery ratio and average end to end delay of $MEPA$ and EEW were comparable (Figures 7.13 and 7.14). Routing overhead of $MEPA$ was also comparable to EEW when the number of connections was more (Figure 7.15).

	PDR	AEED
$RDVBS$	0	ND
$MEPA$	1	.9731
$DENG$	0	ND
EEW	0	ND

Table 7.1: Comparison of $MEPA$ with other algorithms in the absence of mobility and in the presence of blackhole and wormhole nodes

We compared our protocol with $AODV$ in the absence of endangered nodes. When there are no endangered nodes, the pre processing phase and maintenance phases do not

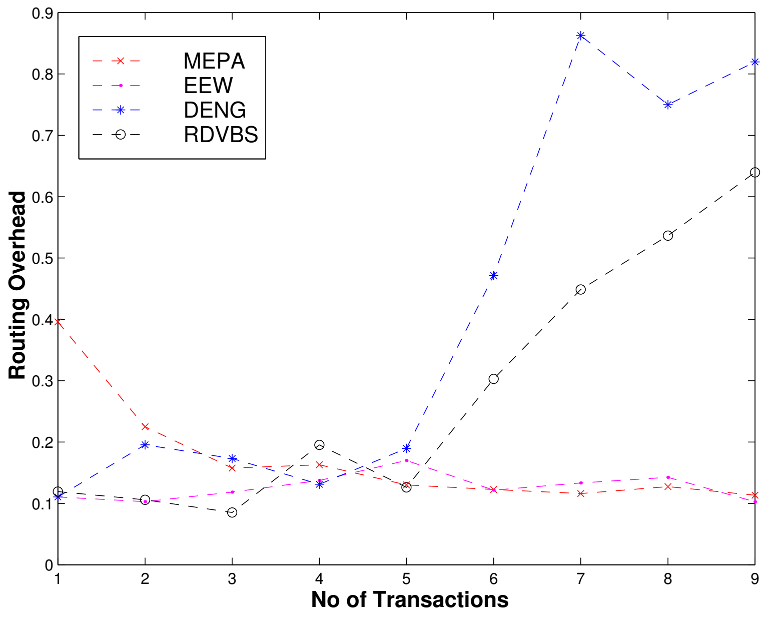

Figure 7.7: Comparison of Routing Overhead in presence of blackhole and wormhole nodes

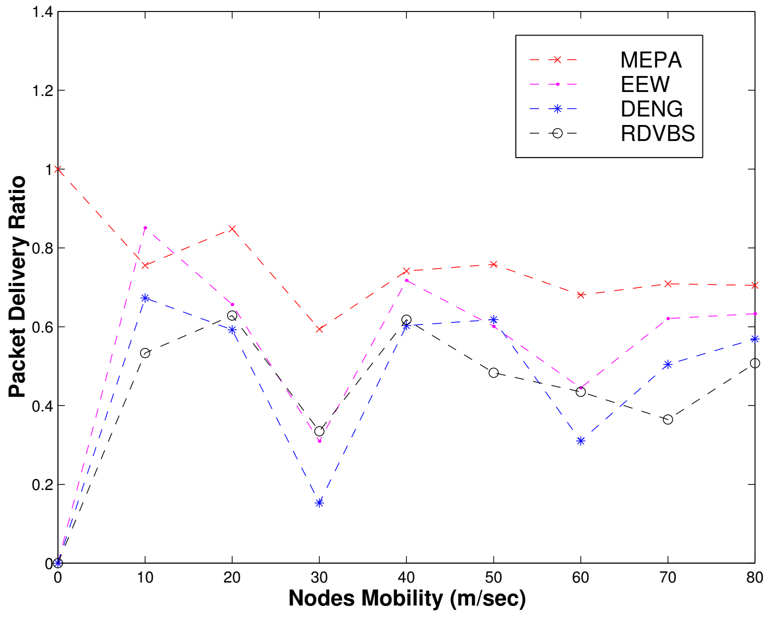

Figure 7.8: Comparison of Packet Delivery Ratio in presence of blackhole and wormhole nodes

come into effect and $MEPA$ reduces to route-discovery phase. Mobility in $MEPA$ is handled in a similar way as it is done in $AODV$. Thus our results (packet delivery ratio and average end to end delay) are comparable to those of $AODV$, see Figures 7.17 and 7.18. The routing overhead is also comparable except in few cases where an intermediate

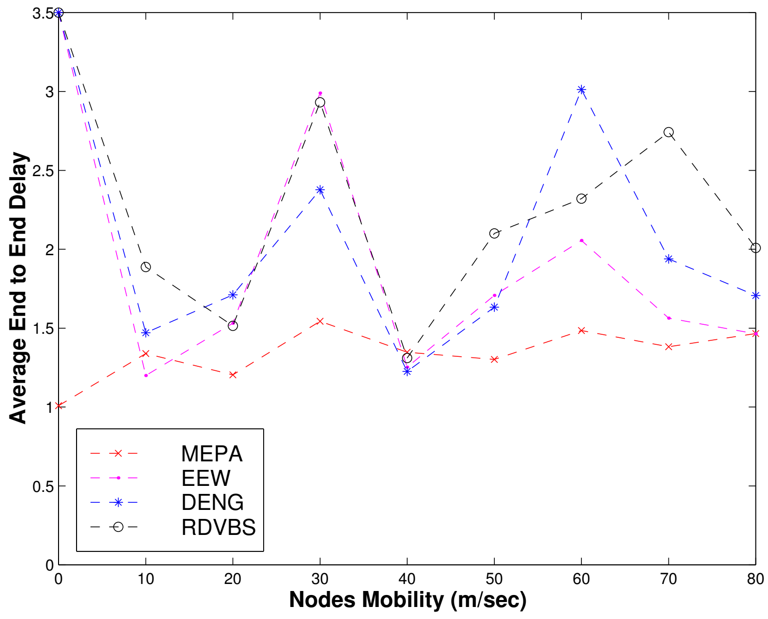

Figure 7.9: Comparison of Average End to End Delay in presence of blackhole and wormhole nodes

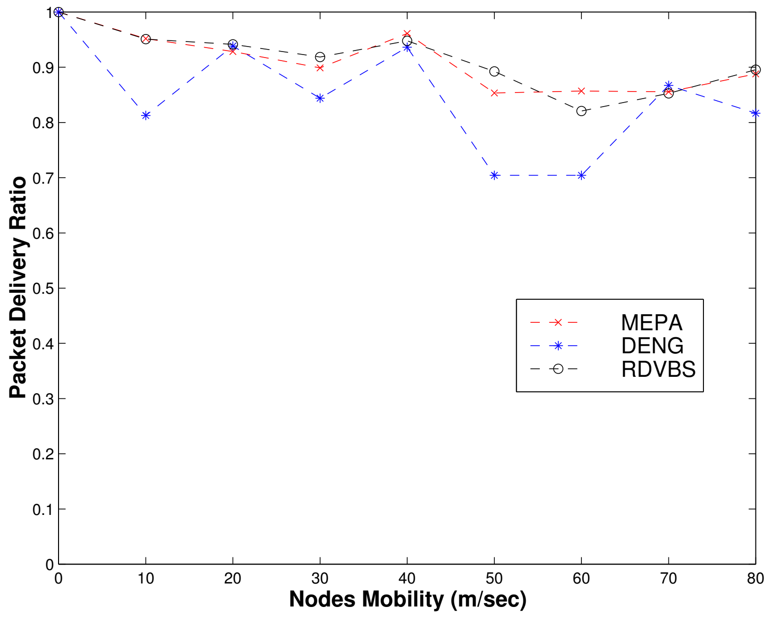

Figure 7.10: Comparison of Packet Delivery Ratio in presence of blackhole node only

node has a path in case of $AODV$. This is exhibited in Figure 7.19

To show that the $MEPA$ path is established in $O(|P|)$ time the algorithm was simulated for a network with 10% of nodes as endangered with various communicating pairs of varying length of $MEPA$ routes. Each simulation was run for 1000 seconds. It

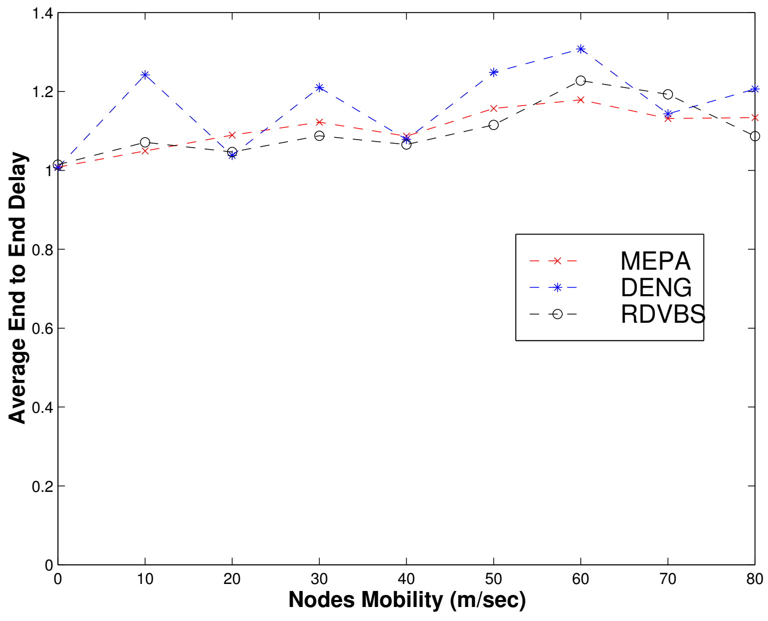

Figure 7.11: Comparison of Average End to End Delay in presence of blackhole node only

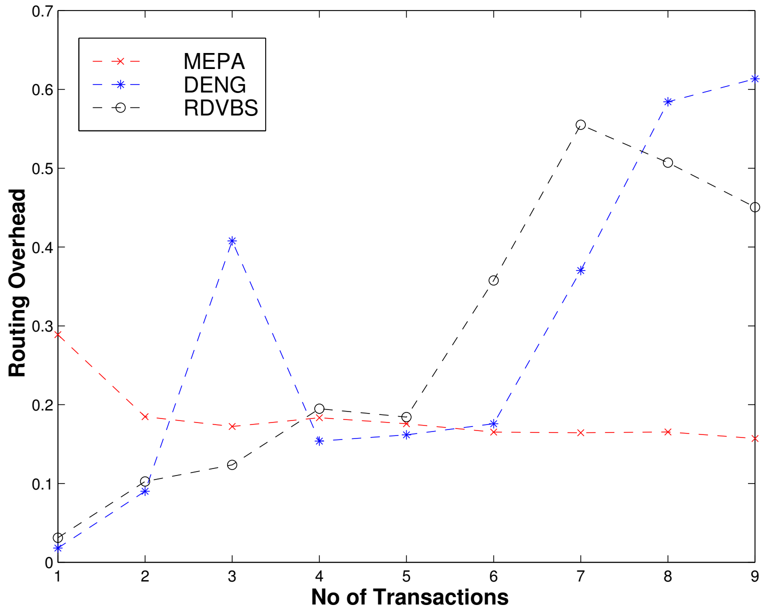

Figure 7.12: Comparison of Routing Overhead in presence of blackhole node only

was observed that the destination received multiple $MEPA\_REQ$ with $DENs$ at increasing time intervals. Destination sends $MEPA\_REP$ corresponding replies to each $MEPA\_REQ$. Whenever the destination received a $MEPA\_REQ$ with higher $DENs$, it discarded the previous route and replied to the new $MEPA\_REQ$ and a new route was

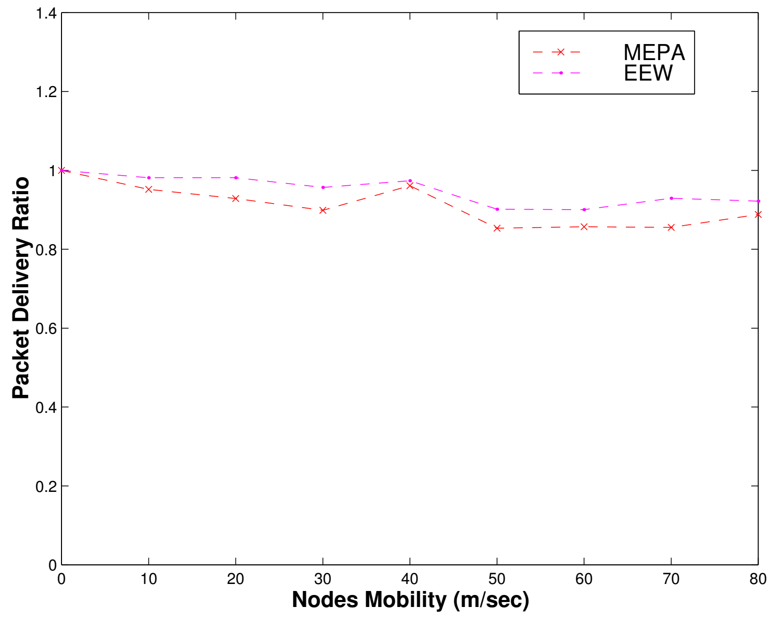

Figure 7.13: Comparison of Packet Delivery Ratio in presence of wormhole node only

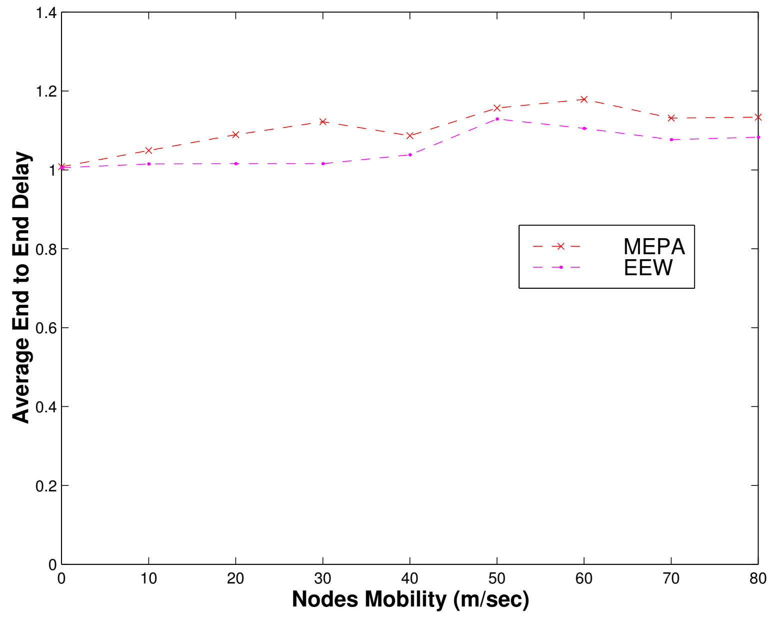

Figure 7.14: Comparison of Average End to End Delay in presence of wormhole node only

established. In Figure 7.16 the time taken to establish the $MEPA$ route is plotted against the length of the $MEPA$ route. Observe that the time to establish the $MEPA$ route is bounded by a linear function of the length of the route.

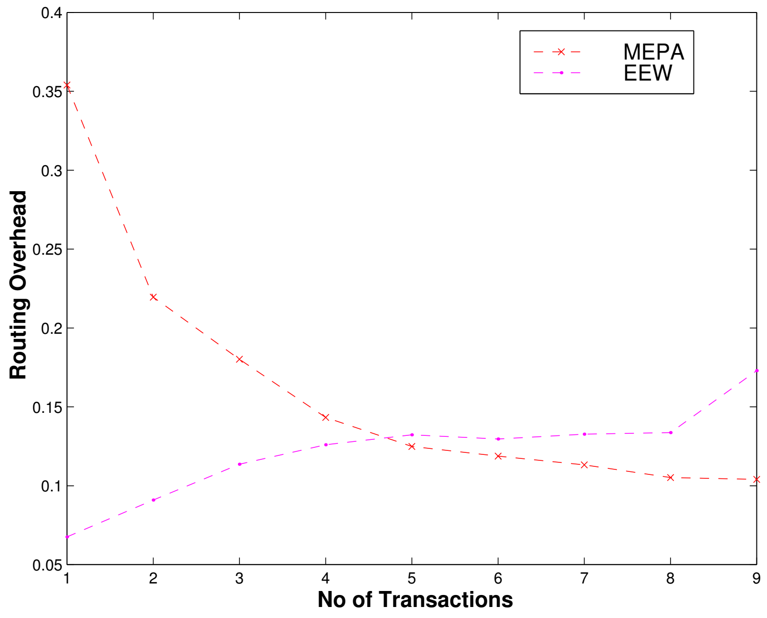

Figure 7.15: Comparison of Routing Overhead in presence of wormhole node only

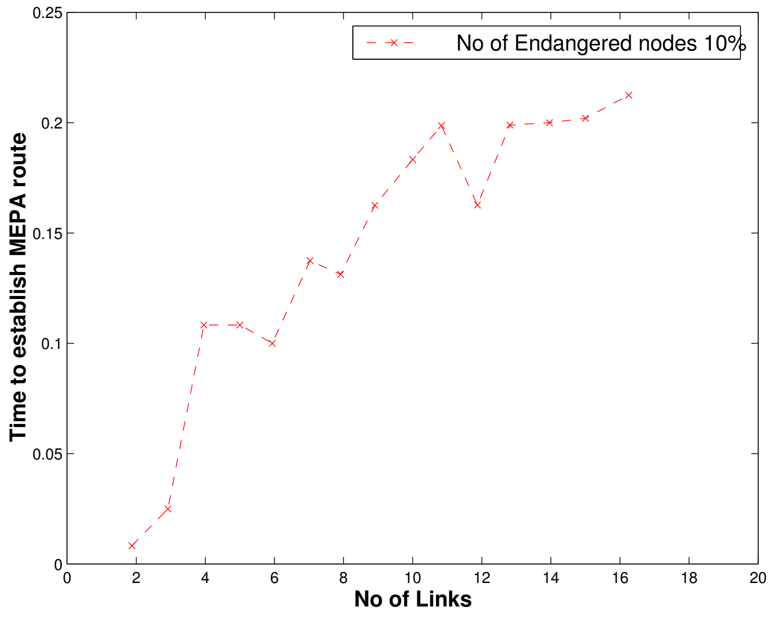

Figure 7.16: Time to establish a $MEPA$ route

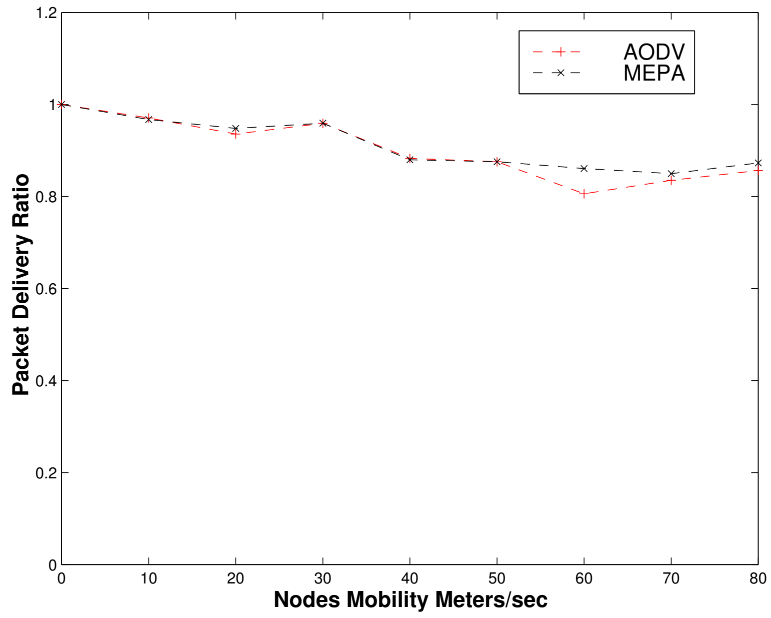

Figure 7.17: Comparison of Packet Delivery Ratio in absence of endangered node

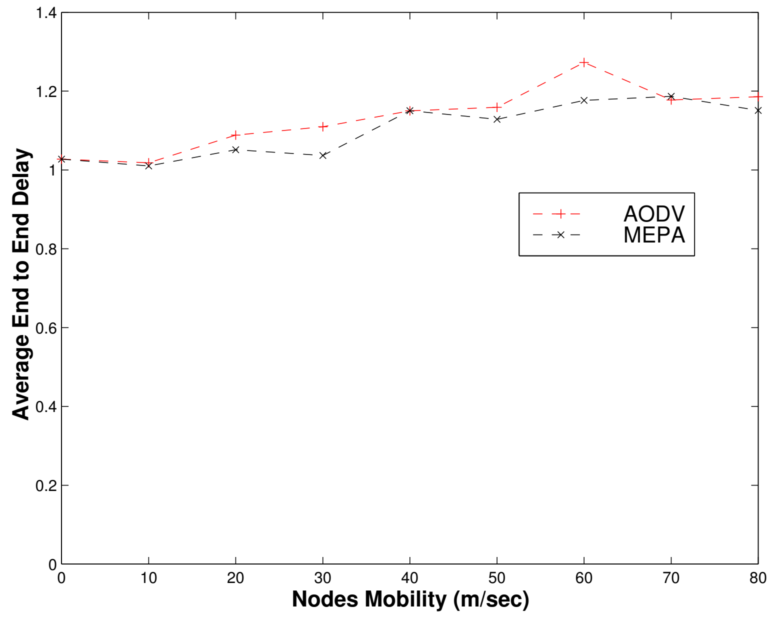

Figure 7.18: Comparison of Average End to End Delay in absence of endangered node

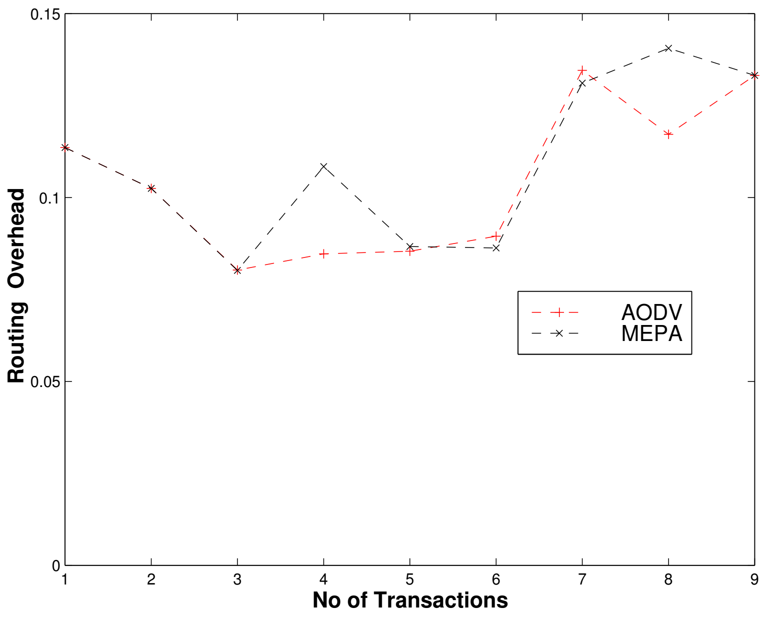

Figure 7.19: Comparison of Routing Overhead in absence of endangered node

When source s wants to communicate with node t, it creates a $MEPA\_REQ$ packet with $DEN(s,s)$ set to ∞ and broadcasts the packet to its neighbors.

Figure 7.20: Processing of $MEPA\_REQ$ at the source node in $MEPA$

For an intermediate node u, $DEN(s,u) = 0$. When u receives a $MEPA\_REQ$ packet from another node v, it compares the received $DEN(s,v)$ from the packet with its $DEN(s,u)$.

1. if $DEN(s,v) > DEN(s,u)$ then it sets $DEN(s,u) = min\{DEN(s,v), \delta(u)\}$, $p(s,u) = v$, replaces the DEN in $MEPA\_REQ$ packet with its own $DEN(s,u)$ and broadcasts it to its neighbors.

2. else it discards the packet.

Figure 7.21: Processing of $MEPA\_REQ$ at an intermediate node in $MEPA$

$DEN(s,t)$ is initialized to 0. When $MEPA\_REQ$ packet arrives at the destination from node v, it compares $DEN(s,v)$ with $DEN(s,t)$.

1. if $DEN(s,v) > DEN(s,t)$ then it sets $p(s,t) = v$, $DEN(s,t) = DEN(s,v)$ and sends $MEPA\_REP$ to v.

2. else it discards the packet.

Note that the destination may send multiple $MEPA\_REPs$ to the source if it receives $MEPA\_REQ$ packets with higher $DENs$ later.

Figure 7.22: Processing of $MEPA\_REQ$ at the destination in $MEPA$

Let v be an intermediate node that receives a $MEPA\_REP$ packet from a node u.

1. if v is receiving $MEPA\_REP$ packet for the first time, it sets $DEN(s,t)$ to the value it receives from the packet, sets the next hop for t to u and forwards $MEPA\_REP$ to $p(s,v)$ on the reverse path.

2. else (a node may receive multiple replies) v updates $DEN(s,t)$ and sets the next hop for t in its routing table if the received $DEN(s,t)$ is higher than the stored value and forwards $MEPA\_REP$ to $p(s,v)$ on the reverse path.

When source node s receives a $MEPA\_REP$ packet from a node u:

1. if s is receiving $MEPA\_REP$ packet for the first time, it sets $DEN(s,t)$ to the value it receives from $MEPA\_REP$, sets the next hop for t to u and starts sending the data packets on this path.

2. else (source node may also receive multiple replies) it updates its $DEN(\ s,t)$ and sets the next hop for t in its routing table if the received $DEN(s,t)$ is higher than the stored value, discards the previous path and starts sending the data packets on the new path.

Figure 7.23: Processing of $MEPA\_REP$ in $MEPA$

Chapter 8

Conclusion and Open problems

We presented several algorithms to mitigate different types of attacks on routing in ad hoc networks. The attacks include blackhole attack, wormhole attack and attack by selfish nodes. The existing approaches to assuage the impact of blackhole nodes do not handle selfish nodes and the ones to mitigate selfish nodes do not alleviate the blackhole nodes. Our algorithm to handle blackhole attack discovers secure path avoiding selfish nodes also. Moreover, it circumvents several collaborative/co-operating attackers. The algorithm outperforms the existing algorithms to mitigate blackhole nodes in terms of routing overhead without affecting other parameters like end-to-end delay and packet delivery ratio. The algorithm to mitigate wormhole attack is end-to-end in the sense that intermediate nodes carry out no verification and all the testing is done at the destination. It improves upon the existing end-to-end approach in terms of storage and computation overhead. It does not require clock synchronization nor does it require neighborhood monitoring in promiscuous mode. Since no approach is known to address all types of attacks, we present an algorithm to compute a path which is farthest from the endangered nodes. In this case, we assume that the set of endangered nodes is known which can be achieved by any intrusion detection mechanism. Another interesting version of the problem would be to compute a shortest path which is at least k hop away from the endangered nodes instead of computing a path which is farthest from the endangered nodes. We feel that it should be easy to modify scheme to compute such a path.

Our algorithm to mitigate blackhole and selfish nodes uses $AODV$ as an underlying routing protocol and the one for wormhole uses DSR. It would be interesting to merge and modify these algorithms to make them independent of the underlying routing protocol. Also, the algorithm handles collaborative attacks of a single type. We feel that since the motive of selfish nodes is not to cripple the network, they do not collaborate with the blacknodes. Hence, in our work we have talked only about collaboration amongst the blacknodes. It is a major challenge to devise a scheme to mitigate collaborative attacks by different types of attackers. The impact of such collaborative attacks would be much more devastating.

We obtained a theoretical bound on the length of the wormhole tunnel for which the wormhole is detected by the protocol. Through extensive simulation we showed that the protocol could detect even shorter wormhole. It would be interesting to prove better theoretical lower bounds for the tunnel length. One trivial approach to achieve this is to relax the bound on the hop-count. For example, if we discard the path if the hop-count is less than $\lceil 2d/r_{max} \rceil$ instead of $\lceil d/r_{max} \rceil$ we will be able to identify worm holes of shorter length. But this introduces a number of false positives. The real challenge would be to reduce the length of the tunnel without discarding too many good paths.

Bibliography

[ABV06] G. Acs, L. Buttyan, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. In *IEEE Transactions on Mobile Computing*, volume 5, pages 1533–1546, 2006.

[AGD08] P. Agrawal, R. K. Ghosh, and S. K. Das. Cooperative black and gray hole attacks in mobile ad hoc networks. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 310–314, Korea, 2008.

[AJ04] I. Aad and J.P.Haubax. Denial of service resilience in ad hoc networks. In *Proceedings of ACM Mobicom*, pages 202–215, Philadelphia PA, 2004.

[AK96] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, 1996.

[AK97] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *IWSP: International Workshop on Security Protocols*, pages 125–136, Paris, France, April 1997.

[Ban08] S. Banerjee. Detection/removal of cooperative black and gray hole attack in mobile ad-hoc networks. In *Proceedings of the World Congress on Engineering and Computer Science (WCECS)*, San Francisco, USA, 22 - 24 October 2008.

[BAS06] C. Buragohain, D. Agrawal, and S. Suri. Distributed navigation algorithms for sensor networks. In *Proceedings of the IEEE INFOCOM*, 2006.

[BB02a] S. Buchegger and J.-Y. Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403–410, IEEE Computer Society, Canary Islands, Spain, 2002.

[BB02b] S. Buchegger and J.-Y. Le Boudec. Performance analysis of the CONFIDANT protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks). In *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 226–236, Lausanne, Switzerland, 9-11 June 2002.

[BB03] S. Bansal and M. Baker. Observation-based cooperation enforcement in ad hoc networks. In *Research Report cs.NI/0307012 Stanford University*, 2003.

[BBO03] S. Bouam and J. Ben-Othman. Data security in ad hoc networks using multipath routing. In *Proceedings of 14th IEEE Personal, Indoor and Mobile Radio Communications(PIMRC)*, pages 1331–1335, 2003.

[BC93] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *Theory and Application of Cryptographic Techniques*, pages 344–359, 1993.

[BdOZI09] B. Bhargava, R. de Oliveira, Y. Zhang, and N. C. Idika. Addressing collaborative attacks and defense in ad hoc wireless networks. In *Second Workshop on Specialized Ad Hoc Network Systems (SAHNS), in conjunction with the ICDCS*, 2009.

[BH00] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *First ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA, August 2000.

[BH01] L. Buttyan and J. P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in selforganized ad hoc networks. In *Technical Report DSC/2001/001, Swiss Federal Institute of Technology – Lausanne*, 2001.

[BH03] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks ACM*, 2003.

[Bha02] B. Bhargava. Intruder identification in ad hoc networks. In *CERIAS Security Center and Department of Computer Sciencies*. Purdue University, research proposal, 2002.

[BOT99] B. Bellur, R.G. Ogier, and F.L. Templin. Topology dissemination based on reverse path forwarding(TBRPF). In *Proceedings of IEEE INFOCOM*, pages 178–186, 1999.

[BR02] E. M. Belding-Royer. Hierarchical routing in ad hoc mobile network. *Journal of Wireless Communication and Mobile Computing*, pages 515–532, 2002.

[Bur03] A. Burg. Ad hoc network specific attacks. Master's thesis, Technische Universitt Mnchen, 2003.

[CBH03] S. Capkun, L. Buttyan, and J. P. Hubaux. Sector: secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 21–32, Fairfax, Virginia, 2003.

[Ch.94] Ch. E. Perkins. *Ad Hoc Networking*. Addison Wesley, 1994.

[CL06] H. S. Chiu and K.-S. Lui. Delphi: wormhole detection mechanism for ad hoc wireless networks. In *Proceedings of the First International Symposium on Wireless Pervasive Computing*, 2006.

[CPS03] H. Chan, A. Perrig, and D. Song. Random key pre-distribution schemes for sensor networks. In *Proceedings of IEEE Security and Privacy Symposim*, pages 197–204, May 2003.

[CRZ00] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of Intenational ACM conference on Measurement and Modeling of Computer System*, pages 01–12, 2000.

[DB05] S. Desilva and R. V. Boppana. Mitigating malicious control packet floods in ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference*, 2005.

[DDHV03] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of 10th ACM conference on Computer and communication security (CCS03)*, Washington D.C., USA, 27-30 October 2003.

[DLA02] H. Deng, W. Li, and D. P. Agrawal. Routing security in wireless ad hoc networks. In *IEEE Communications Magazine*, volume 40, pages 70–75, October 2002.

[DM78] Y. K. Dalal and R. Metcalfe. Reverse path forwarding of broadcast packets. In *Proceedings of Communications of the ACM*, volume 21, pages 1040–1048, 1978.

[Dou02] J. R. Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, March 2002.

[GK08] N. Gupta and S. Khurana. SEEEP: Simple and efficient end-to-end protocol to secure ad hoc networks against wormhole attacks. In *Proceedings of the IEEE International Conference on Wireless and Mobile Communications*, pages 13–18, August 2008.

[GL01] M. Gerla and S.-Ju Lee. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of ICC*, volume 10, pages 3201–3205, June 2001.

[GS04] S. Ganeriwal and M. Srivastava. Reputation-based framework for high integrity sensor networks. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SAN' 04)*, pages 66–77, 2004.

[HCF$^+$01] L. Henrique, M. K. Costa1, S. Fdida1, O. Carlos, and M. B. Duarte. Hop by hop multicast routing protocol. In *Proceedings of ACM USENIX Security System*, pages 249–259, 2001.

[HCJ07] Y.T. Hou, C.M. Chen, and B. Jeng. Distributed detection of wormholes and critical links in wireless sensor networks. In *Proceedings of IIHMSP*, 2007.

[HE04] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2004.

[HFLY03] Y. A. Huang, W. Fan, W. Lee, and P. S. Yu. Crossfeature analysis for detecting ad-hoc routing anomalies. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS' 03)*, pages 478–487, 2003.

[HJP02] Y. C. Hu, D. B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 3–13, Calicoon, NY, 2002.

[HL03] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc networks. In *Proceedings of the First ACM Workshop Security of Ad Hoc and Sensor Networks*, Fairfax, Virginia, 13 October 2003.

[HL04] Y. A. Huang and W. Lee. Attack analysis and detection for ad hoc routing protocols. In *The 7th International Symposium on Recent Advances in Intrusion Detection (RAID' 04)*, pages 125–145, 2004.

[HPJ02] Y. C. Hu, A. Perrig, and D. B. Johnson. Ariadne :a secure on-demand routing protocol for ad hoc networks. In *proceedings of IEEE MOBICOM*, 2002.

[HPJ03a] Y. Hu, A. Perrig, and D. B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of IEEE INFOCOM*, 2003.

[HPJ03b] Y. C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of ACM WiSe*, San Diego, California, USA, 2003.

[HRS05] H. Huang, A. W. Richa, and M. Segal. Dynamic coverage in ad-hoc sensor networks. In *Mobile Networks and Applications*, pages 9–17, 2005.

[HSSS04] M. Hollick, J. Schmitt, C. Seipl, and R. Steinmetz. On the effect of node misbehavior in ad hoc networks. In *Proceedings of IEEE International Conference on Communications*, volume 6, pages 3759–3763, Paris, France, June 2004.

[HWK04] Q. He, D. Wu, and P. Khosla. SORI: A secure and objective reputation-based incentive scheme for ad hoc networks. In *Proceeding of IEEE Wireless Communications and Networking Conference*, Atlanta, GA, USA, 2004.

[JHB03] M. Jakobsson, J. Hubaux, and L. Buttyan. A micro-payment scheme encouraging collaboration in multi-hop cellular networks. In *Proceedings of Financial Crypto*, Gosier, Guadeloupe, 2003.

[JHC$^+$01] P. Jacquet, P. M. Hlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing for mobile ad hoc networks. In *Proceedings of IEEE INMIC*, 2001.

[JM96] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[JWY06] M. Jakobsson, S. Wetzel, and B. Yener. Stealth attacks on ad-hoc wireless networks. In *Proceedings of Vehicular Technology Conference*, 2006.

[KBB97] H. Krawczyk, M. Bellar, and M. Bellar. HMAC: Keyed-hashing for message authentication. In *RFC 2104*, Reston, Virginia, February 1997.

[KBS05] I. Khalil, S. Bagchi, and N. B. Shroff. LITEWORP: A lightweight countermeasure for the wormhole attack in multihop wireless networks. In *Proceedings of International Conference on Dependable Systems and Networks (DSN)*, 2005.

[KBS08] I. Khalil, S. Bagchi, and N. B. Shroff. MOBIWORP: Mitigation of the wormhole attack in mobile multihop wireless networks. *Ad Hoc Networks*, 6:344–362, 2008.

[KG08] S. Khurana and N. Gupta. FEEPVR: First end-to-end protocol to secure ad hoc networks of variable range against wormhole attacks. In *Proceedings of International IEEE Conference on Emerging Security Information, Systems and Technologies*, pages 13–18, 25-31 August 2008.

[KG10] S. Khurana and N. Gupta. End-to-end protocol to secure ad hoc networks against wormhole attacks. *Wiley Journal of Security and Communication Networks*, 4:10.1002/sec.272, 2010.

[KGA06] S. Khurana, N. Gupta, and N. Aneja. Reliable ad-hoc on-demand distance vector routing protocol. In *Proceeding of the IEEE International Conference on Networking(ICN)*, pages 98–103, 23-28 April 2006.

[KK77] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks performance evaluation and optimization. *Journal of Computer Networks*, 1:155–174, 1977.

[KKSW04] F. Kargl, A. Klenk, S. Schlott, and M. Weber. Advanced detection of selfish or malicious nodes in ad hoc networks. In *Proceedings of 1st European Workshop on Security in Ad-Hoc and Sensor Networks*, Germany, 2004.

[KNK$^+$07] S. Kurosawa, H. Nakayama, N. Kato, A. Jamalipour, and Y. Nemoto. Cross-feature analysis for detecting ad-hoc routing anomalies. *International Journal of Network Security*, 5:338–346, 2007.

[Kri11] A. W. Krisch. Heading towards 50 billion connections. 11 February 2011. http://www.ericsson.com/campaign/opportunitysupportsystems/newsfeed/posts/15/.

[KW03] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *International IEEE Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2003.

[L. 07] L. Buttyan and J. P. Hubaux. *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*. Cambridge University Press, 2007.

[LDR03] Q. Li, M. DeRosa, and D. Rus. Distributed algorithms for guiding across a sensor network. In *Proceedings of the IEEE MOBIOCOM*, 2003.

[LN03] D. Liu and P Ning. Establishing pair-wise keys in distributed sensor networks. In *Proceedings of 10th ACM conference on Computer and communication security (CCS03)*, USA, 27-30 October 2003.

[LS10] C. Lee and J. Suzuki. Swat: A decentralized self-healing mechanism for wormhole attacks in wireless sensor networks. In *Y. Xiao, H. Chen and F. Li (eds.) Handbook on Sensor Networks, World Scientific Publishing ISBN: 978-981-283-730-1*, 2010.

[LWF03] X.-Y. Li, P.-J. Wan, and O. Frieder. Coverage in wireless ad-hoc sensor networks. *IEEE Transactions on Computers*, 52:1–11, 2003.

[LWY09] Y. Liu, J. Wang, and Z. Yang. Sensor network navigation algorithms without locations. In *Proceedings of the IEEE INFOCOM*, 2009.

[LY02] Y. Liu and Y. R. Yang. Reputation propagation and agreement in mobile ad-hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, March 2002.

[MGD07] R. Maheshwari, J. Gao, and S. R Das. Detecting wormhole attacks in wireless networks using connectivity information. In *Proceedings of IEEE INFOCOM*, 2007.

[MJ08] S. Madria and J.Yin. SeRWA: A secure routing protocol against wormhole attacks in sensor networks ad hoc networks. *Journal of Ad hoc Networks*, 7:1051–1063, 2008.

[MKPS01] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava. Coverage problems in wireless ad hoc sensor networks. In *Proceedings of 6th IEEE World Multi-Conference on Systemics, Cybernetics and Informatics (SCI)*, pages 1380–1387, Orlando, FL, 2001.

[MLL03] D. P. Mehta, M. A. Lopez, and L. Lin. Optimal coverage paths in ad-hoc sensor networks. In *Proceedings of ICC*, volume 1, pages 507–511, 2003.

[MM00] S. Marti and A. Mishra. Mitigating routing misbehavior in mobile ad hoc networks. In *6th Int'l.Conference Mobile Comp. Network*, pages 255–265, 2000.

[MM02a] P. Michiardi and R. Molva. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the Sixth IFIP conference on security communications and multimedia*, Portoroz, Slovenia, 2002.

[MM02b] P. Michiardi and R. Molva. Simulation-based analysis of security exposures in mobile ad hoc networks. In *Proceedings of European Wireless Conference*, 2002.

[NAT06] F. Nait-Abdesselam and T. Tarik. Detecting and avoiding wormhole attacks in wireless ad hoc networks. In *IEEE Communications Magazine*, volume 46, pages 127–133, April 2006.

[NS2] The network simulator - ns2. http://nsnam.isi.edu/nsnam/index.php/User/_Information.

[NSSP04] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *Proceedings of Intl Symp on Information Processing in Sensor Networks*, 2004.

[PB94] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, August 1994.

[PE09] M. Parsons and P. Ebinger. Performance evaluation of the impact of attacks on mobile ad hoc networks. In *Proceedings of Field Failure Data Analysis Workshop*, USA, 27-30 September 2009.

[Per88] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, MIT LCS TR, October 1988.

[PH02] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of CNDS*, 2002.

[PL07] R. Poovendran and L. Lazos. A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks. *ACM Journal on Wireless Networks (WINET)*, 13:27 – 59, 2007.

[PM03] A. Patcha and A. Mishra. Collaborative security architecture for black hole attack prevention in mobile ad hoc networks. In *Proceedings Radio and Wireless Conference RAWCON*, 2003.

[PM08] A. A. Pirzada and C. McDonald. Detecting and evading wormholes in mobile ad-hoc wireless networks. *International Journal of Network Security*, 3:191–202, September 2008.

[PRD03] C. E. Perkins, E. M. Belding Royer, and S. R. Das. Ad-hoc on-demand distance vector (AODV) routing. In *Mobile Ad-hoc Networking Working Group*. Internet Draft, February 2003.

[PSL06] C. Piro, C. Shields, and B. Levine. Detecting the sybil attack in mobile ad hoc networks. In *Proceedings of IEEE Intl Conference on Security and Privacy in Communication Networks (SecureComm)*, 2006.

[PW02] K. Paul and D. Westhoff. Context aware inferencing to rate a selfish node in dsr-based ad hoc networks. In *Proceedings of the IEEE Globecom Conference*. Taipeh,, Taiwan, 2002.

[PYY+06] Y. Ping, H. Yafei, B. Yiping, Z. Shiyong, and D. Zhoulin. Flooding attacks and defence in ad hoc networks. *Journal of Systems Engineering and Electronics*, 17:410–416, 2006.

[QSL07] L. Qian, N. Song, and X. Li. Detection of wormhole attacks in multi-path routed wireless ad hoc networks: a statistical analysis approach. *Journal of Network and Computer Applications* , 30:308–330, 2007.

[R.E57] R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[RFdAG08] J. C. Ruiz, J. Friginal, D. de Andrs, and P. Gil. Black hole attack injection in ad hoc networks. 2008. http://www.zdnetasia.com/whitepaper/black-hole-attack-injection-in-ad-hoc-networks_wp-2378353.htm.

[RFN05] S. Ramaswamy, H. Fu, and K. Nygard. Effect of cooperative black hole attack on mobile ad hoc networks. In *Proceedings of ICWN*, 2005.

[RR02] R. Ramanathan and J. Redi. A brief overview of ad hoc networks: Challenges and directions. In *IEEE Communications Magazine*, pages 20–22, 2002.

[RSDE05] M. T. Refaei, V. Srivastava, L. DaSilva, and M. Eltoweissy. A reputation-based mechanism for isolating selfish nodes in ad hoc networks. In *Proceedings of Second IEEE Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS)*, pages 3–11, San Diego, CA, 2005.

[SA99] F. Stajano and R. Anderson. The resurrecting duckling. In *Security Issues for Ad-hoc Wireless Networks*, 1999.

[SB07] X. Su and R. V. Boppana. On mitigating in-band wormhole attacks in mobile adhoc networks. In *International Conference on Communications*, pages 1136–1141, 2007.

[SDL$^+$02] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. ARAN: A secure routing protocol for adhoc networks. In *UMass Tech Report*, pages 02–32, 2002.

[SGCW03] B. Sun, Y. Guan, J. Chen, and U. W.Pooch. Detecting black-hole attack in mobile ad hoc networks. In *Proceedings of the 5th European Personal Mobile Communications Conference*, pages 490–495, 2003.

[SSW03] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 1–10, San Diego, USA, 2003.

[STW06] A. Srinivasan, J. Teitelbaum, , and J. Wu. DRBTS: Distributed reputation-based beacon trust system. In *Proceedings of the 2nd IEEE International Symposium on Dependable*, Indianapolis, USA, 2006.

[SWLK03] J. H. Song, V. Wong, V. Leung, and Y. Kawamoto. Secure routing with temper resistant module for mobile ad hoc network. In *Proceedings of MobiHoc03 USA*, 2003.

[SYP04] M. A. Shurman, S. M. Yoo, , and S. Park. Black hole attack in wireless ad hoc networks. In *Proceedings of 42nd ACM Southeast Conference (ACMSE' 04)*, pages 96–97, 2004.

[THL$^+$07] P. V. Tran, L. X. Hung, Y.-K. Lee, S. Lee, and H. Lee. Ttm: An efficient mechanism to detect wormhole attacks in wireless ad-hoc networks. In *Proceedings of IEEE CCNC*, 2007.

[TS07] L. Tamilselvan and V. Sankaranarayanan. Prevention of blackhole attack in manet. In *Proceedings of the 2nd IEEE International Conference on Wireless Broadband and Ultra Wideband Communications*, 2007.

[TS08] L. Tamilselvan and V. Sankaranarayanan. Prevention of co-operative black hole attack in manet. *Journal of Networks*, 3:13–18, 2008.

[VKSM08] H. Vu, A. Kulkarni, K. Sarac, and N. Mittal. Wormeros: A new framework for defending against wormhole attacks on wireless ad hoc networks. In *Proceedings of the Third IEEE International Conference on Wireless Algorithms Systems, and Applications(INFOCOM)*, 2008.

[WBLW06] W. Wang, B. Bhargava, Y. Lu, and X. Wu. Defending against wormhole attacks in mobile ad hoc networks. *Wiley Journal Wireless Communications and Mobile Computing (WCMC)*, 6:483 –503, 2006.

[WLB03] W. Wang, Y. Lu, and B. K. Bhargava. On vulnerability and protection of ad hoc on-demand distance vector protocol. In *Proceedings of the 10th International Conference on Telecommunications (ICT' 03)*, volume 1, pages 375–382, 2003.

[WMKS05] E. Winjum, A. Marrie, Q. Kure, and P. Spiling. Replay attacks in mobile wireless ad hoc networks: Protecting the OLSR protocol. In *Proceedings of the International Conference of Networking, Lecture Notes in Computer Science*, volume 3421, pages 471–479, 2005.

[WSST05] B. Wang, S. Soltani, J. K. Shapiro, and P.-N. Tan. Local detection of selfish routing behavior in ad hoc networks. In *International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN)*, Las Vegas, 2005.

[WW07] X. Wang and J. Wong. An end-to-end detection of wormhole attack in wireless ad-hoc networks. In *Proceedings of International Conference on Computer Software and Applications*, 2007.

[YJWA02] Z. Yu, T. Jiang, X. Wu, and W. A. Arbaugh. Risk based probabilistic routing for ad-hoc networks. In *The 1st ACM Workshop on Wireless Security (WiSe)*, September 2002.

[YKL05] M. Yu, S. Kulkarni, and P. Lau. A new secure routing protocol to defend byzantine attacks for ad hoc networks. In *Proceedings of 13th IEEE International Conference on Network,*, volume 2, pages 1126–1131, 2005.

[YLY$^+$04] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: challenges and solutions. In *Proceedings of IEE Wireless Communication*, volume 11, pages 38– 47, UCLA, USA, 2004.

[YM06] J. Yin and S. Madria. A hierarchical secure routing protocol against black hole attacks in sensor networks. In *Proceedings of IEEE SUTC*, 2006.

[YNK02] S. Yi, P. Naldurg, and R. Kravets. A security-aware routing protocol for wireless ad hoc networks. In *Proceedings Of ACM Symposium On Mobile Ad hoc Networking & Computing (MOBIHOC)*, pages 286–292, 2002.

[YZV03] Z. Yan, P. Zhang, and T. Virtanen. Trust evaluation based security solution in ad hoc networks. In *Technical Report, Nokia Research Center*, Helsinki, Finland, October 2003.

[ZCY03] S. Zhong, J. Chen, and Y.R. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of IEEE INFOCOM*, 2003.

[ZDF06] L. Zhao and J. G. Delgado-Frias. Multipath routing based secure data transmission in ad hoc networks. In *Proceedings of Wireless and Mobile Computing, Networking and Communications (WiMobapos)*, pages 17–23, 2006.

[ZL00] Y. Zhang and W. Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking(MobiCom)*, 6-11 August 2000.

[ZLH03] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. In *Wireless Networks 9, Kluwer Academic Publishers*, pages 545–556, 2003.

[ZMB08] W. Znaidi, M. Minier, and J.-P. Babau. Detecting wormhole attacks in wireless networks using local neighborhood information. In *Proceedings of IEEE PIMRC*, 2008.

[ZS03] J. Zhen and S. Srinivas. Preventing replay attacks for secure routing in ad hoc networks. In *ADHOC-NOW LNCS 2865*, pages 140–150, Dalhousie University, Canada, 2003.

[ZXSJ03] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In *Proceedings of 11th IEEE International Conference on Network protocols (ICNP03)*, Atlanta, Georgia, 4-7 November 2003.

List of Publications

Journals

1. Sandhya Khurana, Neelima Gupta, ‘End-to-end protocol to secure ad hoc networks against wormhole attacks ’, Wiley Journal of Security AND Communication Networks, volume 4, issue 5, 2010.

2. Sandhya Khurana, Neelima Gupta, ‘Reliable Distance Vector routing protocol to handle Blackhole and Selfish (RDVBS) nodes in Ad hoc Network ’, Communicated.

3. Neelima Gupta, Sandhya Khurana, ‘Discovering Minimum Exposed Path to Attack $(MEPA)$ in Mobile Ad hoc Networks (MANETs) in optimal $O(|P|)$ time after pre-processing ’, Communicated .

Conferences

1. Sandhya Khurana, Neelima Gupta, ‘FEEPVR: First End-to-End Protocol to secure Ad Hoc Networks of Variable Range against Wormhole Attacks ’, The International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2008, August 25-31, 2008 - Cap Esterel, France, Page 74-79, IEEE.

2. Neelima Gupta, Sandhya Khurana, ‘SEEEP: Simple and Efficient End-to-End Protocol to secure Ad Hoc Networks against Wormhole Attacks’, The International Conference on Wireless and Mobile Communications, ICWMC 2008, July 27 - August 1, 2008 - Athens, Greece, Page 13-18, IEEE.

3. Sandhya Khurana, Neelima Gupta, Nagender Aneja, ‘Minimum Exposed Path to the Attack (MEPA) in Mobile Ad hoc Network (MANET)’, The International Conference on Networking, ICN 2007, April 22 - 28, 2007 - Sainte-Luce, Martinique, France, IEEE.

4. Sandhya Khurana, Neelima Gupta, Nagender Aneja, ‘Reliable Ad hoc On-demand Distance Vector Routing Protocol ’, The International Conference on Networking, ICN 2006, April 23-28, 2006 - Mauritius, IEEE.