DETECTION AND ISOLATION OF PACKET DROPPERS IN WIRELESS AD-HOC NETWORKS

by

Yu Zhang

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTERS OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2011

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____
Yu Zhang

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

_____ _____
Loukas Lazos
Assistant Professor

Date

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my academic advisor Dr. Loukas Lazos. Your passion for your work encouraged me to pursue my own goals, while your attention to details instilled in me a unique perspective on which to approach problems. I am deeply grateful for your taking me on as a student. Our academic work together has been a truly rewarding and enriching experience, while the lessons learned will help guide me throughout the course of my life in whatever avenue I pursue.

I would also like to thank the members of my defense committee, Dr. Marwan Krunz and Dr. Susan Lysecky, for both supporting my degree goals and the valuable time spent in the classroom.

Lastly I would like to thank my parents, you have provided everything that a son could every ask for, and so much more.

TABLE OF CONTENTS

LIST OF FIGURES 7

LIST OF TABLES 9

ABSTRACT 11

CHAPTER 1 Introduction 13
1.1 Our Contributions 15
1.2 Paper Organization 16

CHAPTER 2 Misbehavior in Wireless Ad Hoc Networks 17
2.1 Related Work on Packet Dropping 19
2.1.1 Credit-Based Systems 19
2.1.2 Reputation-Based Systems 21
2.1.3 Acknowledgment-Based Systems 24
2.2 Related Work on Route Discovery 26
2.2.1 Route Discovery in DSR 27
2.2.2 Route Discovery in AODV 28

CHAPTER 3 System Model and Problem Statement 29
3.1 Network Model 29
3.2 Adversarial Model 30
3.3 Problem Statement 31

CHAPTER 4 The AMD System 33
4.1 The AMD System Architecture 33
4.2 The Reputation Module 34
4.2.1 Computation of the Reputation Value 35
4.2.2 Collection of Reputation Information 38
4.3 The Route Discovery Module 40
4.3.1 DSR-based Route Discovery 42
4.3.2 AODV-based Route Discovery 47
4.3.3 Complexity Analysis of AMD Route Discovery 50
4.4 The Audit Module 51
4.4.1 Audit Phase 51
4.4.2 Search Phase 53

TABLE OF CONTENTS – *Continued*

4.4.3 Identification Phase 54
4.4.4 Multiple Misbehaving Nodes 55
4.4.5 Colluding Adversaries 58
4.4.6 Mobility 59

CHAPTER 5 Performance Evaluation 61
5.1 Simulation Setup 61
5.2 Performance Evaluation Metrics 61
5.3 Evaluation of the Evolution of Reputation Values 63
5.4 Evaluation of the Percentage of Dropped Packets 66
5.5 Evaluation of the AMD Overhead 66
5.5.1 Average Route Expansion Factor 66
5.5.2 Communication Overhead of the Route Discovery Module . . 68
5.5.3 Communication Overhead of the Audit Module 69
5.5.4 Identification Delay 71

CHAPTER 6 Conclusions 73

Bibliography 75

LIST OF FIGURES

1.1 An ad hoc network of 14 nodes. The disk of radius r represents the communication range of node S. The source S uses a multi-hop route to send data to the destination D. Node n_1 does not comply with the routing protocol specifications and drops packets destined to D. . . . 14

2.1 The route discovery process of DSR/AODV. (a) Controlled flooding of RREQ messages. (b) Unicasting of RREP messages. 27

3.1 Node S is sending traffic to D along P_{SD}. Node n_5 drops all packets. Our goal is to identify n_5, provide evidence of its misbehavior, and construct routes that avoid n_5. 31

4.1 The AMD system architecture. 33
4.2 An example of computing the path reputation of our DSR-based route discovery process. 41
4.3 Potential attack on the APR field. Node n_3 resets the APR field to 1 by replacing it with $E_{pk_D}(1)$ in the forward direction. 44
4.4 Example of DSR-based route discovery, for $\lambda = 1.2$ and $\gamma_1 = 0.6$. The destination rejects P_2 and P_4 because $r_{P_2} < \gamma_1$ and $|P_4| > \lambda|P_2|$. Path P_1 is preferred because $r_{P_1} > r_{P_3}$. 47
4.5 AODV-based route discovery. Three candidate paths are evaluated at D. Path P_2 containing the misbehaving node is rejected due to its low reputation value. 50
4.6 The search converges on link $n_3 - n_4$. S makes a slight alteration to P_{SD}, isolating n_3 and n_4, to determine that n_4 is the misbehaving node. 56
4.7 (a) n_5 misbehave while n_3 is audited. $\delta_3 \geq \gamma_0$. (b) Misbehavior strategies change (n_1 misbehaves, n_5 is honest). n_4 is audited and returns $\delta_4 < \gamma_0$. Search converged to $|\mathcal{V}| \leq 2$. 57
4.8 Path is partitioned into two suspicious sets; $\mathcal{V}_0^1 = \{n_1, n_2, n_3\}, \mathcal{V}_0^2 = \{n_4, n_5\}$. 57
4.9 (a) Nodes n_1, n_4 collude, with n_4 dropping all packets. Audited node n_2 claims misbehavior is downstream. (b) Nodes n_1, n_4 alter their behaviors, with n_1 dropping all packets. Audited node n_3 claims misbehavior is upstream. (c) Source simultaneously audits n_2, n_3 to verify if misbehaving link exists. 58

LIST OF FIGURES – *Continued*

5.1 (a) The average reputation value of behaving and misbehaving nodes for different α, β, as a function of the simulation epochs. (b) The average reputation value of nodes experiencing 30% packet loss due to poor channel conditions. 63
5.2 (a) The average reputation value of misbehaving nodes when such nodes drop a different percentage of their incoming traffic at random. (b) The average reputation of misbehaving nodes when they drop packets of type P_1 with different weight distribution. 64
5.3 (a) The percentage of dropped packets as a function of time for performance between AMD and DSR under different fractions of misbehaving nodes. (b) The percentage of dropped packets as a function of time when different fractions of transit traffic is randomly dropped. 65
5.4 (a) The average route expansion factor E as a function of the percentage of misbehaving nodes, for various values of the route selection factor λ. (b) The average route expansion factor E as a function of the nodes' communication range. 67
5.5 The average path reputation value of the paths discovered by AMD as a function of the route selection factor λ and for different number of misbehaving nodes. 68
5.6 The communication overhead of AMD normalized over the overhead of DSR/AODV as a function of the nodes' communication range, for various values of the route expansion factor. 69
5.7 (a) The communication overhead in number of transmitted messages, as a function of the percentage of misbehaving nodes. (b) The communication overhead as a function of audit size, a_{count}. 70
5.8 The identification delay for AMD, CONFIDANT, and 2ACK as a function of path length $|P_{SD}|$, in units of audits. 72

LIST OF TABLES

5.1 Simulation Parameters 62

ABSTRACT

Wireless ad hoc networks realize end-to-end communications in a cooperative manner. In this paradigm, multiple nodes coordinate to form a multi-hop route, when communication needs to take place between a source and a destination that are not within communication range. However, this model presupposes that intermediate nodes are willing to carry traffic other than their own. For ad hoc networks deployed in hostile environments, a protocol-compliant behavior on behalf of all nodes of the network cannot be assumed. Selfish and/or malicious users may misconfigure their devices to refuse forwarding any traffic, in order to conserve energy resources or degrade the network performance.

In this thesis, we address the problem of identifying and isolating misbehaving nodes that refuse to forward packets. We develop a comprehensive system called *Audit-based Misbehavior Detection* (AMD) that effectively and efficiently isolates both continuous and selective packet droppers. The AMD system integrates reputation management, trustworthy route discovery, and identification of misbehaving nodes based on behavioral audits. It consists of three modules: the reputation module, the route discovery module, and the audit module. All three modules are tightly integrated to ensure that multi-hop communications take place over paths free from malicious nodes. Compared to previous methods, our behavior monitoring process allows the evaluation of node behavior on a per-packet basis, without the need for energy-expensive overhearing techniques or intensive acknowledgment schemes. The behavior evaluations made by the audit module and processed by the reputation module, are exploited by the route discovery module in order to establish trustworthy paths and isolate packet droppers. We show via simulations that AMD successfully avoids misbehaving nodes, even when a large portion of the network refuses to forward packets, at the expense of an increased communication overhead during route discovery.

CHAPTER 1

Introduction

Wireless ad hoc network provide rapid on-demand network deployment without the need for the establishment of infrastructure. Nodes spontaneously self-organize into a network by coordinating network functions in a collaborative manner. Because of their infrastructure-less and autonomous nature, ad hoc networks find application on many domains including disaster relief operations, vehicular networks, tactical communications, environmental monitoring, and others [5].

In the absence of a supporting infrastructure, wireless ad hoc networks realize end-to-end communications in a cooperative manner. Nodes rely on the establishment of multi-hop routes to overcome the limitations of their finite communication range. In this paradigm, intermediate nodes are responsible for relaying packets from the source to the destination. As an example, consider Figure 1.1 depicting an ad hoc network of 14 nodes. A source S wants to establish communication with a destination D. S and D are unable to communicate directly due to their limited communication range. Thus, the source uses a multi-hop path to route data to the destination.

This communication model presupposes that intermediate nodes are willing to carry traffic other than their own. When ad hoc networks are deployed in hostile environments (tactical networks), or consist of nodes that belong to multiple independent entities, a protocol-compliant behavior on behalf of all nodes of the network cannot be assumed. Unattended devices can become compromised by the adversary and drop any transit traffic, in order to degrade the network service. Moreover, selfish users may misconfigure their devices to refuse forwarding traffic in order to conserve energy resources. This type of behavior is typically termed as *node misbehavior* [10, 9, 42].

In this article, *we address the problem of detecting and isolating misbehaving*

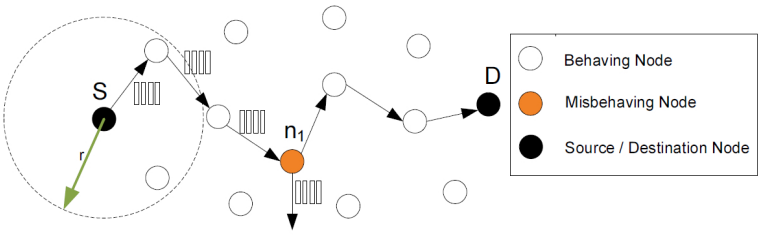

Figure 1.1: An ad hoc network of 14 nodes. The disk of radius r represents the communication range of node S. The source S uses a multi-hop route to send data to the destination D. Node n_1 does not comply with the routing protocol specifications and drops packets destined to D.

nodes in an effective and resource-efficient manner. In particular, we are concerned with packet droppers that either continuously drop packets routed through them, or selectively choose the set of packets to be dropped. Such misbehavior has been shown to have a severe impact on the network throughput [10, 9, 42, 43, 44].

Currently proposed solutions for identifying misbehaving nodes either use some form of per-packet evaluation of peer behavior [10, 9, 42, 43, 44, 11], or provide cooperation incentives to stimulate participation to packet forwarding [15, 14, 64, 33]. Incentive-based approaches do not address the case of malicious nodes who aim at disrupting the overall network operation. Malicious nodes are not interested in generating any legitimate traffic streams, and hence, they are not incentivized to collect credit. On the other hand, misbehavior identification approaches based on per-packet behavior evaluation techniques such as neighbor monitoring [11, 24, 44, 10, 9, 26, 45, 49, 55], or acknowledgment schemes [42, 3, 4, 47, 62], can prove to be too expensive in terms of communication overhead and/or overall energy expenditure.

Systems that employ packet overhearing techniques, require the operation of a large number of nodes in promiscuous mode for monitoring the behavior of their neighbors. Depending on the hardware design, packet reception can be almost as expensive as transmission [22]. Hence packet overhearing consumes significant energy resources. Moreover, operation in promiscuous mode contradicts energy conservation schemes that rely on sleeping patterns, such as those employed in Wireless

Sensor Networks (WSNs). Finally, packet overhearing techniques are complex to implement (if possible) in multi-channel networks where nodes are engaged in parallel transmissions over orthogonal frequency bands.

Acknowledgment-based systems, are based on the verification of packet forwarding from the intended receivers via the transmission of acknowledgments. Typically, every packet must be acknowledged two or several hops upstream, to ensure that intermediate nodes faithfully forward packets [42, 3, 4, 47, 62]. The use of per-packet acknowledgements that must travel multiple hops introduces significant communication and energy overhead in the process of misbehavior detection. To counter this overhead, we introduce a system called *Audit-based Misbehavior Detection* (AMD), which achieves per-packet behavior evaluation without incurring per-packet communication overhead. AMD is a comprehensive solution to misbehavior that encompasses identification of misbehaving nodes, reputation management, and trustworthy route discovery, in a distributed and resource-efficient manner. In particular, this thesis makes the following contributions.

1.1 Our Contributions

We develop the AMD system that integrates reputation management, trustworthy route discovery and identification of misbehaving nodes. AMD consists of three tightly coupled modules: a distributed reputation module responsible for managing the reputation of nodes participating in the network; a route discovery module responsible for discovering paths which exclude misbehaving nodes, according to the recommendations provided by the reputation module; an audit module responsible for identifying misbehaving nodes along the path from the source to the destination. We show that the route discovery module can construct paths with high reputation values, subject to any desired length constraint. When path contains misbehaving nodes, these nodes are efficiently located by the audit module. During the audit process, nodes along a path from a source to a destination are requested to provide proofs of their behavior. Using the behavioral proofs provided by honest nodes,

AMD identifies the misbehaving ones via a series of audits. The nature and order of these audits is dictated by a searching strategy supplemented by the knowledge of reputation values provided by the reputation module. We emphasize that proofs are constructed using Bloom filters which are storage-efficient membership structures [6], thus significantly reducing the communication overhead associated with misbehavior identification. Contrary to existing methods, our proposed audit scheme allows the per-packet evaluation of behavior, without incurring the per-packet overhead. Our extensive simulation results verify that AMD successfully isolates misbehaving nodes and practically eliminates packet dropping even if a large portion of the network (e.g., 50%) is misbehaving. This is achieved at significantly lower communication and energy cost compared to pre-existing methods.

1.2 Paper Organization

The remainder of the paper is organized as follows. In Chapter 2, we present related work. Chapter 3 describes our network and adversary model assumptions and states the problem addressed by this thesis. Chapter 4 describes the architecture and internal structure of the AMD system. Specifically, Sections 4.2, 4.3 and 4.4 present the reputation, route discovery and audit modules, respectively. In Chapter 5, we study the performance of AMD via extensive simulations and compare AMD with representative schemes from the different classes of misbehavior detection schemes. In Chapter 6, we summarize our findings.

CHAPTER 2

Misbehavior in Wireless Ad Hoc Networks

Ad hoc networks are based on the assumption that all nodes collaborate to realize network services. However, this implicit trust placed on the willingness of network nodes to collaborate is often breached. Nodes can act in a selfish manner, in order to conserve their own resources (such as energy), by refusing to relay traffic of other nodes. Moreover, malicious nodes can attempt to degrade network performance by dropping all packets routed through them. Regardless of motive, such misbehaving nodes violate the core ad hoc network principle of collaboration. Node misbehavior in the routing function has been shown to have a severe impact on the network throughput [9, 33, 42, 43, 44].

Several attacks have been demonstrated against routing protocols in ad hoc networks [9, 33, 18, 27, 30, 40, 42, 44, 46, 52, 53]. We first provide a brief description of those attacks. In a sinkhole attack, a misbehaving node attempts to attract traffic by falsely advertising a shortest route to multiple destinations [18, 46, 52]. Thus, neighboring nodes route their traffic through the misbehaving node, allowing it to drop/modify/analyze a large volume of packets. Networks in which all nodes periodically transmit data to a sink, such as a monitoring network, are particularly vulnerable to the sinkhole attack since data packets have a single destination.

In a blackhole attack [39, 61], the misbehaving node advertises the shortest path to a particular destination node whose traffic it wants to intercept. Once a route request is received by the misbehaving node, it immediately replies to the source. Since most route discovery processes accept the first route discovered, the traffic is routed through the misbehaving node.

In a wormhole attack [27, 40, 53], two (or more) nodes establish a low-latency link between distant parts of the network. Messages received at one end of the wormhole are transmitted back on the other end. Since the wormhole is a low-latency

link, nodes on either side of the wormhole will appear as neighbors, and eventually all packets destined from one side of the network to the other will traverse the wormhole. Notice that in this attack, the misbehaving nodes are not required to broadcast messages advertising the low-latency link. This occurs naturally during route discovery since all received packets are rebroadcasted through the wormhole. Once established, the misbehaving nodes can launch additional attacks on the network based on the large volume of traffic traversing the wormhole. Preventative measures against the wormhole attack are based on topological consistency checks [27, 29, 40].

On-demand routing protocols for ad hoc networks have been shown vulnerable to the rushing attack [30]. In this attack, a misbehaving node tampers with the route request packet and modifies the routing path. Once modified, the node rushes the packet to the next hop. Since under most protocols only the first copy of the route request is accepted (all subsequent ones are dropped), the misbehaving node attempts to forward the modified route request before any others are transmitted, thus causing the modified packet to be used for the duration of the route discovery and establish a false routing path.

The most common form of misbehavior is packet dropping [9, 33, 42, 44]. In this attack, the misbehaving node participates in the routing path establishment process. Once the routing path is established, the misbehaving node simply refuses to forward packets to the next hop. The source and destination can recognize that a performance drop has occurred on the routing path, but are unable to determine the problematic link. This thesis focuses on this last type of misbehavior in which the source and destination attempt to determine the node(s) that drop packets along the routing path. We now describe related work with respect to the misbehavior identification problem.

2.1 Related Work on Packet Dropping

Previously proposed methods for addressing packet droppers can be classified into, (a) credit-based systems [13, 15, 17, 58, 64], (b) reputation-based systems [9, 33, 14, 12, 23, 26, 36, 44, 45, 49, 55, 59], and (c) acknowledgment-based systems [3, 4, 42, 47, 62].

2.1.1 Credit-Based Systems

Credit-based systems [13, 15, 17, 58, 64] are designed to provide incentives for forwarding packets. Buttyan and Hubaux [13, 15] proposed a system in which nodes receive credit for each packet they forward, and spend their accumulated credit to transmit their own packets. This is accomplished through the use of a counter called the *nuglet counter*. The nuglet counter is incremented each time the node forwards a packet, and decremented each time the node transmits its own packet. The nuglet counter cannot take on a negative value and cannot be arbitrarily changed by the node. To enforce this rule, the nuglet counter is implemented in a tamper-proof hardware module, called the *security module*. The security module is assumed to provide universal protection from both software and physical attacks.

Zhong et al. [64] proposed Sprite, in which nodes collect *receipts* for the packets that they forward to other nodes. For a packet sent from a source to a destination, each node along the path records a hash of the packet as the receipt, and forwards the packet to its next hop. When the node has a high-speed link to a Credit Clearance Service (CCS), it uploads its receipts. The CCS determines the value of the receipts and provides credit in exchange. Credit is only granted if the destination reports a receipt verifying reception of the packet and if the node was on the routing path. Once verified, credit is removed from the sources account and given to each node who participated in packet forwarding. Thus nodes that transmit their own packets but do not cooperate in packet forwarding will incur a debt at the CSS. Debt accumulation beyond a certain threshold is interpreted as misbehavior.

Crowcroft et al. [17] proposed a scheme which not only rewards nodes for par-

ticipating in packet forwarding with credit, but takes into account congestion and traffic flow. When sending a packet, the source computes a congestion price, which is a metric defined by the required power for transmission and the available bandwidth. It then compares this price to its personal willingness-to-pay parameter, which the source continually adjusts based on its personal observations. By taking into consideration bandwidth in computing the cost (credit) required to send a message to the destination, the scheme avoids overwhelming low cost routes, as they would increase in costs as they become saturated. Power and bandwidth metrics are dynamically updated based on shared information among nodes.

Salem et al. [58] proposed a scheme to provide incentives to nodes in multihop cellular networks. The scheme relies on the fact that all network traffic must travel through the base stations (i.e. cell towers), and that all base stations are owned by a single trusted operator. When the source sends a packet, it appends a keyed hash of the entire packet. Each intermediate node re-hashes the entire packet, including the previously appended hash. The previous node's hash is then replaced with the new intermediate hash. Once at the base station, the hash is verified and the packet is transmitted over the backbone network, where it is re-transmitted to the destination from a nearby base station. The source is charged immediately by the base station upon receipt of a packet, while the destination is charged a small amount when the packet is re-transmitted. This amount is refunded once the destination acknowledges the reception of the packet, thus preventing the destination from cheating the system by claiming packets were never received.

While credit-based systems motivate selfish nodes to cooperate in packet forwarding, they provide no incentive to malicious nodes that target the network throughput. Such nodes have no incentive to collect credit and receive no punishment for non-cooperation. Furthermore, tamper-proof hardware [25] is currently too expensive to integrate in every network device, while providing an unverifiable level of security [2]. Sprite removes this requirement, at the expense of requiring the presence of a CCS. Lastly, credit-based systems lack a mechanism for identifying the misbehaving node(s), allowing them to remain within the network indefinitely.

Huang et al. [31] even go so far as to question whether or not there exists a need for credit-based schemes. Some of the issues they bring to light is that all network nodes are not treated fairly. Nodes located on the outskirts of the network will encounter less traffic to be forwarded, thus accumulating less credit compared to nodes located in the center of the network. Thus there exists an inherent disadvantage to a subset of nodes. They also claim that to be effective, a credit-based system must be uniquely designed for a given network, which conflicts with it idea of ad hoc networks and their corresponding technologies.

2.1.2 Reputation-Based Systems

Reputation-based systems [9, 33, 11, 12, 23, 26, 36, 44, 45, 49, 55, 59] use neighborhood monitoring techniques to identify misbehaving nodes. Marti et al. [44] proposed a scheme which relies on two modules, the *watchdog* and the *pathrater*. The watchdog module monitors the behavior of their next hop node by operating their radio in promiscuous mode. Once a node forwards a packet to the next hop, the node overhears to verify that the next hop node faithfully forwarded the packet. The scheme is based on the assumption that links between nodes are bi-directional and nodes utilize omni-directional antennas. A cache is used to store packets that wait for verification. If packets remain in the cache longer than a threshold period, the watchdog makes an accusation of misbehavior. The pathrater module uses the accusations generated to chose a path that will most likely avoid misbehaving nodes.

Buchegger and Le Boudec [9, 33, 11] proposed a scheme called *CONFIDANT*, which is built upon the watchdog/pathrater model. Nodes perform neighborhood monitoring using their radios in promiscuous mode while selecting paths that attempt to avoid misbehaving nodes. Whereas Marti et al. proposed using only the previous hop for monitoring, CONFIDANT requires all neighboring nodes to operate in promiscuous mode for monitoring, thus replying on a neighborhood watch. In addition, monitoring nodes notify other nodes of detected misbehavior through the broadcast of alarm messages. Instead of including a proof of the misbehavior in the alarm message, a scheme based on Pretty Good Privacy (PGP) [65] is implemented

to determine the trust level of the alarm message.

Soltanali et al. [59] propose a reputation-based scheme consisting of four modules: a Monitor, a Opinion Manager, a Reputation Manager, and a Routing/Forwarding Manager. The Monitor module monitors the nodes neighbors via the watchdog model, verifying that neighboring nodes faithfully participate in packet forwarding. Based on observations from the Monitor, the Opinion Manager formulates opinions of the nodes behavior and periodically advertises them to neighboring nodes. The Reputation Manager accepts these opinions and processes them to arrive at a trust metric for a specific node. When establishing a routing path to a destination, the Routing/Forwarding Manager uses these trust metrics to avoid including untrustworthy (misbehaving) nodes.

Ganeriwal and Srivastava [23] use a Bayesian model to map binary ratings to reputation metrics, using a beta probability density function. Each sensor computes a reputation ranking for its neighbors, defining them as cooperative or noncooperative. The ranking is based on multiple factors, including but not limited to routing consistency and packet integrity. Nodes can also share information regarding their classification of neighbors as cooperative/noncooperative. Jøsang and Ismail [36] presents similar work on how to derive reputation rankings using beta probability functions based on feedback of neighboring node behavior. Likewise, Buchegger and Le Boudec [12] investigate the effects of rumor spreading in ad hoc networks and propose a reputation-based scheme based on a Bayesian model. They also attempt to identify lies and exclude them as input to their reputation model.

He et al. [26] proposed SORI, which monitors neighboring nodes using a watchdog mechanism and propagates this information to nearby nodes, thus relying on both first- and second-hand information. Each node monitors all neighboring nodes, while maintaining a neighborhood list. The neighborhood list contains the number of packets each neighbor received and the number forwarded. Periodically, neighboring nodes exchange reputation information. This second-hand information is added to the nodes observations to compute an overall evaluation record for a node. SORI takes the additional step of punishing nodes deemed to be misbehaving. Neighbors

of a misbehaving node will probabilistically drop its packets, thus encouraging cooperation among nodes. SORI includes a mechanism to prevent retaliation attacks in which nodes continually increase the probability of dropping each others packets. The authors address security issues such as node impersonation by requiring the use of an authentication mechanism based on one-way hash chains.

Rebahi et al. [55] proposed a reputation-based scheme which also relies on first- and second-hand information. However the authors propose two different methods for nodes to acquire the second-hand information, i.e., the reputation information originating from neighboring nodes. In the first method, as soon as a node witnesses misbehavior, defined according to a threshold number of packet drops, the node immediately broadcasts the accusation. Thus the proactive transmitting of reputation information allows all nodes in the network to have up-to-date behavioral information about their neighbors. However, since the proactive broadcasting of information may require unacceptable bandwidth requirements, thus diminishing the networks functionality, nodes can also acquire second-hand information in an on demand manner. In much the same way that on demand routing protocols request route information, a node transmits a packet to the network requesting reputation information from other nodes. Thus network resources are only consumed to transfer reputation information that is requested.

Michiardi and Molva [45] proposed CORE, in which nodes create a composite reputation rating for a given node by combining the nodes subjective reputation, its indirect reputation and its functional reputation. The subjective reputation is calculated from direct observation of the nodes behavior, using a weighted average of both current and past observations. The indirect reputation is a value calculated based on second-hand observations made by other nodes in the network. A node's functional reputation is based on task-specific behavior. Thus it is computed based on its reputation in packet forwarding, routing, etc. Denial-of-service attacks based on misbehaving nodes broadcasting negative ratings for honest nodes are prevented by preventing nodes from broadcasting negative behavior. Thus when sharing reputation metrics, node are restricted to sharing only positive ratings.

Paul and Westhoff [49] proposed a scheme which can identify different types of misbehavior through routing message verification and packet comparisons. In particular, they focus on securing DSR to attacks, in which a misbehaving node either (a) refuse to forward route request packets, (b) forwards route requests without adding itself to the routing path, or (c) adds unrelated nodes to the route request. The scheme verifies routing messages through the use of an un-keyed hash chain, while nodes compare RREQ headers to a local cache consisting of headers from overheard packets to identify misbehavior. Each intermediate node along the path thus monitors its neighboring nodes, and send any accusations of misbehavior to the source, along with the type of misbehavior they witnessed. The source analyzes all accusations received, and takes action based on the type of misbehavior witnessed.

The process of node monitoring becomes complex in the case of multi-channel networks or nodes equipped with directional antennas. Neighboring nodes may be engaged in parallel transmissions in orthogonal channels or different sectors thus being unable to monitor their peers. Moreover, operating in promiscuous mode requires up to 0.5 times the amount of energy for transmitting a message [22], thus making message overhearing an energy expensive operation. Finally, reputation-based systems are proactive in nature, requiring the constant monitoring of nearby nodes. Hence, overhead is incurred on all nodes regardless of whether a misbehaving node exists in a neighborhood.

2.1.3 Acknowledgment-Based Systems

Acknowledgment-based systems [3, 4, 42, 47, 62] rely on the reception of acknowledgments to verify that a message was forwarded to the next hop. Balakrishnan et al. [4] proposed a scheme called TWOACK, where nodes explicitly send 2-hop acknowledgment messages (TWOACK) to verify cooperation. For every packet a node receives, it sends a TWOACK along the reverse path, verifying to the node 2-hops upstream that the intermediate node faithfully cooperated in packet forwarding. Packets that have not yet been verified remain in a cache until they expire. A value is assigned to the quantity/frequency of un-verified packets to determine

misbehavior.

Liu et al. [42] improved on TWOACK by proposing 2ACK. Similar to TWOACK, nodes explicitly send 2-hop acknowledgments (2ACK) to verify cooperation. To reduce overhead, 2ACK allows for only a percentage of packets received to be acknowledged. Additionally, 2ACK uses a one-way hash chain to allow nodes in the routing path to verify the origin of packets they are acknowledging, thus preventing attacks in which a misbehaving node drops the original packet and forwards a spoofed packet.

Padmanabhan and Simon [47] proposed a method called *secure traceroute* to identify the link on which misbehavior is occurring. Instead of the standard traceroute operation, which relies on nodes responding to expired packets, secure traceroute verifies the origin of responses and uses traceroute packets that are indistinguishable from data packets. Secure traceroute proceeds hop by hop, although instead of responding to expired packets, the source establishes a shared key with the node. By encrypting the packets, secure traceroute packets are indistinguishable from data packets and cannot be selectively dropped. A Message Authentication Code (MAC) is utilized for authenticating the packets origin. Although traceroute is considered a reactive approach, secure traceroute is proactive, requiring connected nodes to transmit "keep-alive" packets when they have data to send.

Xue and Nahrstedt [62] proposed the Best-effort Fault-Tolerant Routing (BFTR) scheme, which relies on end-to-end acknowledgment messages to monitor packet delivery ratio and select routing paths which avoid misbehaving nodes. Similar to the DSR routing protocol, the source floods RREQ messages to discover a routing path to a destination. However, RREP packets must be sent along the reverse path and must be signed with a shared secret key between the source and destination. Also, the destination responds to multiple RREQ, thus providing the source with multiple paths to choose from. The source selects the shortest path for packet routing. During transmission to the destination, the source monitors the feasibility of the routing path, based on the end-to-end acknowledgments sent by the destination. Using a proposed heuristic, the source varies the routing path to maintain feasibility. Thus,

the goal of BFTR is to avoid misbehaving nodes.

Awerbuch et al. [3] proposed an on demand routing protocol that probes the path to identify the faulty link. Once misbehavior is identified as occurring, the source begins probing nodes on the routing path by asking nodes to acknowledge all packets received. Probing is performed according to a binary search, in which the binary response of probed nodes are {failed, successful}. Once the faulty link has been identified, a weight metric is utilized to increase the value of the faulty link, thus avoiding including it in future routing paths. To avoid a misbehaving node from dropping the acknowledgments of probed nodes, the acknowledgment are attached to packets from previous nodes such that the misbehaving node cannot drop only a subset of acknowledgment messages. The source makes no attempt to identify the individual node(s) causing the misbehavior.

Acknowledgment-based systems are proactive, and hence incur message overhead regardless of the presence of misbehavior. 2ACK provides a method to reduce message overhead by acknowledging only a fraction of the packets, with the tradeoff of increased delay in misbehavior detection. Awerbuch et al. further reduces overhead through its on demand characteristic, however it only identifies the faulty link, thus failing to identify the node causing the misbehavior.

2.2 Related Work on Route Discovery

When a source node wants to communicate with a destination, it must search the network until either the destination is found or another node has a route to the destination. Due to the lack of infrastructure, and/or possible mobility, ad hoc networks rely on on-demand routing protocols [35, 50] to discover routes. Most on-demand routing protocols implement route discovery using a controlled network flooding mechanism. Examples include the Dynamic Source Routing (DSR) [35] and the Ad hoc On-Demand Distance Vector (AODV) [50] protocols. The route discovery module of AMD is based on such controlled-flooding techniques, but accounts for the reputation of individual nodes. In this section, we give a brief description of

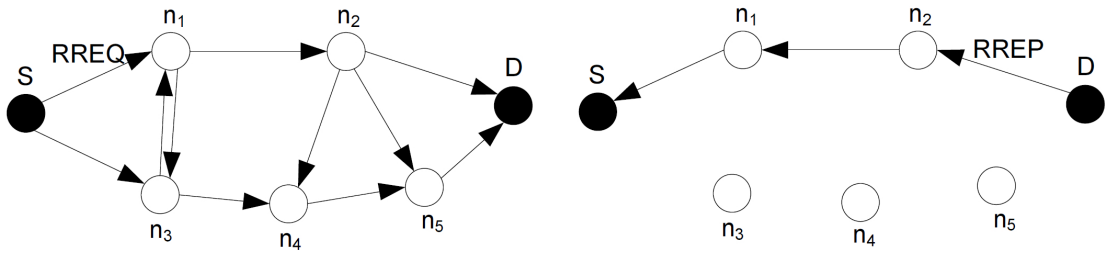

Figure 2.1: The route discovery process of DSR/AODV. (a) Controlled flooding of RREQ messages. (b) Unicasting of RREP messages.

DSR and AODV.

2.2.1 Route Discovery in DSR

When a source S has packets for a destination D, it checks whether a route exists in its cache. If a route does not exist, S broadcasts a Route Request (RREQ) message. This message contains the source ID, destination ID, and the time-to-live (TTL) for this request. Any intermediate node that receives the RREQ, appends its identity to the RREQ message and rebroadcasts the RREQ while decreasing TTL field by one. If a receiving node is the destination (or has a route to the destination), D responds to S with a route reply (RREP) message containing the entire path P_{SD} between S and D. The RREP follows the reverse path from the one indicated in the received RREQ. Note that because of the flooding nature of the route discovery process, the destination can receive RREQs indicating multiple paths from the source. It can then choose the path that satisfies the selection metric, which is typically the smallest number of hops (indicated by the smallest number of intermediate nodes in the RREQ). In Figure 2.1(a), source node S broadcasts RREQ to its neighbors n_1 and n_3. If destination D is not found, n_1 and n_3 will further broadcast this RREP to their neighbors, until D is found or another node n_i has a route to D. During the process of RREQ forwarding, each intermediate node adds its identification number to the RREQ. Each RREQ is broadcasted only once by each node, and hence, any other RREQ that arrives later from other directions will be discarded. Once RREQ arrives at D, it replies with a RREP, which traverses the reverse path indicated by

the RREQ. This path is typically the shortest. In Figure 2.1(b), the reverse path is $D \rightarrow n_2 \rightarrow n_1 \rightarrow S$. Upon reception of the RREP, P_{SD} is established.

2.2.2 Route Discovery in AODV

In AODV [50], when a source S has data for a destination D, it first checks if a route exists in its cache (similar to DSR). If a route does not exist, S broadcasts a RREQ for D. A receiving node will rebroadcast the RREQ if it is not D, it does not have a cached route to D, it is the first time that this RREQ is received, and the TTL field is not equal to zero. It will also record the identity of the one-hop neighbor that broadcasted the RREQ. Once the RREQ reaches the destination, D replies with a RREP message. This message is unicasted to the one-hop neighbor of D that transmitted the RREQ. In the same manner, the RREP will be unicasted back to the source, using the recorded one-hop neighbors that originated the RREQ. In Figure 2.1(a), the RREQ forwarding process of AODV is similar to that of DSR. The only difference is that RREQ does not contain any path information. Intermediate nodes record the identify of the upstream neighbor during the RREQ phase and downstream neighbor during the RREP phase. Each RREQ is also broadcasted only once, and hence, any other RREQ that arrives later from other directions will be discarded. Once D is found, a RREP will be unicasted to S, with each node forwarding the RREP to the recorded upstream neighbor. Similarly to DSR, AODV also typically finds the shortest path.

Note that both DSR and AODV adopt the ring expansion technique for controlling the level of flooding. In this technique, multiple RREQ with incrementing TTL values are broadcasted, until the destination is discovered.

CHAPTER 3

System Model and Problem Statement

In this chapter, we state our assumptions in regards to the network and adversarial models. We also state the problem addressed.

3.1 Network Model

We assume a multi-hop ad hoc network where nodes collaboratively relay traffic according to an underlying routing protocol such as DSR [35] or AODV [50]. The network consists of a set of $\mathcal{N}$ nodes. Any path P_{SD} used to route traffic from a source S to a destination D is assumed to be known to S. This is true for source routing protocols such as DSR. If DSR is not used, P_{SD} can be identified through a traceroute operation, or becomes known once the route is established. For simplicity, we number nodes in P_{SD} in ascending order, i.e., n_i is *upstream* of n_j if $i < j$.

We assume that the source and destination collaboratively monitor the performance of P_{SD}. The destination periodically reports to the source critical metrics such as throughput or delay. If a misbehaving node drops the periodic updates as part of its misbehavior pattern, the source interprets the lack of updates as occurrence of misbehavior. Likewise, the destination explicitly alerts the source in case the performance in P_{SD} is restored. These alerts are used to pause the misbehavior identification process and account for: (a) temporal variations of performance due to traffic or intermittent connectivity, and (b) random behavioral patterns of the misbehaving nodes. Note that the source can implicitly measure the throughput of an end-to-end connection if TCP is used at the tranport layer. Reception of end-to-end acknowledgements give accurate estimate of the "instantaneous" throughput and of the round trip delay. If TCP is employed, explicit alerting of the source is not necessary.

We assume that the integrity, authenticity, and freshness of critical control messages can be verified using cryptographic methods. For example, a public key cryptosystem can be used to verify the authenticity and integrity of messages while providing confidentiality [41]. Note that such cryptosystems require the existence of a trusted certificate authority (CA) for initialization (issuance of keys and certificates) as well as revocation of users via a certificate revocation list (CRL). Several methods have been proposed for the distributed implementation of a CA [20, 54, 63]. Because, our system isolates malicious nodes during the route selection process, a cryptographic revocation is not necessary. In this case, the CA can be eliminated after the network is initialized. Every node n_i in the network is assumed to be in possession of a private/public key pair denoted as (sk_i, pk_i). The public key is assumed to be known to all participating nodes. This is achieved either at initialization, or via the use of a certificate signed by the certificate authority. While any public key cryptosystem can be used in terms of security, in our work, we adopt a system that satisfies the homomorphic multiplicative property [56, 21]. This is because AMD performs multiplication operations on the ciphertext domain.

3.2 Adversarial Model

A fraction of the nodes deployed in the network is assumed to be misbehaving. This misbehavior is manifested by the dropping of transit traffic from a source to a destination. Misbehaving nodes can be continuous droppers, or adopt any selective dropping strategy. For a given path P_{SD} of length k, we assume that a set of $\mathcal{M}$ misbehaving nodes, with $|\mathcal{M}| \leq k$, exist along this path. These nodes can be located anywhere along P_{SD}.

In our model, we do not consider other types of misbehavior against the routing process such as advertisement of false routing information, creation of sinkholes, blackholes, wormholes, etc. [19, 18, 46, 52, 39, 61, 27, 40, 53]. Moreover, we do not attempt to detect misbehaving nodes that are unwilling to participate to routing paths. Such nodes already isolate themselves from the network and do not further

degrade the network performance.

3.3 Problem Statement

The problem addressed in this thesis is shown in Figure 3.1. A set of nodes $\mathcal{N}$ is organized in an ad hoc networks. A subset of $\mathcal{N}$ is acting maliciously by dropping packets either continuously or selectively. Our goals are: (a) to identify the subset of misbehaving nodes, and (b) to construct routes that avoid the use of the misbehaving nodes. In addressing these two goals, we are concerned with the resource-efficiency of the misbehavior identification and route discovery methods.

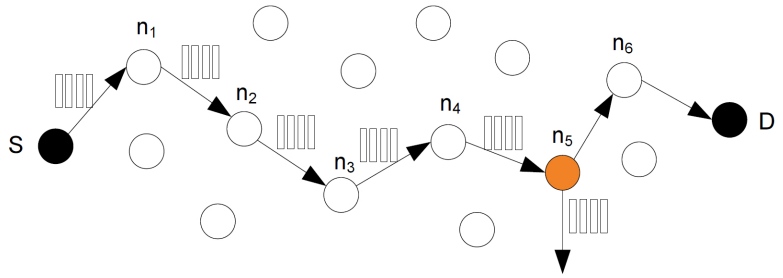

Figure 3.1: Node S is sending traffic to D along P_{SD}. Node n_5 drops all packets. Our goal is to identify n_5, provide evidence of its misbehavior, and construct routes that avoid n_5.

CHAPTER 4

The AMD System

4.1 The AMD System Architecture

In this section, we describe the architecture of the AMD system. AMD consists of three modules: the reputation module, the route discovery module, and the audit module. These modules are implemented in each node of the network and do not need any sort of centralized control. They closely interact to coordinate the functions of misbehavior detection, discovery of trustworthy routes, and evaluation of the reputation of peers. A diagram of the relationship between the three modules of the AMD system is shown in Figure 4.1.

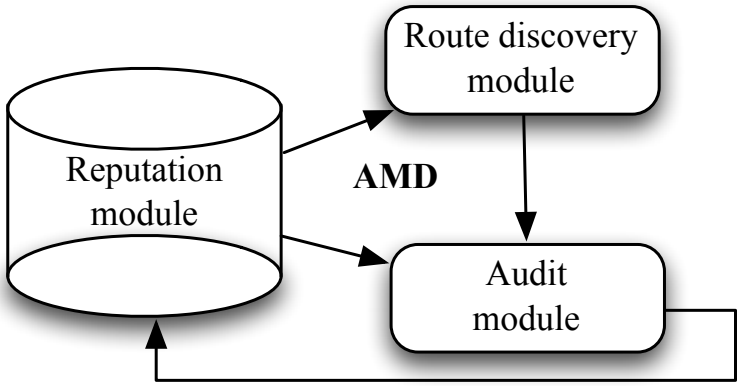

Figure 4.1: The AMD system architecture.

The reputation module is responsible for managing reputation information. Every node of the network collects first-hand and second-hand reputation information for its peers. The reputation module combines both types of information in order to compute and maintain reputation values for the rest of the nodes. The route discovery module establishes a route between a source S and a destination D, using the

reputation values computed by the reputation module. The route discovery process is based on well-known route discovery protocols for ad hoc networks such as DSR [34] and AODV [50], modified to take into account the reputation recommendations of the reputation module, in an efficient and distributed manner. The audit module is responsible for detecting misbehaving nodes along entire paths from a source to a destination. Contrary to previously proposed misbehavior detection mechanisms, the audit module operates on an end-to-end basis, thus allowing the concurrent behavior evaluation of all nodes along a particular path. Moreover, this module is designed to operate in a resource-efficient manner, by eliminating the need for energy-expensive overhearing techniques. Instead of monitoring neighboring nodes on a per-packet basis, behavioral evaluations are performed per flow. To accelerate the identification of misbehaving nodes, the audit module biases its search process according to input received from the reputation module. Once the audit module converges to the misbehaving node(s), it provides input to the reputation module for updating the reputation values.

4.2 The Reputation Module

The reputation module is responsible for computing and managing the reputation of nodes that participate in the ad hoc network. We adopt a decentralized approach in which each node maintains its own view of the reputation of other nodes. Such implementation alleviates the communication overhead for transmitting information to a centralized location, and readily translates to the distributed nature of ad hoc networks. Because reputation information does not propagate to the entire network (we avoid expensive reputation flooding operations [10, 11, 45, 26]), various nodes are likely to assign different reputation values to the same node, depending on their interaction with that node. The individual evaluation of one's reputation can detect selective misbehaviors in which malicious nodes selectively drop traffic that originates from specific sources. This method also mitigates the normalization of the reputation value, when multiple reputation values are combined via an averaging

operation.

4.2.1 Computation of the Reputation Value

The reputation module running on a node n_i maintains a reputation vector $R_i(t) = \{r_i^1(t), r_i^2(t), \ldots, r_i^N(t)\}$ that contains the reputation values $r_i^j(t)$ for all nodes $n_j \in \mathcal{N} \backslash \{n_i\}$. Here, parameter t denotes the current time epoch over which the reputation values are computed. Each epoch has a time duration of T units. The time component is added for ensuring the freshness of the reputation values. To compute and update the reputation value of a node n_j, node n_i considers two types of information; first-hand and second-hand. A reputation evaluation is considered to be first-hand if it originates from the audit module running on n_i, and is considered to be second-hand if it is computed based on information provided by other nodes. Second-hand information is only used if first-hand information is not available, or if the latter is considered to be stale (based on the number of epochs that have passed since the evaluation was made).

Reputation values $r_i^j(t)$ are restricted to range $(0, 1]$ and are initialized to some value c (e.g, $c = 0.5$). To update the reputation values, we adopt an additive increase/multiplicative decrease (AIMD) algorithm [16, 44], with a multiplicative factor $0 < \alpha < 1$ and an additive factor $0 < \beta < 1$. The AIMD algorithm is preferred to ensure that reputation of a malicious node will quickly converge to very low values when it continuously drops packets, even if it has been initially faithfully relaying packets in order to increase its reputation. The reputation value is updated either when new information becomes available (first-hand or second hand), or at the end of an epoch according to the following rules.

First-hand information: If during epoch t, the audit module of node n_i provided a first-hand evaluation of node n_j, the reputation value $r_i^j(t)$ is computed as,

$$r_i^j(t) = \begin{cases} \alpha \times r_i^j(t-1), & \text{if } \delta(t) < \gamma_0, \\ \min\{r_i^j(t-1) + \beta, 1.0\}, & \text{otherwise.} \end{cases} \tag{4.1}$$

In (4.1), $\delta(t)$ denotes a weighted fraction of packets which originated from n_i and forwarded by n_j. Parameter γ_0 denotes a threshold value below which n_j is assumed to be malicious. Parameter $\delta(t)$ becomes available from the audit module. In specific, consider that packets sent from node n_i to a destination via n_j can be classified into K categories. Moreover, $X_1, X_2, \ldots, X_K$ denotes the number of packets sent by n_i corresponding to each of the K packet categories, $Y_1, Y_2, \ldots, Y_K$ denotes the number of packets forwarded by n_j, and $w_1, w_2, \ldots, w_K$ denotes the weights assigned to each packet category, with $0 \leq w_i \leq 1$ and $\sum_i w_i = 1$. Then,

$$\delta(t) = \frac{\sum_{i=1}^{K} w_i Y_i}{\sum_{i=1}^{K} w_i X_i}. \tag{4.2}$$

The classification of packets into different categories is employed to detect nodes that selectively misbehave by dropping packets of high importance, while forwarding other packets. The case where all packets are considered to be of the same importance can be derived as special case of (4.2), by setting $K = 1$ and $w_1 = 1$.

In order to identify the misbehaving nodes, we need to properly determine the values of $w_1, w_2, \ldots, w_K$ and γ_0. The value of γ_0 is determined by the characteristics of the communication channel. That is, γ_0 depends on the physical layer properties of the communication channel. Taking into account the channel model (AWGN, Raleigh, etc.), the modulation type (BPSK, QPSK, QAM, etc.), the channel coding technique and the application of any error correction, an expected value of the packet error rate (PER) of a benign node is derived. Assuming that packets of all types $P_1, P_2, \ldots, P_K$ are uniformly dropped due to pool channel conditions for any distribution of weights $w_1, w_2, \ldots, w_K$, one can compute the threshold γ_0 as,

$$\gamma_0 = \frac{\sum_{i=1}^{K} w_i \mathrm{PER}_i X_i}{\sum_{i=1}^{K} w_i X_i} = \mathrm{PER}. \tag{4.3}$$

Formula 4.3 can be adjusted to take into account different PER values for different packet types. Such differences can occur due to the varying packet sizes and the use of different transmission rates. As an example, in 802.11 networks, control packets are always transmitted at the lowest data rate for reliability purposes while

data packets can be transmitted at a higher rate, and therefore, experience a higher PER.

The values of $w_1, w_2, \ldots, w_K$ are application dependent. The source can assign large values to packet types that are deemed to be highly important for the applications at hand. For instance, control packets are typically of higher importance.

One method for determining $w_1, w_2, \ldots, w_K$ is by setting a condition under which misbehavior is always detected. For instance consider only two types of packets P_1 and P_2 (e.g., control packets and data packets). Assume that we want to always detect a misbehaving node that drops a fraction $\frac{Y_1}{X_1} > f_0$ packets. Then w_1 and w_2 are determined as a feasible solution region derived from the following inequalities,

$$\delta(t) < \gamma_0 \tag{4.4}$$

$$\sum_i w_i = 1 \tag{4.5}$$

$$\frac{Y_1}{X_1} > f_0 \tag{4.6}$$

In the case of two packet types,

$$\frac{w_1 Y_1 + w_2 Y_2}{w_1 X_1 + w_2 X_2} < \gamma_0 \tag{4.7}$$

$$w_1 + w_2 = 1 \tag{4.8}$$

$$\frac{Y_1}{X_1} > f_0 \tag{4.9}$$

Solving (4.7), (4.8) and (4.9) for w_1 and w_2 yields the desired weight distribution.

Second-hand information: Second-hand information is used only if first-hand information becomes stale. The latter occurs when no first hand information is obtained for t_0 epochs. In this case, a node n_i averages all second-hand information reported by other nodes within the last t_0 epochs. Let $\mathcal{I}_i(t)$ denote the set of nodes that have provided second-hand information to node n_i within the last t_0 epochs. The reputation value is computed as,

$$r_i^j(t) = \frac{\sum_{m \in \mathcal{I}_i(t)} r_m^j(t)}{|\mathcal{I}_i(t)|}. \tag{4.10}$$

The reputation value in (4.10), is updated at every epoch by discarding all second-hand information older than t_0 epochs and admitting all new second-hand information. If no second-hand information is available for the last t_0 epochs, the reputation value is restored to the last known first-hand information. Also, if first-hand information becomes available, it replaces the second-hand information.

To simplify our notations, we eliminate the parameter t from the reputation expressions when unnecessary. Current reputation values are assumed unless otherwise specified.

4.2.2 Collection of Reputation Information

We now describe the process of collecting first-hand and second-hand reputation information. This process is facilitated by the audit module. When a source node S establishes a traffic flow[1] with a destination D, the audit module running on S makes evaluations on the behavior of each node along the path P_{SD}. These evaluations are considered as first-hand information for S. Moreover, the source is responsible for sending the evaluations to all nodes along P_{SD}, thus providing second-hand information.

More formally, consider a path $P_{SD} = \{S, n_1, n_2, \ldots, n_k, D\}$. Using the traffic flow from S to D, the audit module of S computes $r_S^j, (j = 1 \ldots k)$ and updates its own reputation values using (4.1). The details of the evaluation process are described in Section 4.4. It then distributes r_S^j to all nodes in $\{n_1, \ldots, n_k\} \backslash \{n_j\}$. For instance, node n_1 will receive all the updated reputation values $r_S^j, (j = 2 \ldots k)$, but its own value r_S^1.

We note that the reputation distribution operation implements a tradeoff between the communication cost of updating reputation values to the entire network

[1] We define a traffic flow as a stream of packets related to a communication session between a source and a destination.

and obtaining a current view of the reputation of other nodes. Most prior methods are either limited to first-hand information (reputation is only computed within one hop) (e.g, [1]), thus not effectively choosing trusted paths, or employ some form of (controlled) flooding thus being communication costly (e.g, [10]). We adopt a path oriented technique that achieves a desirable tradeoff between the two approaches. By limiting the propagation of second-hand information to the nodes along the relevant path of a traffic flow, the communication cost is kept low. At the same time second-hand information is provided to nodes that are likely to participate in the same paths. Nodes located at disparate parts of the network do not often share reputation information. On the other hand, nodes within the same vicinity (not necessarily one hop), receive up-to-date information. Also, the reputation of critical nodes that participate on many routes (such as the ones located in the center of the network) is frequently distributed to all nodes that use them.

Note that malicious nodes acting as sources may distribute low reputation values for nodes participating in a routing path, in order to exclude them from the network. However, this attack is mitigated in our approach. First, second-hand reputation is only used if first-hand reputation is not available. Second, the second-hand reputation value is computed by averaging all values provided by other nodes. A single false evaluation may not be sufficient to make significant difference. Third, even if a node is excluded from one path due to lies, this node may still be included in paths from other directions. Recall that only nodes along a path receive second-hand information. Assuming that the accused node is benign, the sources in other paths provide positive feedback for the falsely accused node. This positive feedback will prevail during the averaging operation, assuming a large number of behaving sources.

The integrity of the reputation values during distribution is protected using standard cryptographic methods [60]. For instance, if symmetric keys have been established in the ad hoc network using one of the key establishment methods [28, 48, 51], a message carrying reputation values from S to a node n_i can be encrypted with symmetric key K_{Si} and accompanied with a keyed message authentication

code (MAC), using K_{Si}. Alternatively, if a public key cryptosystem is established, the source signs the message containing the reputation values with its private key and encrypts it with the public key of the corresponding node.

4.3 The Route Discovery Module

The route discovery module is responsible for the discovery of trustworthy paths from a source to a destination. This module is invoked by the source, whenever there is no cached path to the destination. To ensure uninterrupted traffic flow, the discovered paths not only have to be efficient (e.g., in number of hops), but also have to be composed by nodes with high reputation values. To quantify the trustworthiness of a path, we define the following path reputation metric.

Definition 1. Path reputation value: *The reputation of a path* $P_{SD} = \{S, n_1, n_2, \ldots, n_k, D\}$ *is defined as:*

$$r_{S \to D} = \sqrt[k+1]{\prod_{i=1}^{k} (r_S^i \cdot r_D^i \cdot \prod_{j=1, j \neq i}^{k} r_i^j)}. \tag{4.11}$$

The path reputation value is inline with the in-series nature of the routing operation. Specifically, a path P_{SD} can be seen as an in-series system of independent components. The failure of one component (a node dropping packets) results in the failure of the entire system (path). Therefore, the system (path) is deemed as reliable (trustworthy) as the weakest component (node). The reliability of in-series systems with independent components is typically defined as the product of the reliability values of the individual components. Similarly, in (4.11), the trustworthiness of a path is defined as the product of the reputation values of the nodes that participate in that path. A subtlety in our definition is the fact that there is no universal reputation value for each node, but the reputation values are individual perceptions of trustworthiness of one node in regards to another. Hence, to compute the path reputation value, we multiply the reputation values r_i^j of intermediate nodes as perceived by the nodes participating in P_{SD}. Here, we emphasize that none

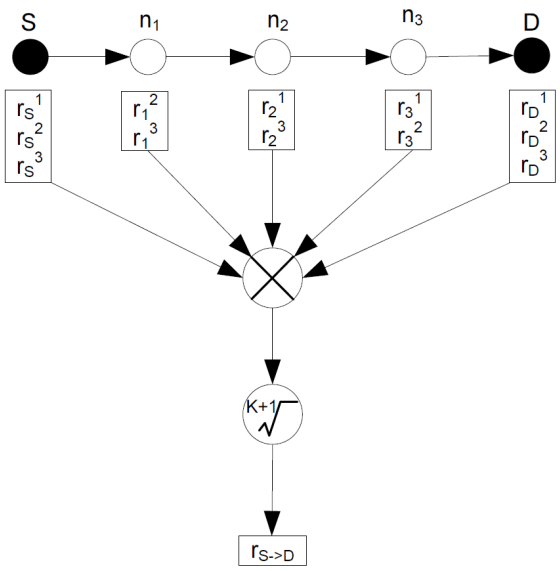

Figure 4.2: An example of computing the path reputation of our DSR-based route discovery process.

of the nodes in P_{SD} is providing an evaluation of itself. The calculation of $r_{S \to D}$ on a sample path is shown in Figure 4.2.

Note that according to Definition 1, a malicious node with low reputation value cannot manipulate the reputation of a path that includes it to appear attractive. That is, it cannot increase the path reputation to a value higher to its own reputation. Hence, it cannot influence the route discovery process to attract routes and then drop packets. A malicious node can, however, lower the reputation value of a path by lying about the reputation values of other nodes. We consider this particular strategy to be of no use to the adversary, since its main goal is to drop packets routed through it. Decreasing the reputation of a path that includes the malicious node can only lead to the exclusion of that node from active paths.

To discover trustworthy paths, we adapt the discovery mechanisms of the two dominant on-demand routing protocols, DSR [34] and AODV [50], to take into account path reputation values.

4.3.1 DSR-based Route Discovery

In DSR, when a source S has packets for a destination D, it checks whether a route exists in its cache. If a route does not exist, S broadcasts a Route Request (RREQ) message. This message contains the source ID, destination ID, and the time-to-live (TTL) for this request[2]. Any intermediate node that receives the RREQ, appends its identity to the RREQ message and rebroadcasts the RREQ while decreasing TTL field by one. If a receiving node is the destination (or has a route to the destination), D responds to S with a route reply (RREP) message containing the entire P_{SD}. The RREP follows the reverse path from the one indicated in the received RREQ. Note that because of the flooding nature of the route discovery process, the destination can receive RREQs indicating multiple paths from the source. It can then choose the path that satisfies the selection metric of interest, which is typically the smallest number of hops (indicated by the smallest number of intermediate nodes in the RREQ).

We modify the DSR discovery phase, in order to account for the reputation values of the intermediate nodes. RREQ and RREP messages are extended to include information used for the computation of the reputation of the paths discovered by DSR. We call this additional field that holds the path reputation as the *accumulated path reputation* (APR) field. This field is initialized by the source to a random value $r_0 \in (0, 1]$. Intermediate nodes receiving a RREQ, multiply the APR field with the reputations of all the nodes indicated in the RREQ (nodes upstream to the destination) and rebroadcast the message. Similarly, nodes receiving a RREP, multiply the APR field with all reputation values of all nodes located downstream towards the destination. Formally, the route discovery is executed as follows (for simplicity, we denote nodes along a path P_{SD} in ascending order).

Step 1: Source S initializes APR to a random value $r_0 \in (0, 1]$. It encrypts APR

[2]To avoid flooding the entire network with RREQs, DSR adopts the ring expansion technique [34]. Initially, the destination is searched among the immediate neighbors by setting TTL=1. If D is not found, the search is expanded to two hops by setting the TTL=2. This expansion is continued until the destination is found.

with D's public key, and computes a signature on the APR field. The broadcasted RREQ is as follows (where || represents concatenation.),

RREQ: $S \,||\, D \,||\, \text{TTL} \,||\, E_{pk_D}(r_0) \,||\, E_{pk_D}(r_0) \,||\, sig_{sk_S}(r_0)$

Step 2: An intermediate node n_i receiving a RREQ, multiplies all of its own reputation values $r_i^j, j = 1 \ldots, (i-1)$, for the nodes n_j included in the path so far (upstream nodes). It encrypts $\prod_{j=1}^{i-1} r_i^j$ with D's public key (pk_D) and multiplies the result with the APR field. It then adds its ID on the RREQ, decreases the TTL field, and rebroadcasts the RREQ. The RREQ broadcasted by intermediate node n_i is as follows, (Here, APR_i denotes the APR value computed by n_i.)

RREQ: $S \,||\, D \,||\, \text{TTL} \,||\, n_1, \ldots, n_i \,||\, E_{pk_D}(\text{APR}_{i-1}) E_{pk_D}(\prod_{j=1}^{i-1} r_i^j)$
$||\, E_{pk_D}(r_0) \,||\, sig_{sk_S}(r_0)$

Step 3: For every RREQ indicating a unique path to the source, the destination decrypts the APR field using its private key sk_D. It also recovers the initial random value r_0 and verifies the signature $sig_{sk_S}(r_0)$ using S's public key. D rejects the route if $\text{APR}_k \geq r_0$. (n_k is the last node before D.) Otherwise, it obtains the actual APR value by dividing APR with r_0, multiplies the result with $r_D^j, j = 1, \ldots, k$, and computes the $[(k+1)/2]^{\text{th}}$ root of the APR. The latter serves as an estimate of $r_{S \to D}$.

Step 4: For all paths longer than the shortest path by a factor $\lambda \geq 1$ (route selection factor), with an $r_{S \to D}$ estimate greater than the threshold value γ_1, the destination issues a RREP message. The RREP contains the APR value multiplies by r_0 and encrypted with pk_S. It also contains a second copy of $E_{pk_S}(\text{APR}_D)$ and a signature $sig_{sk_D}(\text{APR}_D)$. The broadcasted RREP is as follows,

RREP: $S \,||\, D \,||\, \text{TTL} \,||\, n_1, \ldots, n_k \,||\, E_{pk_S}(\text{APR}_D) \,||\, E_{pk_S}(r_0) \,||\, sig_{sk_D}(r_0)$

Step 5: An intermediate node n_i receiving a RREP, multiplies all of its own repu-

Figure 4.3: Potential attack on the APR field. Node n_3 resets the APR field to 1 by replacing it with $E_{pk_D}(1)$ in the forward direction.

tation values $r_i^j, j = i+1 \ldots k$, for the nodes included in the *reverse* path so far. It encrypts $\prod_{j=i+1}^{k} r_i^j$ with S's public key (pk_S) and multiplies the result with the APR field. The broadcasted RREP is as follows,

RREP: $S \,||\, D \,||\, \text{TTL} \,||\, n_1, \ldots, n_k \,||\, E_{pk_S}(\text{APR}_{i+1}) E_{pk_S}(\prod_{j=i+1}^{k} r_i^j)$
$||\, E_{pk_S}(\text{APR}_D) \,||\, sig_{sk_D}(\text{APR}_D)$

Step 6: For any received RREP, the source decrypts the APR field using its private key sk_S and recovers APR_1. (Node n_1 is the first node in the path to D.) It also decrypts the APR value computed by D and verifies D's signature. Upon successful verification, S accepts a route only if $\text{APR}_1 \leq \text{APR}_D$. It then multiplies APR_1 with all reputation values $r_S^j, j = 1, \ldots, k$, and divides it by r_0. At this point, the path reputation is obtained by taking the $(k+1)^{\text{th}}$ root. Finally, the source selects the route with the highest path reputation.

We now explain all steps in detail. In Step 1, S encrypts the APR field using D's public key. Therefore, only D can decrypt the APR field using its private key sk_D. Moreover, S encrypts the initial APR value r_0 and signs it. This value is used by the destination to recover the actual APR value. It is also used to detect a possible manipulation of the APR field by any intermediate nodes. Such manipulation is possible because APR field can be changed by any intermediate node without violating any cryptographic conditioin. Hence, a malicious node may attempt to increase its value in order to make a route appear more attractive. For

instance, consider the path, shown in Figure 4.3, consisting of four intermediate nodes. Assume that n_3 is malicious. Node n_3 may reset the APR field to 1 by replacing it with $E_{pk_D}(1)$ in the forward (backward) direction. Such a replacement will eliminate the opinions of all upstream (downstream) nodes in regards to P_{SD}. To mitigate this attack without employing expensive signature aggregation techniques [7], we initialize APR to a random value $r_0 \in (0, 1]$. Without knowledge of r_0, the malicious node n_3 does not know how to properly replace the APR field. The destination will reject the route if it receives an APR value that is higher than r_0. Random guessing of r_0 can either lead to path rejection or a route with small reputation.

In Step 2, intermediate nodes first compute the reputation of the path upstream to the source. Then, they encrypt this value with D's public key and multiply the result with the old APR value. Here, we employ an encryption method that implements *multiplicative homomorphism* such as RSA [56] or Elgamal encryption [21]. An encryption method is said to be multiplicatively homomorphic if for two plaintext values a, b, it holds that,

$$E_{pk_i}(a) \cdot E_{pk_i}(b) = E_{pk_i}(ab). \tag{4.12}$$

This property is exploited so that the APR value can be computed by performing multiplication on the ciphertext domain rather than the plaintext domain. Hence, intermediate nodes receiving RREQ or RREP messages do not get to know the current path reputation value, and the individual evaluations of intermediate nodes with respect to their peers remain secret.

Note that for the RREQ, we use D's public key, allowing the destination to read the current APR values of all paths from S to D. This is preferred so that the destination can reject all paths with a current path reputation estimate less than a threshold γ_1 (Step 4). Because of the multiplication operation applied on the APR field, these paths cannot have a final path reputation value $r_{S \to D} \geq \gamma_1$ (recall that all reputation values are within the range (0,1]). Hence, such paths are rejected by D, in order to reduce the communication overhead of route discovery. Here, γ_1 can

be determined as the value of $\gamma_1 = (\gamma_0)^H$ (H is the number of the shortest hops from source to destination). In addition, the destination rejects all paths that are longer than the shortest path by a factor of λ. We call this factor as the *route selection factor* because it determines the space of paths over which the most trustworthy path is selected. Here, λ controls the tradeoff between path reputation and route efficiency.

In Step 5, RREP travels on multiple reverse paths, which are determined by λ and γ_1. Intermediate nodes receiving a RREP, multiply the APR value with the reputations of all the nodes located downstream towards the destination, as indicated by the RREP message. Similarly to the RREQ operation, this multiplication occurs on the ciphertext domain using S's public key, thus hiding the APR value from the intermediate nodes.

On the final step, S decrypts the APR value using its own private key. It also decrypts APR_D and verifies D's signature. It then checks that $\text{APR}_1 \leq \text{APR}_D$ to verify that intermediate nodes have not manipulated the APR field. Finally, it multiplies the APR value with its own reputation values, divides with r_0 and computes the $(k+1)^{\text{th}}$ root. It is straightforward to verify that the final APR value is the path reputation value, as defined in Definition 1.

The source then selects the path with the highest path reputation, as the single routing path to D. We emphasize that several alternative selection strategies are possible. For instance, instead of selecting the path with the highest path reputation value, the source can select the shortest path with a reputation value higher than the threshold γ_1. Using this strategy, shorter routes are preferred as long as they satisfy a minimum reputation requirement. Moreover, our route discovery process enables the caching of backup paths in case the primary path fails.

An example of the route discovery process outlined in Steps 1-6 is shown in Figure 4.4. After the RREQs have been forwarded to the destination, D considers 4 paths $P_1 - P_4$. Path P_4 is rejected because $|P_4| > \lambda|P_2|$, where P_2 is the shortest discovered path. Moreover, path P_2 is also rejected because $r_{P_2} < \gamma_1$ (where $\gamma_1 = 0.6$). D replies with a RREP over P_1 and P_3. At the source, P_1 is selected because $r_{P_1} > r_{P_3}$. Note

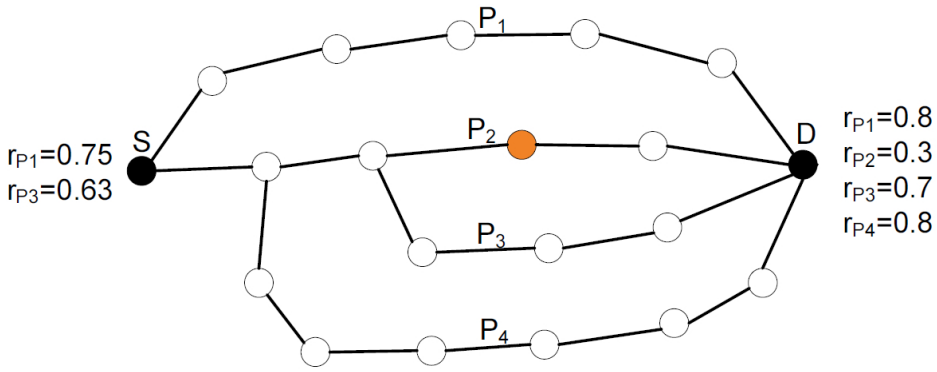

Figure 4.4: Example of DSR-based route discovery, for $\lambda = 1.2$ and $\gamma_1 = 0.6$. The destination rejects P_2 and P_4 because $r_{P_2} < \gamma_1$ and $|P_4| > \lambda|P_2|$. Path P_1 is preferred because $r_{P_1} > r_{P_3}$.

that although in our example, $r_{P_1} > r_{P_3}$ both at the source and the destination, it is possible that the direction of the inequality is reversed when the RREP reaches the source.

4.3.2 AODV-based Route Discovery

In AODV [50], when a source S has data for a destination D, it first checks if a route exists in its cache (similar to DSR). If a route does not exist, S broadcasts a RREQ for D. A receiving node will rebroadcast the RREQ if it is not D, it does not have a cached route to D, it is the first time that this RREQ is received, and the TTL field is not equal to zero. It will also record the identity of the one-hop neighbor that broadcasted the RREQ. Once the RREQ reaches the destination, D replies with a RREP message. This message is unicasted to the one-hop neighbor of D that transmitted the RREQ first. In the same manner, the RREP will be unicasted back to the source, using the recorded one-hop neighbors that originated the RREQ.

In the AODV route discovery process, the path from the source to the destination does not become fully known to the path nodes, including the source and the destination. Because the audit module requires knowledge of P_{SD}, the path can become known after it it established, as an auxiliary service. However, without knowledge

of P_{SD} during the path discovery, the intermediate nodes for which a reputation value must be provided cannot be identified. Therefore, the path reputation metric of Definition 1 cannot be computed.

To overcome this shortcoming, we employ an alternate path reputation definition specific to AODV. This definition is stated as follows.

Definition 2. Path reputation value: *The reputation of a path* $P_{SD} = \{S, n_1, n_2, \ldots, n_k, D\}$ *is defined as:*

$$r_{S \to D} = \sqrt{r_S^1 \cdot r_D^k \cdot r_k^{k-1} \cdot r_1^2 \cdot \prod_{i=2}^{k-1} (r_i^{i+1} \cdot r_i^{i-1})}. \tag{4.13}$$

The path reputation metric in Definition 2 is tailored to the amount of information available during the route discovery process of AODV. Essentially, because only the previous (next) hop becomes known during the reception of a RREQ (RREP), the reputation of the upstream (downstream) node is augmented at each hop. Every node provides the reputation of the previous hop during the RREQ stage and the forwarding hop during the RREP stage. The AODV-based route discovery process of AMD includes the following steps.

Step 1: Source S initializes APR to a random value $r_o \in (0, 1]$. It encrypts APR with D's public key, and computes a signature on the APR field. The broadcasted RREQ is as follows,

RREQ: $S \,||\, D \,||\, \text{TTL} \,||\, E_{pk_D}(r_0) \,||\, E_{pk_D}(r_0) \,||\, sig_{sk_S}(r_0)$

Step 2: Any intermediate node n_i receiving a RREQ from a node n_{i-1}, with $i-1 \neq S$, encrypts r_i^{i-1} with pk_D and multipies the result with the APR field. It then rebroadcasts the RREQ. The RREQ broadcasted by intermediate node n_i is as follows,

RREQ: $S \,||\, D \,||\, \text{TTL} \,||\, E_{pk_D}(\text{APR}_{i-1}) E_{pk_D}(r_i^{i-1}) \,||\, E_{pk_D}(r_0) \,||\, sig_{sk_S}(r_0)$

Step 3: When the destination receives a RREQ, it decrypts the APR field and $E_{pk_D}(r_0)$. It then verifies S's signature on r_0. If the verification is successful,

it verifies that APR $\leq r_0$, and rejects the route otherwise. Finally, it divides APR with r_0, and multiplies with r_D^k. If the final value is smaller than γ_1, the route is rejected.

Step 4: For every valid route, D constructs a RREQ that includes the APR value multiplied with r_0 and encrypted with pk_S, and a unique identifier. If further includes a second encrypted copy of the APR value, and a signature on APR. The RREP format is as follows,

$$\text{RREP:} \quad S \,||\, D \,||\, \text{TTL} \,||\, E_{pk_S}(\text{APR}_D) \,||\, \text{path\_id} \,||\, E_{pk_S}(\text{APR}_D) \,||\, sig_{sk_D}(\text{APR}_D)$$

Step 5: An intermediate node receiving a RREP from a downstream node n_{i+1} ($n_{i+1} \neq D$), encrypts r_i^{i+1} with pk_S and multiplies the result with the APR value. It also records the downstream node for the particular RREP path_id. The RREP format is as follows,

$$\begin{aligned}\text{RREP:} \quad & S \,||\, D \,||\, \text{TTL} \,||\, E_{pk_S}(\text{APR}_{i+1})E_{pk_S}(r_i^{i+1}) \,||\, \text{path\_id} \\ & ||\, E_{pk_S}(\text{APR}_D) \,||\, sig_{sk_D}(\text{APR}_D)\end{aligned}$$

Step 6: When a source receives a RREP, it decrypts $E_{pk_S}(\text{APR}_1), E_{pk_S}(\text{APR}_D)$ and verifies the signature on APR_D. Upon successful verification, it verifies that $\text{APR}_1 \leq \text{APR}_D$. For an accepted route, it multiplies APR_1 with r_S^1, divides with r_0, and computes the square root of the APR to obtain the path reputation value. If $r_{S \to D} < \gamma_1$, the route is rejected.

Step 7: The source selects the route with the highest $r_{S \to D}$. The first packet from S to D contains the path_id, so that intermediate nodes can identify the downstream node.

The proposed AODV-based route discovery mechanism does not compute the path with the highest path reputation value. This is because reputations are encrypted and hence, intermediate path reputations cannot be used by intermediate nodes for selecting optimal reputation paths. Such path selection is only allowed at the destination and the source. If reputation values are transmitted in the clear,

the selection of the most trustworthy path using either Definition 1 or Definition 2 can be solved by any distributed shortest path algorithm [32].

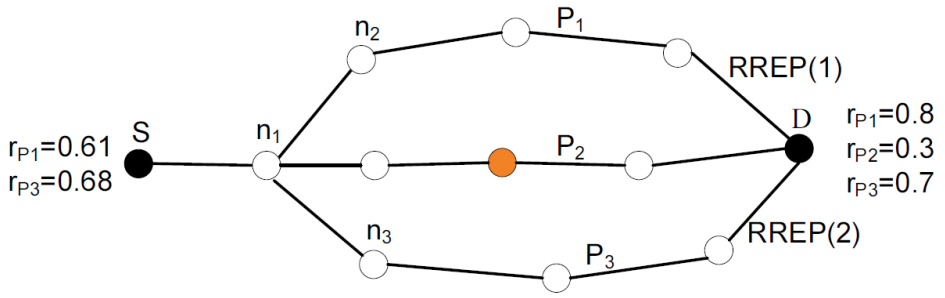

Figure 4.5: AODV-based route discovery. Three candidate paths are evaluated at D. Path P_2 containing the misbehaving node is rejected due to its low reputation value.

An example of the route discovery process outlined in Steps 1-7 is shown in Figure 4.5. The destination receives RREQ from three candidate paths. Path P_2 is rejected due to its low reputation value. The destination initiates two RREPs with path_ids 1 and 2, respectively. Both RREPs are forwarded back to the source via P_1 and P_3. The source selects path P_3 due to its higher reputation value. The initial packets forwarded from S to D contain the RREQ path_id number 2, so that n_1 is informed to forward packets to n_3 rather than n_2. Subsequent packets do not need to carry this information since the path selection is only made once.

4.3.3 Complexity Analysis of AMD Route Discovery

We notice that in the forward stage, AMD is similar to DSR or AODV. The only difference is that both DSR and AODV respond to the first arrived RREQ, which typically corresponds to the shortest path. However, AMD might responds to multiple RREQs, and lets source choose the final path. Now, consider the communication complexity of DSR/AODV and AMD. The worst scenario of DSR/AODV is that all of the nodes in the network are included in the path, and hence, all the nodes are involved in the forward and backward stages. Therefore, the communication complexity of DSR/AODV is O($2N$) (N is the number of nodes in the network) [57].

Actually, the worst scenario of AMD is the same as DSR/AODV, which is also that all the nodes are included in the path. Therefore, the communication complexity is still O($2N$). From the communication complexity's perspective, our AMD does not increase the complexity compared to normal DSR and AODV.

4.4 The Audit Module

The audit module is responsible for detecting misbehaving nodes along entire paths from a source to a destination. It operates on an end-to-end basis, thus allowing the concurrent behavior evaluation of all nodes along a particular path. Instead of monitoring neighboring nodes on a per-packet basis, behavioral evaluations are performed per flow. To accelerate the identification of misbehaving nodes, the audit module biases its search process according to input received from the reputation module. Once the audit module converges to the misbehaving node(s), it provides input to the reputation module for updating the reputation values. In this section, we describe the audit process in several stages: audit phase, search phase, and identification phase. At last, we analyze the case where multiple misbehaving nodes are included in the same path.

4.4.1 Audit Phase

We now describe how the source can perform audits in a resource-efficient manner. The audit mechanism is adopted from our REAct scheme [38] and is based on the compact representation of a membership set via Bloom filters [6]. The audit module is triggered if a performance degradation is experienced on path P_{SD}. Destination D notifies S of a performance drop by sending an explicit alarm message. Alternatively, the source can initiate the audit phase when performance information becomes available from the transport layer (e.g., based on TCP acknowledgements) Upon receipt of an alarm, S initiates a process of audits to identify the misbehaving node. The goal of auditing a node $n_i \in P_{SD}$ is to force n_i to commit to the set of packets X_i that it received and forwarded to the next hop. Contradicting

commitments are used to identify misbehaving links and eventually misbehaving nodes. To respond to an audit, the node n_i records the packets forwarded for a period of time, and reports them to the source. Based on this report, the source compares the packets in X_i with the packets in X_S originally sent to the destination. Buffering the packets themselves requires a large amount of storage and significant overhead for transmission back to the source. On the other hand, Bloom filters provide a storage-efficient way of performing membership testing [6]. The audit process occurs in three steps: sending an audit request, constructing the audit reply, and processing the audit reply. We now describe these steps in detail [37].

Sending an Audit Request

The source audits a node n_i according to the algorithms described in Section 4.4.2. The source selects the audit duration a_d, measured in number of packets, and the initial packet sequence number a_s from which the audit will begin. The value of a_d is a parameter that must be sufficiently large to differentiate misbehavior from normal packet loss. The audit request is routed to n_i via P_{SD}. Values a_s and a_d are randomized thereby preventing any misbehaving nodes from conjecturing the start and duration of audits, unless they are audited by themselves. Note that an audit request may fail to reach the audited node n_i since a misbehaving node along P_{Sn_i} may drop it, or n_i is the misbehaving node and chooses not to respond. In this case, the source tries a threshold number of times to audit n_i. Failure to obtain a reply is interpreted as "Node n_i did not forward packets in X_S to the next hop." This is true since either n_i is the misbehaving node or a misbehaving node is upstream of n_i.

Constructing an Audit Reply

When a node n_i is audited, it constructs a Bloom filter of the set of packets it receives and forwards, from a_s to $a_s + a_d$, denoted by $X_i = \{x_{a_s}, x_{a_s+1}, \ldots, x_{a_s+a_d}\}$. By using a Bloom filter, packets in X_i can be compactly represented in an m-bit vector v_i

with $m \ll |X_i|$ [6]. After a_d packets have been added to v_i, node n_i encrypts v_i with pk_S, signs it with sk_{n_1}, and sends it to S via the reverse path P_{n_iS}. The signed Bloom filter binds the audited node to the set of packets X_i that it claims to have forwarded to the next hop, in a publicly verifiable manner. Based on n_i's signature, any node can verify the authenticity and integrity of v_i. To assess the behavior of audited nodes, the source constructs its own Bloom filter v_S in the same manner as n_i. When S receives n_i's Bloom filter, it compares it against v_S and compute what fraction of packets in X_S was forwarded by n_i.

Processing the Audit Reply

When S receives v_i, it verifies its authenticity and discards v_i if the signature check fails. Otherwise, given the vector length m, the cardinalities of X_i, X_S, filters v_i, v_S, and the number z of hash functions used to generate the Bloom filters, S computes the metric [8], and obtains the reputation value.

In Section 4.2, we talked about how to compute $\delta(t)$, which denotes a weighted fraction of packets which originated from n_i and forwarded by n_j. In Formula 4.2, $Y_1, Y_2, \ldots, Y_K$ denotes the number of packets forwarded by n_j. We can use the Bloom filters v_j to determine $Y_1, Y_2, \ldots, Y_K$. Since there are K types of packets, S maintains K Bloom filters. The intersection between v_j and each of the Bloom filters from S will be $Y_1, Y_2, \ldots, Y_K$.

4.4.2 Search Phase

We now describe the audit selection process for identifying the misbehaving node. We first define the notion of a suspicious set $\mathcal{V}$ as the set of nodes $n_i \in P_{SD}$ which have not been shown to be honest. Initially, all nodes $n_i \in P_{SD}$ are placed in $\mathcal{V}$. Let A denote a one-dimensional array of length $|P_{SD}|$ with $A[i] = \delta_i$, $1 \leq i \leq |P_{SD}|$. Array A is sorted in descending order. This is a valid assumption since for any node upstream of the misbehaving node, $\delta_i \geq \gamma_0$; and for any node downstream of the misbehaving node $\delta_i < \gamma_0$. By converging on the transition $A[i] \geq \gamma_0$ and

$A[i+1] < \gamma_0$, the misbehaving link can be identified. This convergence can be achieved by performing binary search on A, requiring a maximum of $\log_2 |P_{SD}|$ steps. Let $p = |P_{SD}|$ and $\mathcal{V}_n$ be the suspicious set at iteration n, with $\mathcal{V}_0 = \{n_1, \ldots, n_p\}$. The source selects $n_i \in \mathcal{V}_0$ such that $i = \lceil \frac{|\mathcal{V}_0|}{2} \rceil$. If n_i returns $\delta_i \geq \gamma_0$, then $\mathcal{V}_1 = \{n_i, \ldots, n_p\}$; else $\mathcal{V}_1 = \{n_1, \ldots, n_i\}$. The process is repeated until $|\mathcal{V}| = 2$. Once $|\mathcal{V}| = 2$, the search has converged on the misbehaving link with termination time $\log_2 |P_{SD}|$.

Since the reputation of each node is available from the reputation module, we can exploit this information to achieve faster convergence to the misbehaving node. The assumption here is that nodes with low reputation are more likely to be malicious. Therefore, we can save time and communication overhead by only auditing those nodes that are more likely to be malicious.

Instead of selecting $n_i \in \mathcal{V}$ at random, the source identifies $n_i \in \mathcal{V}$ such that,

$$r_S^{n_i} = \min_{\mathcal{V}} r_S^i \tag{4.14}$$

It then audits node n_{i+1}. If the audit of node n_{i+1} indicates that the misbehaving node is upstream ($\delta_{i+1} < \gamma_0$), the source reduces $\mathcal{V} = \{n_1, \ldots, n_{i+1}\}$ and immediately audits n_{i-1}. If n_{i-1} indicates that the misbehaving node is downstream ($\delta_{i-1} \geq \gamma_0$), the source reduces $\mathcal{V} = \{n_{i-1}, n+i, n_{i+1}\}$and audits n_i. This step concludes in the identification of the misbehaving link. Else, if n_{i-1} indicates an upstream misbehaving node, the source reduces $\mathcal{V} = \{n_1, \ldots, n_{i-1}\}$ and repeats the audit process by selecting the node in $\mathcal{V}$ with the lowest reputation. The reputation-based binary search algorithm is shown in Algorithm 1.

4.4.3 Identification Phase

Once the search has converged on the misbehaving link, the two suspicious nodes n_i, n_{i+1} are excluded in turn from the routing path to the designation D. The node preceding the first suspicious node will split the traffic between n_i, n_{i+1} in turn. In Figure 4.6, S uses node n_2 to exclude in turn suspicious node n_3 and n_4. The source alerts D that two suspicious nodes are monitored via path exclusion. The

Algorithm 1 Reputation-based Binary Search Audit Algorithm

```
Initialize: v_l ← n_1, v_r ← n_{|P_SD|}, V_n = {v_l, ..., v_r}
while |V| > 2 do
    n_i ← arg (min_V(r_S^i))
    audit(n_{i+1})
    if δ_{i+1} ≥ γ_0 then
        v_l ← n_{i+1}
    else
        audit(n_{i−1})
        if δ_{i−1} < γ_0 then
            v_r ← n_{i−1}
        end if
    end if
end while
return V
```

destination creates two Bloom filters, v_{D_i}, $v_{D_{i+1}}$ corresponding to the packets routed through suspicious nodes n_i, n_{i+1}, and send them to S. The source compares v_i, v_{i+1} with its own filters v_{S_i}, $v_{S_{i+1}}$, and identifies the misbehaving node.

4.4.4 Multiple Misbehaving Nodes

We now examine the case of multiple independently misbehaving nodes. There exists two strategies for the nodes: (a) continuous misbehavior, and (b) randomly oscillation between honesty and misbehavior. In either case, we show S can identify, isolate, and locate the misbehaving nodes. The first step is to identify that more than one misbehaving node exists in P_{SD}, which is achieved as follows.

Case 1: Continuous Misbehavior

Assume $n_i, n_j \in P_{SD}$ are independently misbehaving and $n_i < n_j$, i.e., n_i is upstream of n_j. If n_i is misbehaving, then regardless of n_j's strategy, for all downstream nodes

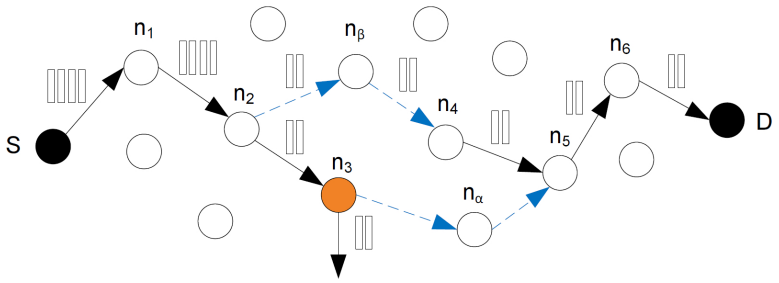

Figure 4.6: The search converges on link $n_3 - n_4$. S makes a slight alteration to P_{SD}, isolating n_3 and n_4, to determine that n_4 is the misbehaving node.

n_k, $\delta_k \geq \gamma_0$ including n_j. Executing the search process of REAct will terminate with $|\mathcal{V}| = 2$, and S will make the path alterations to exclude in turn the two suspicious nodes. Note that the suspicious set $\mathcal{V} = \{n_i, n_{i+1}\}$, where one of n_i, n_{i+1} is the misbehaving node. Let that node be n_i. Since n_j is downstream from n_i, when S excludes n_i from the path, D will still report that both paths misbehave since both paths contain n_j. The only placement such that both paths do not contains n_j is for $\mathcal{V} = \{n_i, n_j\}$, in which case excluding in turn will cause misbehavior in each path. Thus, S identifies that multiple misbehaving nodes exist.

Case 2: Changing Strategies

In the case where multiple misbehaving nodes independently change their strategy from misbehaving to honesty, it is possible for these nodes to avoid detection. As an example, consider Figure 4.7(a) and assume n_1, n_5 are malicious. Node n_5 is misbehaving while n_1 faithfully forwards all packets. If the algorithm selects node n_3 for audit, the suspicious set will reduce to $\mathcal{V} = \{n_3, \ldots, n_5\}$. In Figure 4.7(b), the behavior pattern changes while searching with n_1 misbehaving and n_5 being honest. Thus, the search converges on $\mathcal{V} = \{n_3, n_4\}$, since any node downstream of n_3 has $\delta_i < \gamma_0$. The problem arises because n_3 cannot change its response.

However, when excluding nodes in $\mathcal{V}$ in turn, both alternating paths contain misbehaving n_1. This results in both suspicious nodes misbehaving, indicating to S that P_{SD} contains multiple misbehaving nodes. If n_1, n_5 happen to change their strategy,

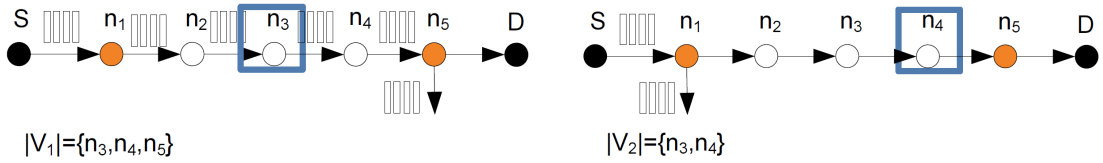

Figure 4.7: (a) n_5 misbehave while n_3 is audited. $\delta_3 \geq \gamma_0$. (b) Misbehavior strategies change (n_1 misbehaves, n_5 is honest). n_4 is audited and returns $\delta_4 < \gamma_0$. Search converged to $|\mathcal{V}| \leq 2$.

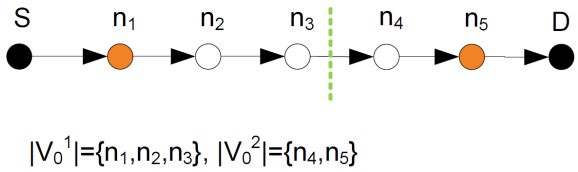

Figure 4.8: Path is partitioned into two suspicious sets; $\mathcal{V}_0^1 = \{n_1, n_2, n_3\}, \mathcal{V}_0^2 = \{n_4, n_5\}$.

both nodes appear honest. Hence, S always detects the existence of multiple misbehaving nodes. The randomization of the audit start and duration, in addition to the random search, prevents a misbehaving node from devising a strategy to frame honest nodes.

Isolation and Identification of Misbehavior

Once the source determines the existence of multiple misbehaving nodes, path P_{SD} is partitioned into two parts to isolate the misbehaving nodes from each other. The partition occurs between the two nodes in the suspicious set $\mathcal{V} = \{n_i, n_{i+1}\}$, i.e., P_{SD} is partitioned to P_{Sn_i}, $P_{n_{i+1}D}$. In Figure 4.8, S partitions $P_{SD} \rightarrow P_{Sn_2}, P_{n_3D}$.

With P_{SD} partitioned to P_{Sn_i}, $P_{n_{i+1}D}$, S executes REAct on each partition to locate the misbehaving node(s). In the single misbehaving node case, S uses alert messages from D to verify misbehavior in P_{SD} before accepting an audit response. Once P_{SD} is partitioned, S can only accept an audit response from $n_j \in P_{Sn_i}$ if misbehavior is occurring in P_{Sn_i}. The destination is unable to provide this, as D can only determine if misbehavior is in P_{SD}; not which partition.

Therefore, S simultaneously audits two nodes in each partition. In P_{Sn_i}, S audits

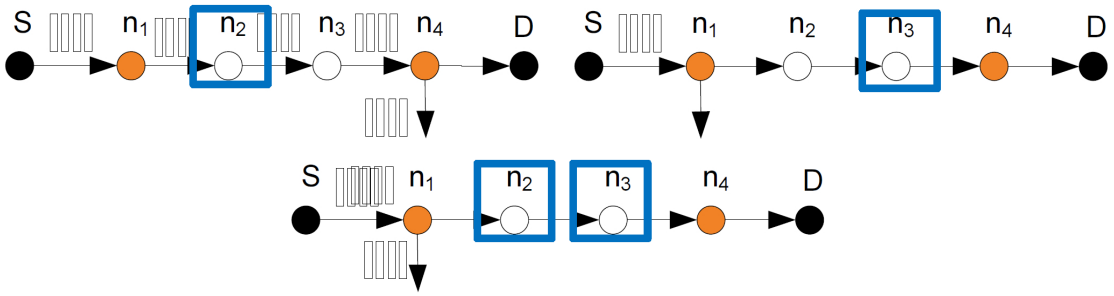

Figure 4.9: (a) Nodes n_1, n_4 collude, with n_4 dropping all packets. Audited node n_2 claims misbehavior is downstream. (b) Nodes n_1, n_4 alter their behaviors, with n_1 dropping all packets. Audited node n_3 claims misbehavior is upstream. (c) Source simultaneously audits n_2, n_3 to verify if misbehaving link exists.

a randomly selected $n_j \in P_{Sn_i}$ for the search, and node n_i. Auditing n_i determines if misbehavior is occurring in P_{Sn_i} by checking if X_i: $\delta_i \geq \gamma_0$. Thus node n_i acts as a pseudo-destination. If $n_i = n_M$, it has only two strategies, (a) respond honestly with X_M: $\delta_M \geq \gamma_0$, thus facilitating REAct or (b) lie and return X_M: $\delta_i < \gamma_0$, in which case node n_{M+1} would return X_{M+1}: $\delta_{M+1} \geq \gamma_0$, thus identifying the link $n_M - n_{M+1}$ has misbehaving.

Likewise on $P_{n_{i+1}D}$, S audits two nodes; a randomly selected $n_k \in P_{n_{i+1}}$ for the search, and node n_{i+1}. The audit response of n_{i+1} acts as a verification of how many packets from X_S have reached the partition to detect misbehavior. Node n_{i+1} therefore acts as a pseudo-source for $P_{n_{i+1}D}$. With these steps, REAct can be recursively executed to find multiple independently misbehaving nodes on P_{SD}.

4.4.5 Colluding Adversaries

If two or more nodes collude, the source may converge on a link in which both nodes are behaving, as shown in the following example. In Figure 4.9(a), $M = \{n_1, n_4\}$ with nodes n_1, n_4 colluding. Initially, n_4 drops all packets, while n_1 behaves. Let node n_2 be audited and report no misbehavior, thus $\mathcal{V} = \{n_2, n_3, n_4\}$. Assume now that nodes n_1, n_4 switch their behavior with node n_1 dropping packets while n_4 is behaving, as shown in Figure 4.9(b). If node n_3 is audited, it will report misbehavior upstream, reducing $\mathcal{V}$ to $\{n_2, n_3\}$ and thus removing n_4 from $\mathcal{V}$. Hence, link (n_2, n_3)

is incorrectly identified as misbehaving.

The problem can be solved through the repetitive questioning of the result, thereby exhausting the responder's lies. In our case, a simultaneous audit on nodes n_i, n_{i+1} of an identified link $\mathcal{V} = \{n_i, n_{i+1}\}$ is sufficient to identify a misbehaving link or the occurrence of a lie. If $\delta_i \geq \gamma_0$ and $\delta_{i+1} < \gamma_0$, a misbehaving link is identified. Else, the source concludes that a lie occurred. Returning to our previous example, in Figure 4.9(c), n_2 and n_3 are simultaneously audited. Since both nodes are honest, they return identical audit replies and no misbehaving link is identified. Therefore, S can identify that a lie occurred.

When the source identifies a lie occurred, it can also reach to the following conclusion: either (a) $n_M \in \mathcal{V}$ but lied during the simultaneous audit, or (b) $|M| \geq 2$ with at least one misbehaving node upstream of n_{i+1} and one downstream of n_i. Note that if $|M| = 1$ and the misbehaving node stops misbehaving (due to the fact that it is being audited) the destination alerts the source that misbehavior has stopped in P_{SD}. In such a case, the source will take two steps. First, any outstanding audits will be discarded. Second, the search will be suspended at the current state until misbehavior re-appears on P_{SD}. When misbehavior is resumed, the source continues the search from where it left off the last time misbehavior occurred.

4.4.6 Mobility

We now relax our assumption that P_{SD} does not change during the identification process. Let a node n_i be removed from P_{SD}. If $n_i \notin \mathcal{V}$, then its removal has no effect on the search. The source identifies misbehaving links from the nodes in $\mathcal{V}$. Let $n_i \in \mathcal{V}$. There are two cases, either n_i is a behaving node, or n_i is misbehaving. If n_i is behaving, then removing it is analogous to reducing $\mathcal{V}$ to a smaller set that still contains the misbehaving node. If n_i is misbehaving, then the performance in P_{SD} is restored or one less misbehaving node is present.

Consider now adding a new node n_i to P_{SD}. If n_i is added between nodes in $\mathcal{V}$, then regardless of n_i's behavior, this is equivalent to n_i being in $\mathcal{V}$, in the first place and not yet been audited. Let n_i be added in P_{SD} outside $\mathcal{V}$. If n_i is an honest

node, there is no effect on the audit process. If n_i is a misbehaving node, then this is equivalent to the situation in which $|M| \geq 2$ and one of the n_M has been removed from $\mathcal{V}$. However, we have shown that our auditing strategies can address the case of multiple misbehaving nodes. Once this node is removed, the source will continue to identify the newly added misbehaving node.

CHAPTER 5

Performance Evaluation

In this chapter, we evaluate the performance of the AMD system in detecting and isolating misbehaving nodes. We consider the cases where: (a) the misbehaving nodes continuously drop packets, and (b) the misbehaving nodes alternate between periods of behavior and misbehavior by selectively dropping packets. We further evaluate the resource overhead associated with the various processes of AMD, namely the route discovery process and the audit process. Finally, we compare our method with representative schemes from each class of the misbehavior detection algorithms.

5.1 Simulation Setup

We randomly deployed 100 nodes within an area of 100m x 100m. A fraction of these nodes was randomly selected to misbehave. The misbehaving nodes independently implemented a packet dropping strategy, either continuous or selective. The reputation of each node is initialized to $c = 0.5$.

We randomly selected source/destination pairs from the set of honest nodes. For each pair, we ran the route discovery module to construct a trustworthy path. In each session, the source routed 1,000 packets to the destination via the established path. If packets were dropped, the audit module was launched in order to identify the misbehaving nodes. Each experiment was repeated for 50 random network topologies. A summary of the basic simulation parameters is shown in Table 5.1.

5.2 Performance Evaluation Metrics

In our experiments, we measured the following performance evaluation metrics.

Table 5.1: Simulation Parameters

Number of nodes	100
Number of source/destination pairs	1,000
Size of area	100m x 100m
Simulation epochs	10,000
Average path length	5
Packets transferred per session	1,000
Threshold γ_0	0.6
Threshold γ_1	0.6

-**Average reputation value $r^i(t)$:** The average reputation value $r^i(t)$ of a node n_i at time epoch t is defined as the average of the individual reputations $r^i_j(t)$ of all other nodes in the network,

$$r^i(t) = \frac{\sum_{j=1, j\neq i}^{N} r^i_j(t)}{N-1}. \tag{5.1}$$

-**Percentage of dropped packets D:** The percentage of dropped packets, denoted by Fr is computed over all source/destinaton pairs every 50 epochs. It is defined as,

$$Fr = \frac{\text{\# packets dropped}}{\text{\# packets sent by sources}} \times 100\%. \tag{5.2}$$

-**Average route expansion factor E:** The average route expansion factor E is defined as the length of the path P_{SD} discovered by AMD, over the length of the shortest path P_{SD}, averaged over all paths discovered during the course of the simulation.

-**Normalized route discovery overhead:** The normalized route discovery overhead is measured as the communication overhead of AMD for discovering routes, normalized over the communication overhead of AODV and considered over all paths discovered during the simulation.

-**Audit communication overhead:** The audit communication overhead is defined as the number of messages that need to be transmitted/received in order to

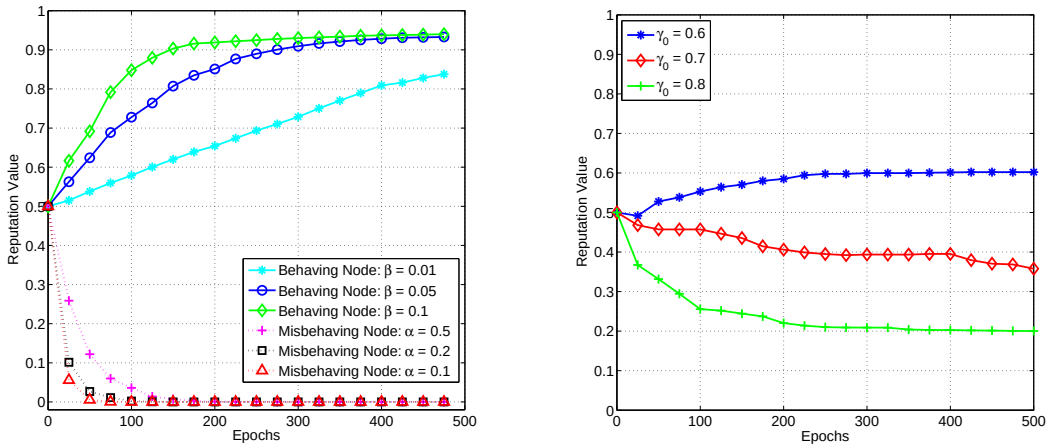

Figure 5.1: (a) The average reputation value of behaving and misbehaving nodes for different α, β, as a function of the simulation epochs. (b) The average reputation value of nodes experiencing 30% packet loss due to poor channel conditions.

identify the misbehaving node(s) on a given path. The transmission communication overhead is considered to be double the reception overhead.

5.3 Evaluation of the Evolution of Reputation Values

In the first set of experiments, we evaluated the evolution of the average reputation value as a function of time and for different parameters α, β, when 20% of the nodes are misbehaving and the misbehaving nodes are continuous droppers. Parameter α determines the rate of decrease of the reputation value of a misbehaving node, while parameter β determines the rate of increase of the reputation value of an honest node.

Figure 5.1(a) shows the average reputation value of honest and misbehaving nodes for different α, β, as a function of the simulation epochs. We observe that the reputation value of malicious nodes rapidly decreases with the progress of time, even for large values of α. On the other hand, the reputation value of honest nodes increases with the progress of time, and approaches the maximum value of one.

Figure 5.1(b) shows the average reputation value of nodes experiencing 30% packet loss due to poor channel conditions for different threshold γ_0. We notice that

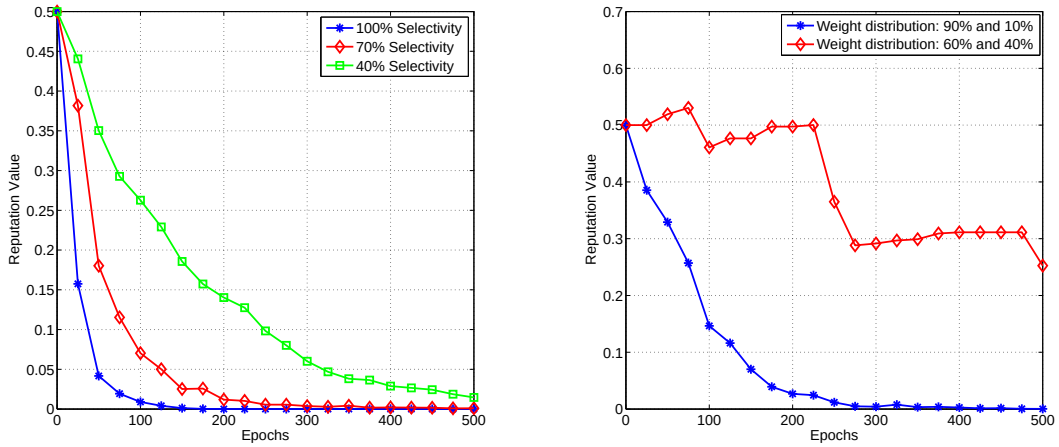

Figure 5.2: (a) The average reputation value of misbehaving nodes when such nodes drop a different percentage of their incoming traffic at random. (b) The average reputation of misbehaving nodes when they drop packets of type P_1 with different weight distribution.

the reputation value increases slowly when $\gamma_0 = 0.6$. This is expected because the expectation of the received packets is 70%. Therefore, most of the bad channels are considered as behaving nodes. However, when γ_0 reaches 0.7 or larger, the reputation value decreases, because the number of lost packets becomes larger than the threshold. For the sake of the routing function, a node that drops packets due to poor channel conditions should not be included in the routing paths. Hence, its reputation should be lowered. In this thesis, we don't make any attempt to differentiate the patterns of a dropper due to pool channel conditions versus a malicious dropper.

In Figure 5.2(a), we show the average reputation value of misbehaving nodes, when such nodes drop a different percentage of their incoming traffic at random. We observe that the reputation of such nodes approaches zero, though the decrease is slower for smaller percentages of dropped packets. We further observe that for a node that drops a fraction of packets close to $(1 - \gamma_0)$, the reputation value can increase on certain time intervals. However, this value will eventually converge to zero, because of the multiplicative decrease property.

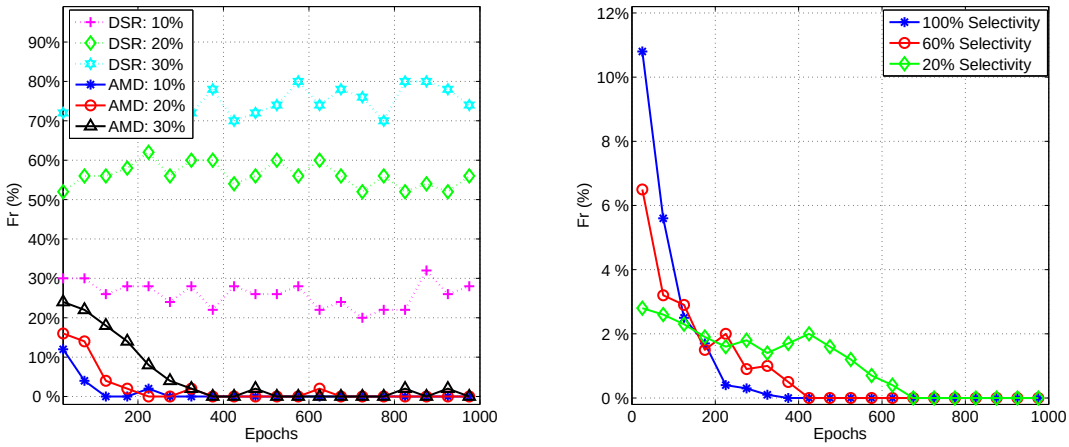

Figure 5.3: (a) The percentage of dropped packets as a function of time for performance between AMD and DSR under different fractions of misbehaving nodes. (b) The percentage of dropped packets as a function of time when different fractions of transit traffic is randomly dropped.

In Figure 5.2(b), we show the average reputation of misbehaving nodes when they employ a selective dropping strategy. In this experiment, we considered two types of packets P_1 (e.g., control packets), and P_2 (e.g., data packets). In the traffic stream, 10% are P_1 and 90% are P_2. Packets of different types were assigned weights w_1 and w_2, based on their significance. The misbehaving nodes were assumed to drop packets of type P_1 with 90% of probability, and to drop packets of type P_2 with 10% of probability. From Figure 5.2(b), it is evident that when the weight distribution is 90% and 10%, dropping packets results in identifying droppers as misbehaving nodes faster. This is expected since packets of type P_1 are assumed to be the more important on the traffic stream. On the other hand, for a weight distribution of 60% and 40%, the reputation value of droppers exhibits an oscillating behavior with an downward trend. Note that this reputation value oscillation may be sufficient to exclude misbehaving nodes from routing paths because their reputation is still lower compared to the reputation of honest nodes.

5.4 Evaluation of the Percentage of Dropped Packets

We first measured the percentage of dropped packets when misbehaving nodes continuously drop packets. We considered networks with different percentages of misbehaving nodes. We compared the performance of AMD with the performance of DSR (route discovery independent of misbehavior). Figure 5.3(a) shows Fr as a function of time. As expected the percentage of dropped packets for DSR is fairly constant, since DSR is not designed to avoid nodes with poor reputation. The small variation observed is related to the random selection of source/destination pairs. On the other hand, AMD quickly reduces the percentage of dropped packets to almost zero. This is due to the fact that misbehaving nodes are excluded from the routing paths as their reputation value decreases. Hence, most routes consist only of honest nodes.

We further implemented a random dropping behavior in which misbehaving nodes randomly drop a fraction of the transit traffic. Figure 5.3(b) depicts the percentage of dropped packets in the entire network as a function of time when 20% of the nodes are misbehaving. We observe that a less aggressive behavior can only delay the isolation of misbehaving nodes. Eventually the reputation values of such nodes becomes low enough that they are excluded from routing paths and the percentage of dropped packets becomes practically zero.

5.5 Evaluation of the AMD Overhead

In this section, we evaluate the overhead of the route discovery and audit modules of AMD.

5.5.1 Average Route Expansion Factor

We first evaluated the average route expansion factor of AMD, relatively to the shortest paths discovered by DSR and AODV. Routes computed by AMD are expected to be longer when misbehaving nodes are located in the shortest path from a source to a destination, because the path reputation value of those shortest paths is

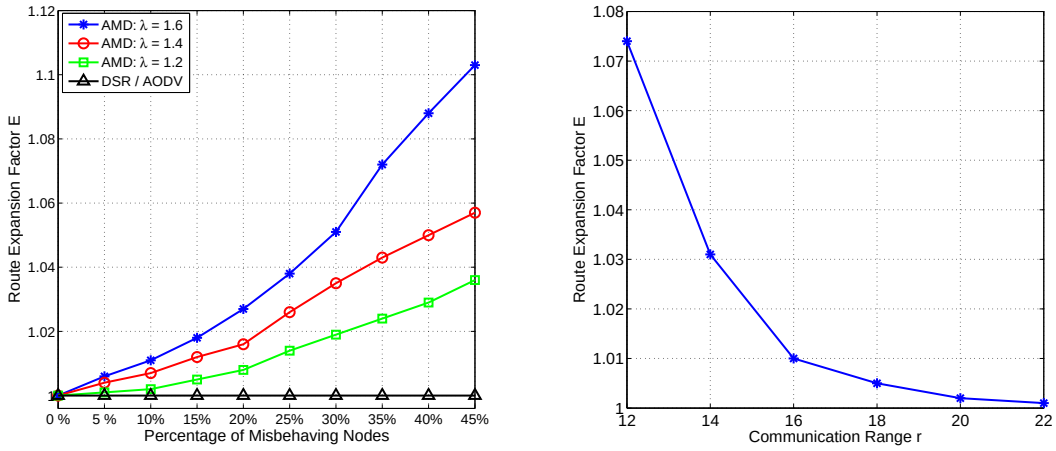

Figure 5.4: (a) The average route expansion factor E as a function of the percentage of misbehaving nodes, for various values of the route selection factor λ. (b) The average route expansion factor E as a function of the nodes' communication range.

expected to be low. Figure 5.4(a) shows the average expansion factor as a function of the percentage of misbehaving nodes, for various values of the route selection factor λ. The value of λ represents the extend to which AMD expands the search for the most reputable path, relative to the shortest path. From Figure 5.4(a), we observe that for $\lambda = 1.2$, the average length of paths discovered by AMD are only 4% longer than the shortest paths even when 45% of the nodes misbehave. This value increases to only 11% when the search is expanded to paths up to 60% longer than the shortest path for the same number of misbehaving nodes. This performance indicates that AMD pays a relatively small penalty for avoiding the misbehaving nodes. Moreover, we can conclude that even when the search is expanded over a large number of relatively long paths, shorter paths are still preferred. This can be explained by the multiplicative nature of the reputation path value that penalizes longer paths.

In Figure 5.4(b), we show the average expansion factor as a function of the nodes' communication range. An increase in communication range translates to an increase on the number of neighboring nodes, thus increasing the number of paths from a source to a destination. With the increase of the number of available route choices,

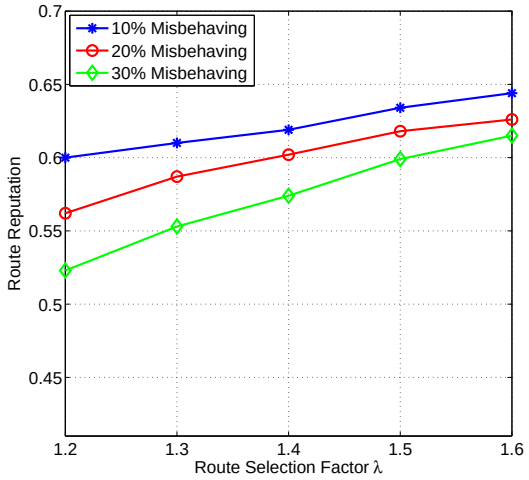

Figure 5.5: The average path reputation value of the paths discovered by AMD as a function of the route selection factor λ and for different number of misbehaving nodes.

the route expansion factor becomes almost equal to one.

Figure 5.5 shows the average path reputation value of the paths discovered by AMD as a function of the route selection factor λ and for different number of misbehaving nodes. We observe that increase of λ marginally increases the reputation of the paths discovered by AMD with the largest increase being realized when a large fraction of nodes is misbehaving. This reinforces the fact that a small value of λ should be sufficient to discover reputable paths. As we show in the next section, small λ is desirable in order to reduce the communication overhead associated with route discovery.

5.5.2 Communication Overhead of the Route Discovery Module

We also measures the communication overhead associated with the route discovery stage of AMD. We compares the performance of AMD with the overhead of the route discovery process of DSR and AODV. Figure 5.6 shows the communication overhead of AMD normalized over the overhead of DSR/AODV as a function of the nodes' communication range, for various values of the route selection factor λ. We observe

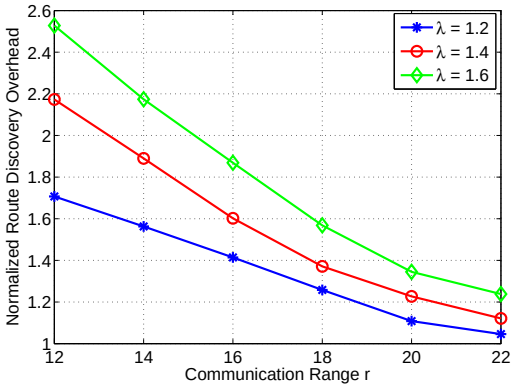

Figure 5.6: The communication overhead of AMD normalized over the overhead of DSR/AODV as a function of the nodes' communication range, for various values of the route expansion factor.

that the route discovery phase of AMD is as much as 2.5 times more expensive than that of DSR for large λ. However, as it is indicated by Figures 5.4(a) and 5.5, a value of $\lambda = 1.2$ is sufficient to discover reputable paths (the average length of the paths discovered by AMD was within 10% of the shortest path).

For $\lambda = 1.2$, the normalized communication overhead varies from 1.7 when $r = 12$ to 1.05 when $r = 22$, indicating a fairly efficient route discovery of trustworthy paths. The additional overhead of AMD is due to the multipath nature of the propagation of RREP messages. RREP messages need to travel on multiple paths so that the source can pick the one with the highest reputation value. Moreover, RREQ and RREP messages are longer due to the inclusion of the reputation-related fields.

5.5.3 Communication Overhead of the Audit Module

We compared the performance of AMD with the performance of CONFIDANT [10] from the class of reputation-based systems and the performance of 2ACK [42] from the class of acknowledgment-based schemes. Specifically, we evaluated the communication overhead associated with the identification of compromised nodes and the incurred delay. In the CONFIDANT scheme, every one-hop neighbor of a transmitting node was assumed to operate in promiscuous mode, thus overhear-

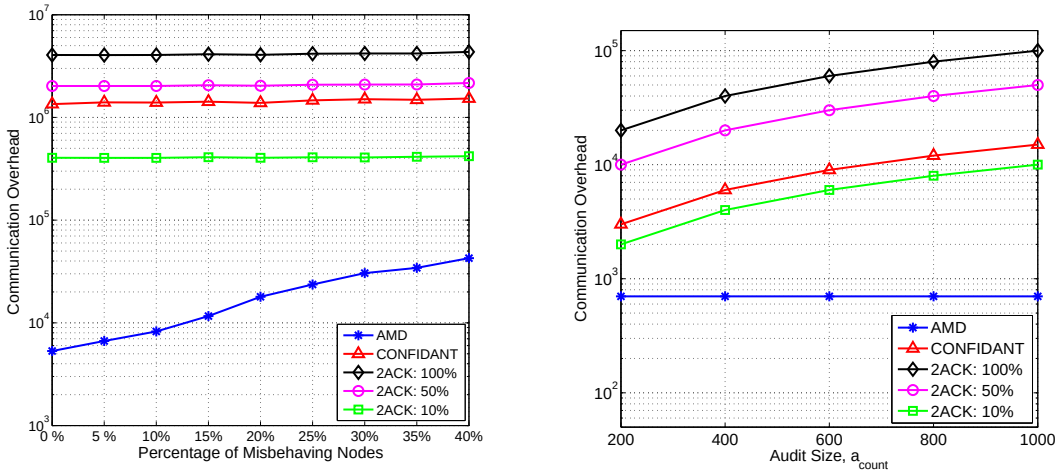

Figure 5.7: (a) The communication overhead in number of transmitted messages, as a function of the percentage of misbehaving nodes. (b) The communication overhead as a function of audit size, a_{count}.

ing transmitted messages. The energy for overhearing a message was set to 0.5 times the energy required to transmit [22]. For 2ACK, a fraction p of the messages transmitted by each node was acknowledged two hops upstream of the receiving node. We set that fraction to $p = \{1, 0.5, 0.1\}$ [42]. For all the schemes we measured the communication overhead in terms of number of messages that need to be transmitted or overheard to perform neighbor monitoring. Both CONFIDANT and 2ACK are proactive, incurring communication overhead regardless of the existence of misbehavior. However, AMD incurs overhead only if misbehavior exists, due to its reactive nature. The audit duration for AMD was set to 200 packets, i.e., each node had to provide proof for $a_{count} = 200$ packets every time it was audited.

Impact of percentage of misbehaving nodes

In Figure 5.7(a), we show the communication overhead in number of transmitted messages, as a function of the percentage of misbehaving nodes. The Y axis is logarithmic scale. We observe that communication overhead of the AMD scheme is almost 3 orders of magnitude less than the overhead of the proactive schemes. The AMD scheme incurs overhead on a per-flow basis while the other schemes

incurs overhead on a per-packet basis. For CONFIDANT and 2ACK schemes, the overhead is independent of the percentage of misbehaving nodes. However, for the AMD scheme, if more nodes are misbehaving, the overhead increases accordingly because of incurring more audits.

Impact of audit size

The audit size parameter defines the number of packets needed to differentiate normal dropping rate from misbehavior. In Figure 5.7(b), we show the communication overhead as a function of audit size, a_{count}. Both proactive schemes incur a linear increase in communication overhead with audit size. The overhead for AMD depends on the number of audits, not the duration of each audit. Hence, it is independent of the audit size. The savings in the communication overhead of AMD are due to the compact representation of the number of packers forwarded by each node in P_{SD}.

5.5.4 Identification Delay

The identification delay defined as the time elapsed from the occurrence of misbehavior until the misbehaving nodes are identified. While the AMD scheme provides significant savings in communication overhead, it requires a longer time to identify misbehavior, as multiple audits need to be performed. Proactive schemes requires only a single audit duration to identify misbehavior since all nodes in path P_{SD} are monitored in parallel. Fortunately, the audits required by the AMD grows logarithmically with path length due to the random binary search algorithm employed, resulting in fairly small increases in identification delay compared to savings in communication overhead.

In Figure 5.8, we show the identification delay for AMD, CONFIDANT, and 2ACK as a function of path length $|P_{SD}|$, in units of audits. We observe the logarithmic increase of identification delay with path length for the AMD scheme. CONFIDANT requires a single audit duration to identify misbehavior. 2ACK also requires a single audit duration when all packets are acknowledged. However, the

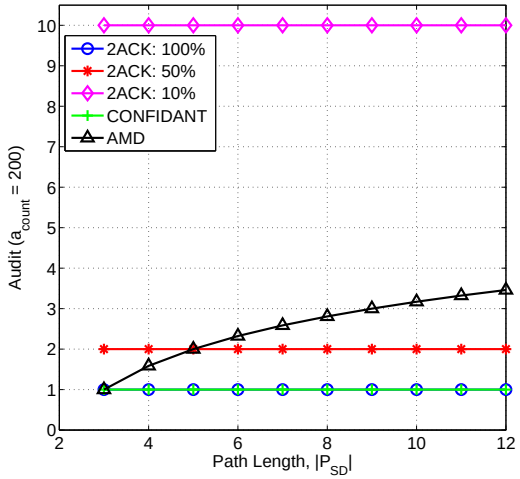

Figure 5.8: The identification delay for AMD, CONFIDANT, and 2ACK as a function of path length $|P_{SD}|$, in units of audits.

identification delay increases with the audit size.

CHAPTER 6

Conclusions

In this thesis, we addressed the problem of identifying and isolating misbehaving nodes that refuse to forward packets in wireless ad hoc network. The impact of such nodes has been shown to be detrimental to network performance, lowering the network throughput and dramatically increasing the end-to-end delay. To mitigate the problem of malicious packet dropping, we developed a comprehensive misbehavior detection and suppression system named AMD. The AMD system integrates three critical functions: reputation management, route discovery, and identification of misbehaving nodes based on behavioral audits. All three functions are coordinated in a distributed manner without the need for centralized control. This is achieved by the implementation of three modules at every node: the reputation module, the route discovery module, and the audit module.

We showed that the route discovery module can construct trustworthy paths that exclude nodes with low reputation values. Moreover, we showed that misbehaving nodes cannot manipulate reputation metrics to attract transit traffic. In case a misbehaving node is included in a path (by, for example, increasing its reputation value via honest behavior), this node is prominently identified by the audit module. This module is responsible for locating misbehaving nodes along established paths. Its function is complementary to the trustworthy route discovery function. Identification of misbehaving nodes is achieved by requesting the provision of behavioral proofs from the nodes participating in a path. These proofs are compactly represented using Bloom filters which are storage-efficient membership structures.

Our extensive simulation results showed that AMD could recover the network operation even if a large fraction of nodes were misbehaving. This is due to the fact that misbehaving nodes are excluded from routing paths by the route discovery module. This significant performance improvement comes at a cost of a small

increase in the overall path length, but a moderate increase in the communication cost for establishing trustworthy paths.

Moreover, we showed that our misbehavior identification process significantly reduced the communication overhead of locating packet droppers compared to other methods. This improved performance is justified by the savings achieved when the behavior evaluation is performed on a per-flow basis, instead of on a per-packet basis. Finally, we showed that the audit module could identify the misbehaving nodes in a time logarithmic the path length by exploiting recommendations provided by the reputation module.

REFERENCES

[1] G. Acs, L. Buttyan, and L. Dora. Misbehaving router detection in link-state routing for wireless mesh networks. In *Hot Topics in Mesh Networking (HotMesh '10)*, 2010.

[2] R. Anderson and M. Kuhn. Tamper resistance-a cautionary note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, volume 2, pages 1–11, 1996.

[3] B. Awerbuch, D. Holmer, C.-N. Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *Proceedings of the ACM Workshop on Wireless Security (WiSe'02)*, 2002.

[4] K. Balakrishnan, J. Deng, and P. K. Varshney. Twoack: Preventing selfishness in mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC'05)*, 2005.

[5] S. Basagni, M. Conti, S. Giordano, and I. Stojmenović. *Mobile ad hoc networking*. Wiley-IEEE press, 2004.

[6] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[7] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. *CryptoBytes*, 6(2):1–10, 2003.

[8] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[9] S. Buchegger and J.-Y. L. Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (EUROMICRO-PDP'02)*, pages 403–410, 2002.

[10] S. Buchegger and J.-Y. L. Boudec. Performance analysis of the confidant protocol (cooperation of nodes: Fairness in dynamic ad-hoc networks). In *Proceedings of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC 2002)*, June 2002.

[11] S. Buchegger and J.-Y. L. Boudec. Self-policing mobile ad-hoc networks by reputation systems. *IEEE Communications Magazine*, pages 101–107, 2005.

[12] S. Buchegger and J. Le Boudec. The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In *Proceedings of WiOpt 03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, volume 4, pages 4–1. Citeseer, 2003.

[13] L. Buttyan and J. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. *ICCA, Swiss Federal Institute of Technology*, 2001.

[14] L. Buttyan and J.-P. Hubaux. Enforcing service availability in mobile ad-hoc wans. In *Proceedings of the First Annual Workshop on Mobile and Ad Hoc Networking and Computing (MobiHOC'00)*, pages 87–96, 2000.

[15] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), 2003.

[16] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems*, 17(1):1–14, 1989.

[17] J. Crowcroft, R. Gibbens, F. Kelly, and S. Östring. Modelling incentives for collaboration in mobile ad hoc networks. In *Proceedings of WiOpt03*, 2003.

[18] B. Culpepper, H. Tseng, N. Center, and C. Moffett Field. Sinkhole intrusion indicators in DSR MANETs. In *Proceedings of Broadband Networks. First International Conference on*, pages 681–688, 2004.

[19] J. Deng, R. Han, and S. Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *Information Processing in Sensor Networks*, pages 552–552. Springer, 2003.

[20] Y. Dong, H. Go, A. Sui, V. Li, L. Hui, and S. Yiu. Providing Distributed Certificate Authority Service in Mobile Ad Hoc Networks. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks 2005 (SecureComm 2005)*, pages 149–156, 2005.

[21] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer, 1985.

[22] L. M. Feeney and M. Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, volume 3, 2001.

[23] S. Ganeriwal, L. Balzano, and M. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(3):1–37, 2008.

[24] 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 (SASN '04)*, pages 66–77, 2004.

[25] V. Gligor. Handling new adversaries in secure mobile ad-hoc networks. In *ARO Planning Workshop on Embedded Systems and Network Security (ESNS '07)*, 2007.

[26] Q. He, D. Wu, and P. Khosla. Sori: A secure and objective reputation-based incentive scheme for ad hoc networks. In *In IEEE Wireless Communications and Networking Conference (WCNC04)*, 2004.

[27] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. In *Network and Distributed System Security Symposium (NDSS)*, pages 131–141, 2004.

[28] Y. Hu, D. Johnson, and A. Perrig. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, 1(1):175–192, 2003.

[29] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *IEEE INFOCOM 2003*, volume 3, 2003.

[30] Y. Hu, A. Perrig, and D. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2nd ACM workshop on Wireless security*, pages 30–40, 2003.

[31] E. Huang, J. Crowcroft, and I. Wassell. Rethinking incentives for mobile ad hoc networks. In *Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 191–196. ACM New York, NY, USA, 2004.

[32] P. Humblet et al. An adaptive distributed dijkstra shortest path algorithm. *an unpublished paper*, 1989.

[33] M. Jakobsson, J.-P. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks. In *Proceedings of Financial Crypto 2003*, 2003.

[34] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. *Mobile computing*, pages 153–181, 1996.

[35] D. Johnson, D. Maltz, and Y.-C. Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr). *draft-ietf-manet-dsr-09.txt*, 2003.

[36] A. Jøsang and R. Ismail. The beta reputation system. In *Proceedings of the 15th Bled Electronic Commerce Conference*, pages 324–337, 2002.

[37] W. Kozma and L. Lazos. Dealing with liars: Misbehavior identification via rényi-ulam games. *Security and Privacy in Communication Networks*, pages 207–227, 2009.

[38] W. Kozma Jr. and L. Lazos. REAct: Resource-Efficient Accountability for Node Misbehavior in Ad Hoc Networks based on Random Audits. In *Proceedings of the Second ACM Conference on Wireless Network Security (WiSec '09)*, 2009.

[39] S. Kurosawa, H. Nakayama, N. Kato, A. Jamalipour, and Y. Nemoto. Detecting blackhole attack on AODV-based mobile ad hoc networks by dynamic learning method. *International Journal of Network Security*, 5(3):338–346, 2007.

[40] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Proceedings of IEEE Wireless Communications and Networking Conference*, volume 2, 2005.

[41] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*. SPOTS Track, 2008.

[42] K. Liu, J. Deng, P. Varshney, and K. Balakrishnan. An acknowledgment-based approach for the detection of routing misbehavior in manets. *IEEE Transactions on Mobile Computing*, 6(5):536–550, May 2007.

[43] Y. Liu and Y. R. Yang. Reputation propagation and agreement in mobile ad-hoc networks. In *Proc. of IEEE Wireless Communication and Networking Conference (WCNC'03)*, pages 1510–1515, March 2003.

[44] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 255–265, 2000.

[45] 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 (CMS02)*, 2002.

[46] E. Ngai, J. Liu, and M. Lyu. On the intruder detection for sinkhole attack in wireless sensor networks. In *Proceedings of the IEEE International Conference on Communication (ICC)*, 2006.

[47] V.-N. Padmanabhan and D.-R. Simon. Secure traceroute to detect faulty or malicious routing. *SIGCOMM Computer Communication Review*, 33(1), 2003.

[48] P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, pages 01–27, 2002.

[49] K. Paul and D. Westhoff. Context aware detection of selfish nodes in dsr based ad-hoc networks. In *Proceedings of the IEEE Globecom Conference*, 2002.

[50] C. Perkins, E. Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing, 2003.

[51] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, 2002.

[52] A. Pirzada and C. McDonald. Circumventing sinkholes and wormholes in wireless sensor networks. In *Proceedings of the International Conference on Wireless Ad Hoc Networks (IWWAN)*, 2005.

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

[54] S. Raghani, D. Toshniwal, and R. Joshi. Dynamic Support for Distributed Certification Authority in Mobile Ad Hoc Networks. In *Proceedings of the 2006 International Conference on Hybrid Information Technology-Volume 01*, pages 424–432. IEEE Computer Society Washington, DC, USA, 2006.

[55] Y. Rebahi, V. Mujica, and D. Sisalem. A reputation-based trust mechanism for ad hoc networks. In *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC05)*, pages 37–42, 2005.

[56] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[57] E. Royer and C. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE*, 6(2):46–55, 1999.

[58] N. Salem, L. Buttyán, J. Hubaux, and M. Jakobsson. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 13–24. ACM New York, NY, USA, 2003.

[59] S. Soltanali, S. Pirahesh, S. Niksefat, and M. Sabaei. An Efficient Scheme to Motivate Cooperation in Mobile Ad hoc Networks. In *Proceedings of the Third International Conference on Networking and Services*, pages 98–98, 2007.

[60] D. Stinson. *Cryptography: theory and practice*. CRC press, 2006.

[61] L. Tamilselvan and V. Sankaranarayanan. Prevention of Blackhole Attack in MANET. In *Proceedings of the 2nd International Conference on Wireless Broadband and Ultra Wideband Communications*, pages 21–21, 2007.

[62] Y. Xue and K. Nahrstedt. Providing fault-tolerant ad-hoc routing service in adversarial environments. *Wireless Personal Communications, Special Issue on Security for Next Generation Communications*, 29(3–4):367–388, 2004.

[63] S. Yi and R. Kravets. Moca: Mobile certificate authority for wireless ad hoc networks. In *The 2nd Annual PKI Research Workshop Pre-Proceedings*. Citeseer, 2003.

[64] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A simple cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of INFOCOM 2003*, pages 1987–1997, March 2003.

[65] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 2005.