ABSTRACT

Title of Dissertation: SECURING WIRELESS AD HOC NETWORKS UNDER NOISE AND IMPERFECT MONITORING

Wei Yu, Doctor of Philosophy, 2006

Dissertation directed by: Professor K. J. Ray Liu
Department of Electrical and Computer Engineering

While wireless communication has dramatically changed the way people work and interact, the wireless era continues to be plagued by insufficient security. Without necessary countermeasures, even a few attackers can break down the whole network. On the other hand, attacker detection can be extremely challenging in realistic scenarios because misbehavior may also be caused by various other factors, such as noise and uncertainty, and perfect monitoring is either impossible to achieve or too expensive to afford. In this dissertation we have investigated how to secure wireless ad hoc networks against insider attacks in noisy and hostile environments, based only on local and imperfect monitoring.

In traditional ad hoc network applications, nodes usually belong to the same authority and pursue some common goals. The inherent cooperative nature of

such networks makes them extremely vulnerable to insider attacks. For example, by dropping other nodes' packets and/or injecting an overwhelming amount of traffic, insider attackers can easily break down the whole network. In this dissertation we have first studied how to secure such ad hoc networks against insider attacks under noise and imperfect monitoring. Besides devising a set of efficient monitoring and attacker detection mechanisms to defend against routing disruption and injecting traffic attacks, we have also formally analyzed the dynamic interactions between good nodes and attackers under a game theoretic framework, where both the optimal defending strategies and the maximum damage that can be caused by insider attackers have been derived.

In many civilian applications, nodes in ad hoc networks tend to act selfishly. Stimulating selfish nodes to act cooperatively poses one key research challenge, especially in realistic contexts. In this dissertation we have also investigated how to design attack-resistant cooperation mechanisms for such networks. We have first designed an attack-resistant cooperation stimulation strategy for mobile ad hoc networks, then formally analyzed the issue of secure cooperation in ad hoc networks under a game theoretic framework. Finally, we have derived a set of reputation-based attack-resistant and cheat-proof cooperation strategies for such ad hoc networks that can work well in noisy and hostile environments under imperfect monitoring.

SECURING WIRELESS AD HOC NETWORKS UNDER NOISE AND IMPERFECT MONITORING

by

Wei Yu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:

Professor K. J. Ray Liu, Chairman
Professor Virgil D. Gligor
Professor Min Wu
Professor Gang Qu
Professor Lawrence C. Washington

DEDICATION

To my parents.

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my advisor, professor K. J. Ray Liu, who has placed his confidence on me during my Ph.D. studies. During these years, he has put so many efforts to help me find good directions and conduct high quality research. I also value his influences for my vision, attitude, energy, and desire of my professional and personal developments.

Meanwhile, I am also grateful to my parents for their love and unconditional support. They have made countless efforts to let me have better education chances and make sure that I can explore my future according to my dreams. As a consequence, all my accomplishments are also their accomplishments.

I would also like to explicitly thank the following people. I would like to thank Dr. Yan Sun, Dr. Zhu Han, Dr. Hong Zhao, Dr. Zoltan Safar, Zhu Ji, Ahmed Sadek, Beibei Wang, and Peng Qiu for invaluable cooperations during the last four years. I would like to thank professor Wu, professor Qu, professor Gligor, and professor Washington for invaluable suggestions on my research. I would also like to thank all the members of CSPL lab and other friends throughout University of Maryland for giving me such happy four years.

Finally, no happiness of achievement can be completed without sharing with my wife Fangting, and my lovely son Alex.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Motivation and Contributions

1.1 Motivation

A wireless *ad hoc network* is a group of nodes without requiring centralized administration or fixed network infrastructure, in which nodes can communicate with other nodes out of their direct transmission ranges by cooperatively forwarding packets for each other through wireless connections [69, 79]. Since ad hoc networks can be easily and inexpensively set up as needed, they have a wide range of applications, such as military exercises, disaster rescue, mine site operations, etc.

In traditional military and emergency applications, nodes in an ad hoc network usually belong to the same authority and pursue some common goals. To maximize the overall system performance, nodes usually work in a fully cooperative way, and will unconditionally forward packets for each other. We refer to such ad hoc networks as *cooperative ad hoc networks*. Recently, emerging applications of ad hoc networks are also envisioned in civilian usage [16, 18, 19, 45, 58, 60, 78, 99]. In civilian applications, nodes typically do not belong to a single authority and may not pursue a common goal. Consequently, fully cooperative behaviors, such as

unconditionally forwarding packets for each other, cannot be taken for granted. On the contrary, in order to save limited resources, such as battery power, nodes may tend to be "selfish". We refer to such ad hoc networks as *autonomous ad hoc networks*.

Since ad hoc networks are usually deployed in hostile environments and nodes may tend to be selfish, before they can be successfully deployed in practice, the following two critical issues must be resolved first: *security* and *node cooperation*. When ad hoc networks are designed without taking into consideration necessary security concerns, adversaries can easily exploit the possible vulnerabilities of the network to cause damage. In many critical applications, such as in battle fields, the potential damage caused by attacks can be fatal. Meanwhile, since nodes may be selfish, without necessary cooperation mechanisms, nodes may not be willing to help the others, and consequently the network may reach a non-cooperative state. This contradicts the original purpose of designing ad hoc networks: nodes should help each other to extend the coverage and/or to best utilize the limited resources.

Security in wireless ad hoc network has drawn extensive attentions over the past several years [3, 34, 36, 42, 58, 96, 101]. It has been realized that securing ad hoc networks can be extremely challenging. In wireless networks, due to unrestricted channel access, a variety of attacks can be easily launched, ranging from passive eavesdropping to active interfering. Since ad hoc networks usually do not have centralized monitoring or management points, the situation can be further deteriorated. Meanwhile, nodes in ad hoc networks may not have enough physical protection, and can be easily captured and compromised by adversarial parties and become insider attackers. Without necessary countermeasures, even a few attackers can break down the whole network.

Past research on securing ad hoc network have mainly focused on preventing attackers from entering the network through mechanisms such as secure key distribution and secure neighbor discovery [34–37, 42, 65, 76, 95, 101]. However, the research on defending ad hoc networks against insider attacks is still in its preliminary stage. For example, in the literature, only very few works have implicitly considered insider attackers, such as in [16, 58]. On the other hand, insider attacks can be very common in ad hoc networks due to the self-organized nature of ad hoc networks and due to that nodes can be easily captured and compromised.

Besides insider attacks, another important aspect that has been widely ignored is the effects of noise and imperfect monitoring. In general, the wireless environments are usually full of noise and uncertainties. Meanwhile, due to the limited resource constraint and distributed nature, imperfect monitoring in ad hoc networks is either impossible to achieve or too expensive to afford. How to secure ad hoc networks under noise and imperfect monitoring and how to design effective and robust monitoring mechanisms pose another research challenges. For example, how to design robust attacker detection mechanisms that are able to distinguish those malicious behavior caused by attackers from those caused by noise or incorrect monitoring? What is the maximum possible damage that can be caused by attackers under noise and imperfect monitoring? What are the optimal defending strategies? None of them have been fully addressed in the literature.

Besides security, node cooperation is also a critical issue that needs to be addressed. Recently, many schemes have been proposed to stimulate node cooperation in ad hoc networks, such as in [6, 7, 16, 18, 19, 26, 31, 58–60, 78, 80, 99, 100]. These schemes can be roughly categorized into two types: payment-based and reputation-based. In payment-based methods, such as in [7, 18, 19, 99, 100], nodes

request help from others by paying them. In reputation-based schemes, such as in [6, 16, 26, 31, 58–60, 78, 80], whether a node can get help from the others based on its reputation, which is usually determined by its past actions observed by its peers. Comparing to the payment-based schemes, the advantage of reputation-based schemes lies in that they do not require tamper-proof hardware or centralized banking service to process billing information, while the drawback is that in some situations cooperation cannot be effectively stimulated.

Although many schemes have been proposed to stimulate cooperation among selfish nodes in ad hoc networks, most of them have assumed that nodes have accurate monitoring, and will act rationally whose only goal is to maximize their own payoff. In other words, they have not considered possible malicious behaviors. However, since ad hoc networks are usually deployed in noisy and hostile environments, without taking into consideration possible malicious behaviors, those proposed schemes can easily collapse. The situation is further deteriorated in realistic contexts when the environment is noisy and the monitoring cannot be perfect, since both selfish and malicious nodes can take advantage of noise and imperfect monitoring to improve their performance or cause more damage. How to design attack-resistant and cheat-proof cooperation strategies that can work well in noisy hostile environments will be one major challenge in autonomous ad hoc networks, and none of the existing works have fully addressed it.

1.2 Contributions

In summary, this dissertation has investigated how to establish secure and reliable ad hoc network services in hostile environments under noise and imperfect monitoring. Specifically, two important issues have been addressed: 1) how to

secure cooperative ad hoc networks against insider attacks under noise and imperfect monitoring, and 2) how to design attack-resistant and cheat-proof cooperation strategies for autonomous ad hoc networks that can work well in noisy and hostile environments. The contributions lie in three aspects: secure ad hoc network protocol design, formal analysis of security and cooperation in ad hoc networks, and effective and robust monitoring mechanism design.

Secure ad hoc network protocol design: In this dissertation we have designed a series of effective secure protocols to handle various attacks. Specifically, we have presented a set of secure routing protocols to handle various routing disruption attacks, such as dropping packets attack, black hole attack, etc. By using a novel self-evaluation mechanism, malicious node detection can be significantly speeded up. Meanwhile, since they can distinguish routing disruptions caused by nodes' temporary misbehavior and those caused by malicious attacks, the proposed protocols can work well in noisy environments. Following that, we have also presented a set of effective protocols to handle various types of injecting traffic attacks. Under which the optimal strategies from the attackers' point of view is not to launch injecting traffic attacks. When stimulating cooperation among selfish nodes, we have also designed a set of secure receipt submission and credit update protocols to support attack-resistant cooperation stimulation.

Formal analysis of security and cooperation in ad hoc networks: One major contribution of this dissertation would be the formal analysis of security and cooperation in ad hoc networks under noise and imperfect monitoring, as well as under insider attacks. Specifically, for cooperative ad hoc networks, we have modeled the dynamic interactions between good nodes and attackers as securing routing and packet forwarding game. Under a game theoretic framework, we have

derived the optimal defending strategies and the maximum possible damage that the attackers can cause. For autonomous ad hoc networks, we have jointly studied the security and cooperation issues under a game theoretic framework. We have first derived the cheat-proof cooperation strategies for two nodes scenarios, then demonstrated when cooperation among selfish nodes can be effectively stimulated and how to obtain optimal cheat-proof and attack-resistant cooperation strategies for autonomous ad hoc networks under realistic scenarios.

Effective and robust monitoring mechanism design: To defend against insider attacks, one has to base on what being observed about the others' behaviors. However, due to the fully distributed nature of ad hoc networks and the limited resource constraints, perfect monitoring is either impossible to achieve or too expensive to afford. In this dissertation, we have presented several robust monitoring mechanisms to detect various malicious behaviors. Specifically, to handle routing disruption attacks, we have proposed a light-weight monitoring mechanism based on end-to-end acknowledge and intermediate node reporting. When the proposed monitoring mechanism is used, a malicious node has to either admit dropping packets, or provide reports that are most likely conflicting with others which can make them easily be detected. We have also designed a robust and cost-efficient monitoring mechanism to detect possible injecting traffic attacks, where only packet headers need to be listened and decoded. Besides that, we have also designed another robust monitoring mechanism for autonomous ad hoc networks to help detecting whether some nodes have dropped other nodes' packets or receipts, which are used to claim credits from the requesters.

1.3 Thesis Organization

The rest of this dissertation is organized as follows. Chapter 2 introduces the related works, describes the system models, and presents some notations that will be used throughout this dissertation.

The first major part of this dissertation consists of the following chapters: Chapter 3, Chapter 4, and Chapter 5. This part is dedicated to study how to secure cooperative ad hoc network against insider attacks under noise and imperfect monitoring. Specifically, Chapter 3 studies how to defend against routing disruption attacks [93], Chapter 4 investigates how to defend against injecting traffic attacks [90], and Chapter 5 provides formal analysis of securing cooperative ad hoc networks under noise, imperfect monitoring, and insider attacks [89].

The second major part of this dissertation consists of Chapter 6 and Chapter 7. This part studies how to design attack-resistant cooperation strategies for autonomous ad hoc networks that can work well in noisy and hostile environments. Specifically, Chapter 6 presents an attack-resistant cooperation stimulation system for autonomous mobile ad hoc networks [91], and Chapter 7 focuses on the game theoretic analysis of cooperation and security in autonomous ad hoc networks [92].

Finally, Chapter 8 concludes this dissertation and presents future directions.

Chapter 2

Background

2.1 Related Works

2.1.1 Security in Wireless Ad Hoc Networks

In wireless ad hoc networks, since all nodes share the common communication medium, attackers can easily launch a variety of attacks ranging from passive eavesdropping from active interfering. For example, a simple and straight-forward attack is jamming attack, where attackers can disrupt the other nodes' normal communications by introducing interferences. Various schemes have been proposed to handle jamming attack in the literature. One way to handle jamming attack is to design robust physical layer technologies, such as spread spectrum, which are resistant to RF jamming [71, 72, 77]. By using some spreading codes only known to the communicating peers, nodes have created a secret channel among them. Recently, several new approaches have also been proposed to handle jamming attacks in a more efficient way, such as those proposed in [20, 55, 81, 83, 84]. In this dissertation we will not focus on jamming attack, and will assume that some

existing schemes, such as those proposed in [71,83], have been employed to address such attacks.

Besides physical layer attacks, attackers can also try to interrupt the normal Medium Access Control (MAC) layer behaviors, such as described in [53, 74]. In this dissertation, we will not focus on specific types of MAC layer attacks. Instead, we will focus on some general attack models which have incorporated the effects of MAC misbehavior. In [22], Cagalj et. al. have also studied the possible MAC layer selfish and cheating behaviors in wireless CSMA/CA networks.

To secure wireless ad hoc network, we can first try to prevent attackers from entering the networks. This can be achieved by applying necessary access control and authentication [42, 101], such as secure key distribution [8, 21, 43] and secure neighbor discovery [34, 36], etc. For example, Zhou and Haas investigated distributed certificate authorities in ad hoc networks using threshold cryptography [101]. Hubaux et al. developed the idea of self-organized public-key infrastructure similar to PGP in the sense that public-key certificates are issued by the users [42, 43]. The difference with PGP is that in their system, certificates are stored and distributed by the users. Capkun et al. have also discussed how to build security associations with the help of mobility in mobile ad hoc networks [24].

Since in ad hoc network nodes relies on each other to forward packets, routing has become one of the most active research topics during the last decade, and various routing protocols have been proposed, such as DSR [47, 48], AODV [70], OLSR [25], and TBRPF [63]. Some performance comparison among various routing protocols have been demonstrated in [2]. However, in order to work properly, these protocols need trusted working environments, while in reality the environments is usually adversarial. Some examples of routing attacks are: *black hole*,

gray hole, *wormhole*, *rushing attack*, and *frame-up* [34–37]. For example, the attackers can create a wormhole through collusion in the network to short circuit the normal flow of routing packets [35], or can apply rushing attack to disseminate route request quickly through the network [36]. By creating a wormhole or applying rushing attacks, the attackers can prevent good routes from being discovered, and increase their chance of being on discovered routes. Once an attacker is on a certain route, it can create a black hole by dropping all the packets passing through it, or create a gray hole by selectively dropping some packets passing through it. If the protocols have the mechanism to track malicious behavior, an attacker can also try to frame up good nodes. In addition, an attacker can modify the packets passing through it, which has similar effects as dropping packets, but a little bit more severe because more network resources will be wasted when the following nodes on this route continue forwarding this corrupted packet.

In the literature, various secure routing protocols have been proposed, such as [3, 9, 12, 13, 17, 23, 27, 34–39, 58, 62, 65, 66, 68, 76, 86, 87, 94, 95, 98]. For example, Papadimitratos and Haas [65] have proposed a secure routing protocol for mobile ad hoc networks that guarantees the discovery of correct connectivity information over an unknown network in the presence of malicious nodes. Sanzgiri et al [76] have considered a scenario that nodes authenticate routing information coming from their neighbors while not all the nodes on the route will be authenticated by the sender and the receiver. Hu, Perrig and Johnson [34] have proposed Ariadne, a secure on-demand ad hoc network routing protocol, which can prevent attackers or compromised nodes from tampering with uncompromised routes that (only) consist of uncompromised nodes. In [35, 36], they have described how to defend against rushing attacks through secure neighbor discovery and how to apply packet leashes

to defend against wormhole attacks. Later, Capkun and Hubaux have investigated secure routing in ad hoc networks in which security associations exist only between a subset of all pairs of nodes [23].

However, most of the existing secure routing schemes have focused on preventing illegitimate nodes from being on the routes. In other words, they have focused on defending against outside attackers. In ad hoc networks, due to the loose access control and weak physical protection, insider attackers can be very common. In the literature, very few schemes have considered insider attacks. Among them the most representative one is proposed by Marti et al [58]. They focused on the case that nodes agree to forward packets but fail to do so, and proposed two tools that can be applied upon source routing protocols: *watchdog* and *pathrater*. Specifically, each node launches a "watchdog" to monitor its neighbors' packet forwarding activities and to make sure that these neighbors have forwarded the packets according to its requests. Pathrater will be used to prevent misbehaving nodes from being on the selected routes when performing route discovery. However, this system suffers some problems, and many attacks can cause a malicious behavior not being detected, such as ambiguous collisions, receiver collisions, limited transmission power, collusion, and partial dropping. Meanwhile, due to noise and possible attacks, good nodes can also be easily marked as malicious. In other words, the proposed scheme may suffer both high false alarm ratio and high miss detect ratio when performing attacker detection.

Following [58], CONFIDANT was proposed to detect and isolate misbehaving node and thus make it unattractive to deny cooperation [16]. Comparing to the schemes proposed in [58], CONFIDANT allows the reputation to propagate throughout the network. However, since the scheme still rely on watchdog, they

also suffer the same types of problems as [58]. Furthermore, once reputation is allowed to propagate, attackers can also collude to frame up or blackmail other nodes [15]. Besides [16,58], Ning and Sun have also provided a case study of insider attacks against mobile ad hoc routing protocols by focusing on AODV.

Security in ad hoc networks has also been addressed from the intrusion detection point of view, such as [40,41,96,97]. In these works, the authors have discussed how to apply intrusion detection techniques to secure wireless ad hoc networks. They examined the vulnerabilities of a wireless ad hoc network, then introduced multi-layer integrated intrusion detection and response mechanisms. Such techniques can also be used to deal with insider attacks. However, in their work they have not described specific mechanisms to secure ad hoc networks. Furthermore, no formal analysis of securing ad hoc networks against insider attacks has been provided.

Besides the above mentioned attacks, attackers can also launch various types of other attacks to disrupt the normal communications. For example, one severe attack is Sybil attack [29, 61], where an attacker can behave as if it were a larger number of nodes, for example by impersonating other nodes or simply by claiming false identities. In [3] the authors have also studied JellyFish attacks. Another types of severe attacks, which will be thoroughly studied in this dissertation, is injecting traffic attacks, that is, the attackers will try to inject an overwhelming amount of traffic into the network to consume valuable network resources and degrade the network performance. Section 2.3 describes the attack model considered in this dissertation.

2.1.2 Cooperation Stimulation in Ad Hoc Networks

In the literature, many schemes have been proposed to address the issue of cooperation stimulation in ad hoc networks [6,7,16,18,19,26,31,58–60,78,80,99,100]. One way to stimulate cooperation among selfish nodes is to use payment-based methods, such as those proposed in [7,18,19,99,100]. In [18], a cooperation stimulation approach was proposed by using a virtual currency, called nuglets, as payments for packet forwarding, which was then improved in [19] using credit counters. However, tamper-proof hardware is required in each node to count the credits. In [99], Sprite was proposed to stimulate cooperation. It releases the requirement of tamper-proof hardware, but requires a centralized credit clearance service trusted by all nodes. Furthermore, these schemes consider only nodes' selfish behavior, while in many situations nodes can be malicious. Payment-based cooperation stimulation mechanisms have also been proposed in [7,100]. Although these schemes can effectively stimulate cooperation among selfish nodes, the requirement of tamper-proof hardware or central billing services greatly limits their potential applications.

Another way to stimulate cooperation among selfish nodes is to use reputation-based methods with necessary monitoring [58,60]. Actually, the watchdog mechanism proposed in [58] can also be regarded as a reputation-based cooperation stimulation scheme. Following [58], Core has been proposed to enforce cooperation among selfish nodes [60], which uses watchdog as a basic building module. As mentioned before, these schemes suffer some problems. For example, many attacks can cause a malicious behavior not being detected in these schemes, and malicious nodes can easily propagate false information to frame up others. Meanwhile, these schemes can only isolate misbehaving nodes, but cannot actually punish them, and malicious nodes can still utilize the valuable network resources even after be-

ing suspected or detected.

Besides that, efforts have also been made toward mathematically analyzing cooperation in autonomous ad hoc networks by applying game theory, such as [6, 26, 30, 31, 59, 78, 80]. In [78], Srinivasan et. al. provided a mathematical framework for cooperation in ad hoc networks by focusing on the energy-efficient aspects of cooperation. In [30, 31], Felegyhazi et. al. defined a game model and identified the conditions under which cooperation strategies can form an equilibrium. In [59], Michiardi et. al. studied the cooperation among selfish nodes in a cooperative game theoretic framework. In [6], Altman et. al. studied the packet forwarding problem in a non-cooperative game theoretic framework and provide a simple punishing mechanism considering end-to-end performance objective of the nodes. The study of selfish behavior in ad hoc networks has also been addressed in [26,80]. All these schemes consider only selfish behavior and most of them study cooperation enforcement under a repeated game framework. Since these works are highly related to our work, in later chapters the difference between our work and these works as well as the uniqueness of our work will be further demonstrated.

Recently, Cagalj et al have also studied the selfish and cheating behaviors in wireless CSMA/CA networks under a game theoretic framework [20,22]. They have used both cooperative and non-cooperative game theory to model and analyze the co-existence of multiple CSMA/CA selfish users, and proposed a simple channel access protocol that discourages selfish behavior and results in the optimal and fair allocation of the available bandwidth. In their work, besides Nash equilibrium, Pareto optimality and fairness have also been considered when deriving the optimal strategies. However, in their work, they have not considered the situations where some nodes' goals are to harm other specific nodes. Meanwhile, the game model

in [20, 22] is also different from ours presented in Chapter 7 in the sense that the strategy space, utility functions, and solution formats are totally different.

In this dissertation, we have used game theory to analyze the dynamic interactions among different nodes. Roughly speaking, game theory deals with multi-person decision making, in which each decision maker tries to maximize his own utility [32,64]. Game theory has been used to solve various problems in networking and telecommunication applications, such as resource allocation [5,46,57,73,82,85], flow and congestion control [4], routing games [7,50,51,54,75], cooperation enforcement in ad hoc networks [6, 26, 31, 59, 78, 80]. In our work, we mainly focus on the most important concept in game theory: Nash equilibrium[1]. Specifically, we adopt Nash equilibrium as a basic optimality metric to measure the performance of those derived strategies.

2.2 System Description

Now we introduce some basic assumptions of the wireless ad hoc networks networks to be considered. We assume that each node is equipped with a battery with limited power supply, communicates with other nodes through wireless connections, and can move freely inside a certain area when mobile ad hoc networks are considered. In cooperative ad hoc networks, nodes are classified into *good* or *malicious*, while in autonomous ad hoc networks, nodes are classified into *selfish* or *malicious*.

We mainly focus on the scenarios that the wireless links are bidirectional, but not necessarily be symmetric. That is, if node A is capable of transmitting data to node B directly, then node B is also capable of transmitting data to A directly,

[1]A Nash equilibrium is a strategy profile for a game with the property that no player can benefit by changing his strategy while the other players keep their strategies unchanged [64].

though the two directions may have different bandwidths. This assumption holds in most wireless communication systems. In this paper, *neighbor* refers to that two nodes are in each other's transmission range, and can directly communicate with each other. We assume that the MAC layer protocol supports *acknowledgement* (ACK) mechanism. That is, if node A has sent a packet to node B, and B has successfully received it, then node B needs to notify A of the reception immediately.

In this dissertation we will mainly focus on source routing, where *source routing* means that when sending a packet, the source lists in the packet header the complete sequence of nodes through which the packet is to traverse. In general, due to the multihop nature, when a node wants to send a packet to a certain destination, a sequence of nodes will usually be requested to help forwarding this packet. We refer to the sequence of ordered nodes as a *route*, the set of intermediate nodes on a route as *relays*, and the procedure to discover a route as *route discovery*. In general, the route discovery can be partitioned into three stages. In the first stage, the requester notifies other nodes in the network that it wants to find a route to a certain destination. In the second stage, other nodes in the network will make their decisions on whether they will agree to be on the discovered route. In the third stage, the requester will determine which route should be used.

Without otherwise explicit specification, we will use DSR [47] as the underlying routing protocol. There are two basic operations in DSR: *route discovery* and *route maintenance*. In DSR, when a source S wishes to send packets to a destination D but does not know any routes to D, S will initiate a route discovery by broadcasting a ROUTE REQUEST packet, specifying the destination D and a unique ID. When a node receives a ROUTE REQUEST not targeting on it, it first checks whether this request has been seen before. If yes, it will discard this packet, otherwise,

it will append its own address to this REQUEST and rebroadcasts it. When the REQUEST arrives at D, D then sends a ROUTE REPLY packet back to S, including the list of accumulated addresses (nodes). A source may receive multiple ROUTE REPLYs from the destination, and can cache these routes in its Route Cache. Route Maintenance handles link breakages. If a node detects the link to the next hop is broken when it tries to send a packet, it will send a ROUTE ERROR packet back to the source to notify this link breakage. The source then removes the route having this broken link from its Route Cache. For subsequent packets to the destination, the source will choose another route in its Route Cache, or will initiate a new Route Discovery when no route exists.

As we have mentioned before, in this dissertation we will focus on defending against insider attacks. We assume that each node has a unique and verifiable identity, such as a public/private key pair, and there is a secure binding between a node's public key and its address. We also assume that a node can know or authenticate other nodes' public keys, but no node will disclose its private key to others unless it has been compromised. We do not assume that nodes trust each other, since some nodes may be malicious or be compromised.

In [14], Bobba et. al. have studied how to establish security associations between pairs of nodes in mobile ad hoc networks without relying on any trusted security service. Specifically, they have shown how to bootstrap security for the routing layer, and have used the notion of statistically unique and cryptographically verifiable (SUCV) identifiers to implement a secure binding between IP addresses and keys. They have also demonstrated that the solution is applicable to various routing protocols, such as Ariadne [34] and SEAD [37]. In our work, we assume the scheme proposed in [14] will be applied to bootstrap security association

among nodes.

In ad hoc networks, in general not all packet forwarding decision can be perfectly executed. For example, when a node has decided to help another node to forward a packet, the packet may still be dropped due to link breakage or the transmission may fail due to channel errors. In this dissertation we refer to those factors that may cause decision execution error as *noise*, which include environmental unpredictability and system uncertainty, channel noise, mobility, etc. When necessary, we will use p_e to denote the *average* packet dropping probability due to noise. It is worth mentioning that the packet dropping probability may vary over time due to the varying channel conditions, mobility, etc.

We also assume that some underlying monitoring schemes have been employed (such as those proposed in [58] and those described in Chapter 3, Chapter 4, and Chapter 6) which can let the source know whether its packets have been successfully delivered to their destinations. Meanwhile if a packet has been dropped by some relay, the underlying monitoring mechanism can let the source know who has dropped this packet. However, we do not assume any perfect monitoring, instead, we assume that even a node has successfully forwarded a packet, with probability no more than p_f it can be observed as dropping packet (i.e., false alarm). On the other hand, when a packet has been dropped by a certain relay, with probability no more than p_m this can be observed as a forwarding event (i.e., miss detect). Here p_f and p_m characterize the capability of the underlying monitoring mechanism. It is easy to understand that p_f and p_m may vary according to the underlying monitoring mechanism and the monitoring environment.

When evaluating the performance of proposed strategies, besides theoretical analysis, we have also conducted various simulations. In our simulations, we use

an event-driven simulator to simulate mobile ad hoc networks. The physical layer assumes a fixed transmission range model, where two nodes can directly communicate with each other successfully only if they are in each other's transmission range. The MAC layer protocol simulates the IEEE 802.11 Distributed Coordination Function (DCF) [44]. DSR will be used as the underlying routing protocol. When considering mobile ad hoc networks, we assume that each mobile node moves according to the *random waypoint* model [88], which can be characterized by the following three parameters: a node starts at a random position, waits for a duration called the *pause time* that is modeled as a random variable with exponential distribution, then randomly chooses a new location and moves towards the new location with a velocity uniformly chosen between v_{min} and v_{max}. When it arrives at the new location, it waits for another random pause time and repeats the process.

2.3 Attack Model

Next we exploit the possible attacks that can be launched in such networks. We say a route $R =$ "$R_0 R_1 \ldots R_M$" is *valid at time t* if for any $0 \leq i < M$, R_i and R_{i+1} are in each other's transmission range. We say a link (R_i, R_{i+1}) *is broken at time t* if R_i and R_{i+1} are not in each other's transmission range. It is easy to see that at time t, a packet can be successfully delivered from its source S to its destination D through the route $R =$ "$R_0 R_1 \ldots R_M$" with $R_0 = S$ and $R_M = D$ within the delay constraint τ if and only if all of the following conditions are satisfied:

1. R is a valid route at time t, and no links on route R are broken during the transmission.

2. No errors have been introduced to the packet.

3. No intermediate nodes (including S) on route R will drop the packet.

4. The total transmission time is less than τ.

In order to degrade the network performance, the attackers can either directly break the ongoing communications, or try to waste other nodes' valuable resources. Based on the above analysis we can see that from the attackers' point of view, the following attacks can be used:

A1. *Emulate link breakage:* When a node R_i wants to transmit a packet to the next node R_{i+1} on a certain route R, if R_{i+1} is malicious, R_{i+1} can simply keep silent to let R_i believe that R_{i+1} is out of R_i's transmission range, which can dissatisfy the condition 1.

A2. *Drop/modify/delay packets:* Dropping a packet can dissatisfy the condition 3, modifying a packet can dissatisfy the condition 2, and delaying a packet can dissatisfy the condition 4.

A3. *Prevent good routes from being discovered*: Such attacks can either dissatisfy the condition 1, or increase their chance of being on the discovered routes and then launching various attacks such as A1 and A2. Two examples are wormhole and rushing attacks [35, 36].

A4. *Inject traffic*: Malicious nodes can inject an overwhelming amount of packets to overload the network and consume other nodes's valuable energy. When other nodes forward these packets but cannot get payback from attackers, the consumed energy is wasted.

A5. *Collusion attack*: Attackers can work together in order to improve their attacking capability.

A6. *Slander attack*: Attackers can also try to say something bad about the others.

A7. *Impersonation*: Attackers can also try to impersonate good nodes to achieve their various malicious goals, such as causing other nodes to believe these good nodes are malicious.

A8. Sybil attack: An attacker can behave as if it were a larger number of nodes, for example by impersonating other nodes or simply by claiming false identities [29, 61].

Among these inside attacks, A1, A2, and A3 can be regarded as the specific types of routing disruption attacks, A4 can be regarded as resource consumption attacks, while A5, A6, A7, and A8 can be regarded as auxiliary methods to further improve attackers' capabilities.

Chapter 3

Secure Ad Hoc Networks Against Routing Disruption Attacks

In this chapter we investigate how to defend cooperative ad hoc networks against routing disruption attacks, that is, attackers attempt to cause legitimate data packets to be routed in dysfunctional ways, and consequently cause packets to be dropped or extra network resources to be consumed. To defend against such attacks, we have designed a set of light-weight mechanisms with low overhead. First, each node launches a *route traffic observer* to monitor the behavior of each valid route in its route cache, and to collect the packet forwarding statistics submitted by the nodes on this route. Since malicious nodes may submit false reports, each node also keeps *cheating records* for other nodes. If a node is detected as dishonest, this node will be excluded from future routes, and the other nodes will stop forwarding packets for it. Third, each node will try to build *friendship* with other nodes to speed up malicious node detection. *Route diversity* will also be explored by each node to discover multiple routes to the destination, which can increase the chance of defeating malicious nodes who aim to prevent good routes from being

discovered. In addition, *adaptive route rediscovery* will be applied to determine when new routes should be discovered.

This chapter is organized as follows. Section 3.1 describes the proposed mechanisms in detail. Section 3.2 analyzes the security of the proposed mechanisms. Section 3.3 presents the simulation results and performance evaluation. Finally, Section 3.4 summarizes this chapter.

3.1 Defense Mechanism Description

Before describing the detail of proposed mechanisms, which is referred to as HADOF (the acronym of Honesty, Adaptivity, Diversity, Observer, and Friendship), we first introduce some notations, as listed in Table 3.1. We focus on insider attacker, and assume that all nodes in the network are legitimate. Meanwhile, if two nodes set up communication between them, they must have built a trust relationship, and trust the information reported by each other. This trustiness can be built outside of the context of the network (e.g. friends), or through certain authentication mechanisms after the network has been set up. In this chapter we use S to denote the source and D to denote the destination, and use *traffic pair* to refer to a pair of nodes (S, D) communicating with each other directly or indirectly.

3.1.1 Route Traffic Observer

In HADOF, each node launches a route traffic observer (RTO) to periodically collect the traffic statistics of each valid route in its route cache. Here a *valid route* refers to a route without receiving any link breakage report. At the end of each pre-determined interval, the RTO examines each traffic pair (S, D) and each route

Table 3.1: Notations for each traffic pair

R_i	The i^{th} available route from S to D in S's Route Cache.
L_i	Number of intermediate nodes on the route R_i.
$FN_{cur}(A, S, R_i)$	The number of packets originated from S and forwarded by A via route R_i in this interval.
$RN_{cur}(A, S, R_i)$	The number of packets originated from S and received by A via route R_i in this interval.
$FN_{tot}(A, S)$	The total number of packets originated from S and forwarded by A.
$RN_{tot}(A, S)$	The total number of packets originated from S and received by A.
$P_{cur}(A, S, R_i)$	$\frac{FN_{cur}(A,S,R_i)}{RN_{cur}(A,S,R_i)}$, the packet delivery ratio of A for S via route R_i in this interval.
$P_{avg}(A, S)$	$\frac{FN_{tot}(A,S)}{RN_{tot}(A,S)}$, the overall packet delivery ratio of A for S.
H(A, S)	A's honesty score in S's point of view.

R_i to D in S's route cache that has been used in this interval. In particular, the RTO collects $RN_{cur}(A, S, R_i)$ and $FN_{cur}(A, S, R_i)$ reported by each node A on this route. This can be done by letting D periodically send back an agent packet to collect such information, or letting each node periodically report its own statistics to S. For each node A known by S, S's RTO also keeps a record of $RN_{tot}(A, S)$ and $FN_{tot}(A, S)$. To reduce overhead, the RTO of S will request reports from the intermediate nodes of a route only when S realizes that some packets have been dropped on this route in this interval based on the reports submitted by D.

After the RTO has finished collecting packet forwarding statistics, it recalculates the expected quality of those routes that have been used in this interval. In general, the expected route quality is affected by many factors, such as the forwarding history of each node on this route, the hop number, the current traffic load

and traffic distributions, etc. Before defining the expected route quality metric, we first define the expected packet delivery ratio of A for S, $P(A, S)$, as follows:

$$P(A, S) = (1 - \beta)P_{avg}(A, S) + \beta P_{cur}(A, S, R_i). \tag{3.1}$$

That is, $P(A, S)$ is a weighted average of $P_{cur}(A, S, R_i)$ and $P_{avg}(A, S)$, and β is used to adjust the weight between them. The intuition behind this is that when predicting a node's future performance, we consider not only this node's current performance, but also its past history. It is easy to see that the range of $P(A, S)$ is between 0 and 1. In HADOF, the expected route quality $Q(R_i)$ for route R_i is calculated as follows:

$$Q(R_i) = \prod_{A \in R_i} P(A, S) * H(A, S) - \lambda * L_i, \tag{3.2}$$

where $H(A, S)$ is A's honesty score in S's view indicating the suspicious degree of A. $H(A, S)$ ranges from 0 to 1, with 1 indicating being honest and 0 indicating being malicious. The criteria of calculating $H(A, S)$ is presented in Section 3.1.2. In (3.2), a small positive value λ is introduced to account for the effects of hop number. As a result, if two routes have the same value for the product in the right hand of (3.2), the route with less hops is favored. The intuition behind this is that we expect a route with less hops having less influence on the network. In HADOF, the values of $P(S, S)$, $P(D, S)$, $H(S, S)$ will always be 1, since a source trusts itself and the corresponding destination.

3.1.2 Cheating Record and Honesty Score

When S's RTO collects packet forwarding statistics, malicious nodes may submit false reports. For example, it may report a smaller RN value and a larger FN value to cheat the source and frame up its neighbors. To address this, each source

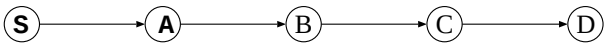


Figure 3.1: Detection of cheating behavior

keeps a *Cheating Record* (CR) database to track whether some nodes have ever submitted or been suspected to submit false reports to it. S will mark a node as malicious if S has enough evidence that the node has submitted false reports.

Initially, S assumes that all nodes are honest, and sets the honesty score $H(A, S)$ for each node A to be 1. After each report collection which is performed periodically, S will try to detect whether some nodes on a route are cheating through checking the *consistence* of the received reports. For example, in Fig. 3.1, both A and B are on the route R with A being ahead of B. A cheating behavior is detected if S finds that $FN_{cur}(A, S, R) \neq RN_{cur}(B, S, R)$. If one of them (A or B) is trusted by S (e.g., that node is S itself or D), then the other node can be marked as cheating by S, and the honesty score of the cheating node will be set to be 0. Otherwise, S can only suspect that at least one of them is cheating. In this case, the honesty scores of both nodes are updated as

$$H(A, S) = \alpha H(A, S) \tag{3.3}$$

$$H(B, S) = \alpha H(B, S) \tag{3.4}$$

where $0 < \alpha < 1$ is used to indicate the punishment degree. In addition, if $FN_{cur}(A, S, R) > RN_{cur}(B, S, R)$, S will reset the value of $FN_{cur}(A, S, R)$ using $RN_{cur}(B, S, R)$, reset the value of $RN_{cur}(B, S, R)$ using $FN_{cur}(B, S, R)$, and recalculate $FN_{tot}(A, S)$ and $RN_{tot}(B, S)$ using the updated values. Since it makes no sense that $FN_{cur}(A, S, R) < RN_{cur}(B, S, R)$, we will not consider this situation.

Once a node has been detected as cheating, punishment should be applied on it. In HADOF, when S detects a node B being malicious, S will put B in its blacklist

(equivalent to set $H(B,S)$ to be 0), stop forwarding any packets originated from B, and refuse to be on the same route as B in the future.

Next we introduce a mechanism to recover the honesty scores of nodes that have been framed up by malicious nodes. We still use the example in Fig. 3.1 to illustrate this mechanism. When S finds the reports submitted by A and B conflicting with each other, that is, $FN_{cur}(A,S,R) > RN_{cur}(B,S,R)$, besides decreasing A's honesty score, S will also increase the number of possible frame-up attacks launched by B to A, and records the difference between $FN_{cur}(A,S,R)$ and $RN_{cur}(B,S,R)$. Similarly, S does the same thing to B. If later S detects that B is a cheating node, S will check how many nodes have ever been framed up by B and for each node how many times. Assume A has been framed up by B m times, S will recover A's honesty score as follows:

$$H(A,S) = \frac{H(A,S)}{\alpha^m}, \tag{3.5}$$

which is always bounded by 1. Meanwhile, S also needs to increase $FN_{tot}(A,S)$ or decrease $RN_{tot}(A,S)$ to recover the inaccuracy caused by frame-up attacks launched by B.

3.1.3 Friendship

Since a malicious node knows the source and destination of each route that it is on, to avoid being detected, it will only frame up its neighbors who are neither the source nor the destination. Therefore, even when the CR database has been activated, the malicious nodes can only be suspected, but cannot be proved as cheating by the source. This can be mitigated by taking advantage of the existing trustiness relationship. Each node maintains a private list of trust nodes that it

considers to be honest. Now if B submits false reports to S to frames up A, while S trusts A, B can be detected by S immediately, and $H(B, S)$ will be set to be 0.

3.1.4 Route Diversity

Since there may exist more than one route from a source to a destination, it is usually beneficial to discover multiple routes. In [2, 67], the authors have shown that using multiple routes can reduce the route discovery frequency. In this chapter, we investigate how route diversity can be used to defend against routing disruption attacks. In DSR, discovering multiple routes from a source to a destination is straight-forward. Let $MaxRouteNum$ be the maximum number of ROUTE REPLYs that the destination can send back for the route requests with the same request ID. By varying $MaxRouteNum$, we can discover different number of routes. By exploring route diversity, we have better chance to defeat attackers who aim to prevent good routes from being found. Meanwhile, since there may exist multiple routes, the source can always use the route with the best quality according to certain criteria.

When a new route R is discovered, for each node A on this route, $FN_{cur}(A, S, R)$ and $RN_{cur}(A, S, R)$ should be initialized to be 0. Since this route has never been used before, its expected quality can be calculated as

$$Q(R) = \prod_{A \in R} P_{avg}(A, S) * H(A, S) - \lambda * L. \tag{3.6}$$

The difference between (3.6) and (3.2) lies in that only nodes' past history on the route are used in (3.6).

Since there may exist multiple routes to D in S's Route Cache, S needs to decide which route should be used. One possible way is to always use the one with the best expected quality. However, this may not be the best choice. For

example, the quality of a route may degrade dramatically after being injected into a lot of traffics. In this chapter, the following procedure is used to distribute traffics among multiple routes, and adaptively determine which route should be used. Let $Q_{threshold}$ be a pre-determined quality threshold, and let $R_1, \ldots, R_K$ be the K routes with the expected quality higher than $Q_{threshold}$. Once S wants to send a packet to D, S randomly picks a route among them. The probability that route R_i $(1 \leq i \leq K)$ will be picked is determined as

$$Prob(R_i) = \frac{Q(R_i)}{Q(R_1) + \cdots + Q(R_K)} \tag{3.7}$$

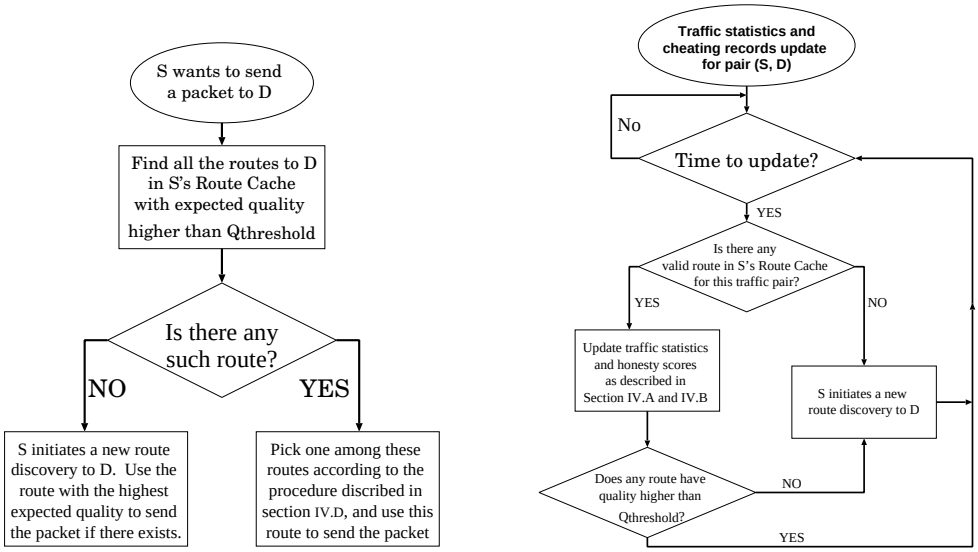
If no route has expected quality higher than $Q_{threshold}$, the route with the highest expected quality will be selected.

3.1.5 Adaptive Route Rediscovery

Due to mobility and the dynamically changing traffic patterns, some routes may become invalid after a while, or their quality may change. Usually, a new route discovery should be initiated by S when there exist no available routes from S to D. In this chapter, we use an *adaptive route rediscovery* mechanism to determine when a new route discovery should be initiated: if S wants to send packets to D, and there exist no routes to D with quality higher than $Q_{threshold}$ in S's route cache, S then initiates a new route discovery.

3.1.6 Implementation

We have implemented HADOF upon DSR, which includes two major procedures: packet sending procedure and traffic statistics and cheating records updating procedure. The packet sending procedure is described in Fig. 3.2(a). When S wants to



(a) Packet sending procedure (b) Statistics update procedure

Figure 3.2: HADOF implementation

send a packet to D, S first checks its route cache to find whether there exist valid routes to D. If there exist no valid routes, S initiates a new route discovery with the destination being D. If there exist some valid routes, but none has expected quality higher than $Q_{threshold}$, S picks the route with the best expected quality, and initiates a new route discovery. Otherwise, S randomly picks one route according to the procedure described in Section 3.1.4.

The procedure for updating/maintaining traffic statistics and cheating records is described in Fig. 3.2(b). The source S periodically calls this procedure to collect traffic statistics for each route that has been used in this interval. Based on the mechanisms described in Section 3.1.1 and Section 3.1.2, S updates the expected route quality and cheating records. If necessary, a new route discovery should be initiated when certain conditions are satisfied, as described in Section 3.1.5.

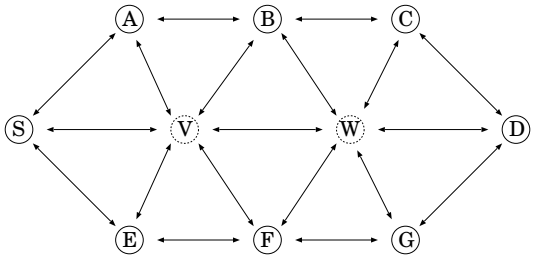


Figure 3.3: A simple example

3.2 Security Studies

This section analyzes the security aspects of HADOF in terms of defending against various routing disruption attacks. Throughout this section, we will use Fig. 3.3 as a simple example to illustrate different situations.

Black Hole and Gray Hole Attacks: In HADOF, the source can quickly detect a gray hole or black hole based on the reports it has collected and past records of each node. Without loss of generality, assume B has created a gray hole on route "SABCD" in Fig. 3.3. Based on the reports submitted by A, B, C, and D, S can know that some of them have dropped packets. Node B can be detected as creating a black/gray hole by S if $P_{avg}(B, S)$ and $P_{cur}(B, S, \text{“}SABCD\text{”})$ are low, and $RN(B, S)$ value is larger than a pre-defined threshold, where a relatively large $RN(A, S)$ is used to make sure that this is not transient phenomenon.

Frame-up Attacks without Collusion: Besides dropping packets, a malicious node can also submit false reports to cheat the source and frame up its neighbors. For example, on the route "SABCD", if B is malicious, B can submit a smaller RN value to frame up A and a larger FN number to frame up C. In HADOF, a source can detect frame-up attacks through checking the consistence of the reports it has collected. We still use the route "SABCD" as an example, and assume that the malicious nodes work alone. If B has reported a larger $FN_{cur}(B, S, R)$ to

frame up C, S can detect this by finding $FN_{cur}(B, S, R) > RN_{cur}(C, S, R)$ where R denotes the route "SABCD". Now we analyze the possible consequence of this frame-up. First, B cannot increase its $P_{cur}(B, S, R)$ and $P(B, S)$ since S will use $RN_{cur}(C, S, R)$ to replace $FN_{cur}(B, S, R)$. Second, B can only make S suspect C, but cannot make S believe that C is malicious. Third, if C is trusted by S, then B can be detected immediately, and will be excluded from any route originated from S in the future. Fourth, B's own honesty score will be decreased. Therefore, B can cause only limited damage by framing up others, but has to take the risk of being detected as malicious, especially when friendship has been introduced.

Frame-up Attacks with Collusion: Next we show that collusion in frame-up attacks cannot further deteriorate the situation. We still use the route "SABCD" as an example. In the first case, the malicious nodes are neighbors of each other. For example, B and C. Without loss of generality, we can view them as one node B′, with $RN_{cur}(B', S, R) = RN_{cur}(B, S, R)$ and $FN_{cur}(B', S, R) = FN_{cur}(C, S, R)$. That is, B and C together have the same effects as B′ working alone, and the only difference is that they can release one node by sacrificing of the other one, that is, by letting it take all the responsibilities. In the second case, the malicious nodes are not neighbors of each other. For example, A and C are malicious and work together to frame up B. It can be seen that the effect of A and C jointly framing up B is the same as that of A and C framing up B independently. Thus we conclude that in HADOF collusion cannot further improve the capability of frame-up attacks.

Rushing Attacks: In rushing attacks, an attacker can increase its chance of being on the route by disseminating ROUTE REQUESTs quickly and suppressing any later legitimate ROUTE REQUESTs [36]. For example, in Fig. 3.3, if V can

broadcast the ROUTE REQUESTs originated from S more quickly than A and E, then all the ROUTE REQUESTs broadcasted by A and E will be ignored. The direct consequence is that V appears on all the routes returned by D. Later V can drop packets and frame up its neighbors. Now we show how rushing attacks can be handled using HADOF. If S detects that no routes to D in its route cache work well, it will check whether these routes share a critical node where all packets from S to D pass through it. In this example, the critical node is V. If V has low $P_{avg}(V, S)$ value and low $H(V, S)$, S has reasons to suspect that V has launched rushing attacks. S then initiates a new route discovery and explicitly exclude V from being on discovered routes.

Wormhole Attacks: A pair of attackers can create a wormhole in the network via a private network connection to disrupt routing by short circuiting the normal flow of routing packets [35]. For example, in Fig. 3.3, if W and V are attackers and have created a wormhole between them, V can quickly forward any ROUTE REQUESTs it receives to W, and let W broadcast them. There are two variations based on whether V and W append their addresses to the REQUESTS. If they append their addresses, they are similar as rushing attackers, and the method discussed above can be used to handle them. The situation becomes more severe if they do not append their addresses. For example, W and V can make S believe that D is its neighbor, and later V can create a black hole to drop all the packets originated from S and targeting D. In HADOF, if S finds no routes returned by D are valid, or S has not received any acknowledgement from D, S has reason to suspect that there exists a wormhole between S and D. S then activates an neighbor discovery techniques such as in [35,36] to prevent attackers from creating wormholes.

In summary, HADOF can handle various routing disruption attacks very well, such as gray hole, black hole, frame-up, and rushing attacks, and wormhole attacks, and is collusion-resistant.

3.3 Simulation Studies

In this set of simulations, 100 nodes are randomly deployed inside a rectangular area of 1000m $\times$ 1000m. The maximum transmission range is 250m. There are 20 traffic pairs randomly generated for each simulation. For each traffic pair, the packet arrival is modelled as a Poisson process, and the average packet inter-arrival time is uniformly chosen between 0.04 and 0.2 second, such that each traffic pair injects different traffic load to the network. The size of each data packet after encryption is 512 bytes, and the link bandwidth is 1 Mbps. Among the 100 nodes, we vary the total number of malicious nodes from 5 to 20. In our implementation, the malicious nodes will submit false reports only when it has dropped packets and this false reports cannot be detected easily. For example, a malicious node will not submit false reports to frame up the sources or the destinations.

In the simulations, we focus on mobile ad hoc networks, and each node moves randomly according to a *random waypoint* model, and two sets of average pause time are used: 0 second and 50 seconds. The average pause time of 0 second represents a high mobility case where nodes keep moving, while the average pause time of 50 seconds represents a moderate mobility case. We set $\alpha = 0.9$, $\beta = 0.6$, $\lambda = 0.02$, and $Q_{threshold} = 0.8$. The maximum number of returned routes is set 5 and the maximum number of hops per route is set 10.

In our simulations, the baseline system is implemented as follows: the baseline DSR is used, and for each route discovery, only one route is returned. No adaptive

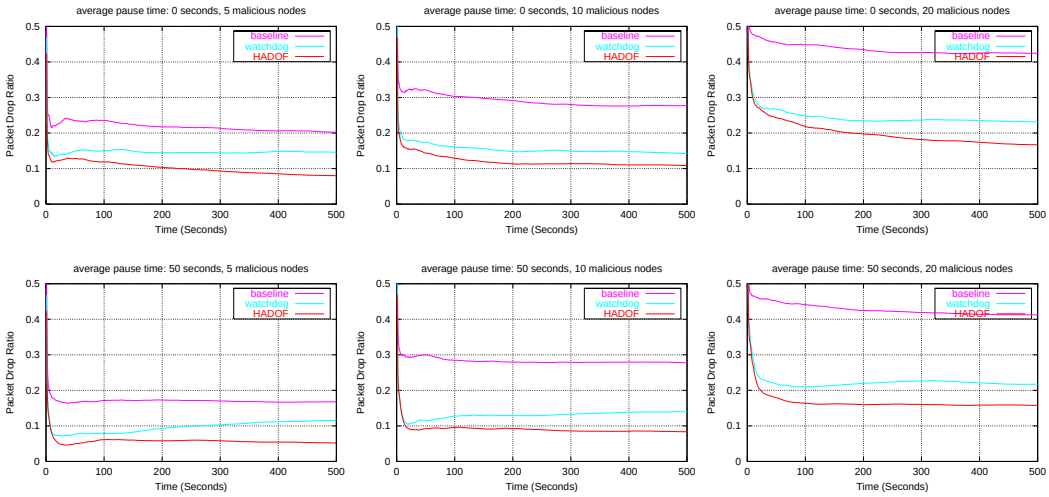


Figure 3.4: Packet drop ratio comparisons under gray hole attacks

route rediscovery is used, and no malicious node detection mechanisms are applied. It is expected that the baseline system will perform badly in most situations.

For comparison, the mechanism proposed in [58] has also been implemented, which includes two major components: watchdog and pathrater. To make watchdog work properly, we have modified the MAC layer protocol to ensure the following property: after node B receives a packet from node A and needs to forward this packet to node C, B can start the forwarding only if both A and C are idle and ready to receive packets. When using watchdog, a node will report to the source when another node refuses to forward more than certain number of packets for it. In our implementation, we set the threshold to be 5. In addition, each route discovery initiated by source S will return at most 5 routes, and the route with the best quality (calculated using pathrater) will be used. When the route in use becomes invalid due to link breaks, instead of using the routes in S's Route Cache, S will initiate a new route discovery. The reason is that with a very high probability those routes may also not work or may work badly due to mobility and traffic dynamics. The SSR (Send extra Route Request) extension has also been

implemented.

The following metrics will be used to evaluate the performance of HADOF.

- Packet drop ratio: The percentage of data packets that have been sent by not been received by the destinations, which equals to 1 minus end-to-end throughput.

- Overhead: In this chapter, we consider *routing overhead*, *energy consumption overhead*, *encryption overhead*, and *complexity overhead*. Given a traffic pattern, routing overhead indicates how many route discoveries have been initiated by the sources. Energy consumption overhead denotes how much extra energy need to be consumed. To keep the confidentiality and integrity of the transmitted content, extra cryptographic operations are needed, and the encryption overhead describes how many extra cryptographic operations are needed by these mechanisms. Complexity overhead accounts for the extra storage and computations needed by applying these mechanisms.

We use "baseline" to denote the baseline system, "watchdog" to denote the system based on watchdog and pathrater, and "HADOF" to denote the system based on HADOF. We use different node movement patterns for each simulation by changing the average pause time and the seed of random number generator. By varying the number of malicious nodes and the average pause time, we get different configurations. For each configuration, the results are averaged over 25 rounds of simulations, where at each round we change the random seed to get different movement and traffic patterns. To make fair comparison, for each configuration and each round of simulation, the same movement and traffic patterns were used by all the three systems.

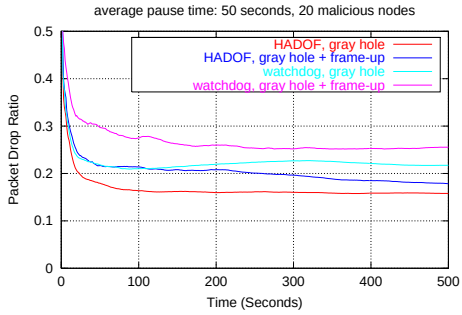


Figure 3.5: Effects of frame-up attacks

3.3.1 Packet Drop Ratio Comparisons

We compare the packet drop ratios of the three systems under different scenarios. First, we compare the packet drop ratios under only gray hole attacks. That is, no nodes will submit false reports. Second, we compare the packet drop ratios under both gray hole and frame-up attacks, where some malicious nodes will drop packets and frame up their neighbors when possible. Third, we show how friendship mechanism can mitigate the effects of frame-up attacks.

Gray hole: In our simulations, we vary the number of malicious nodes from 5 to 20. The gray hole is implemented in such a way that each malicious node drops half of the packet passing through it. The simulation results under different configurations are plotted in Fig. 3.4. From these results we can see that HADOF outperforms watchdog in all situations. For example, under the configuration of pause time 50 seconds, 20 malicious nodes, the packet drop ratio of baseline is more than 40%, watchdog can reduce the packet drop ratio to 22%, while for HADOF, the packet drop ratio is only 16%, that is, more than 33% improvement is obtained over watchdog under this configuration. Under the configuration of pause time 50 seconds, 5 malicious nodes, more than 55% improvement is obtained over watchdog by HADOF.

Gray hole plus frame-up attacks: We investigate the packet drop ratio under both gray hole and frame-up attacks. In HADOF, the only way for a malicious node to frame up a good node is to let the source *suspect* that the good node is cheating. To achieve this, a malicious node can report a smaller RN number than the actual value to frame up the node ahead of it on the route, and/or report a larger FN number than the actual value to frame up the node just following it on the route. However, the malicious node can never make the source *believe* that a good node is cheating, since malicious node cannot create solid evidence.

In watchdog, there exist a variety of ways for a malicious node to frame up good ones. For example, if node A has sent a packet to B and asks B to forward it to C, B has many ways to make A believe that it has sent the packet to C while B did not send packets or intentionally caused transmission failure. As reckoned in [58], many reasons can cause a misbehaving node not being detected, such as ambiguous collisions, receiver collisions, limited transmission power, false misbehavior, collusion, and partial dropping. In our simulations, we only implement the frame-up attacks through receiver collisions. That is, B will forward packet to C only when it knows that C cannot correctly receive it (e.g., C is transmitting data to another node, or receiving data from another node). Since A can only tell whether B has sent the packet to C, but cannot tell whether C has received it successfully, B can easily frame up its neighbors.

Fig. 3.5 shows the simulation results with the configurations of 20 malicious nodes, half of them applying frame-up attacks. First we can see that the degradation of HADOF caused by frame-up attacks is limited. Second, we see that frame-up degrades the performance of both, and affects watchdog more than HADOF. Meanwhile, it is important to point out that we have shown the best-case results for

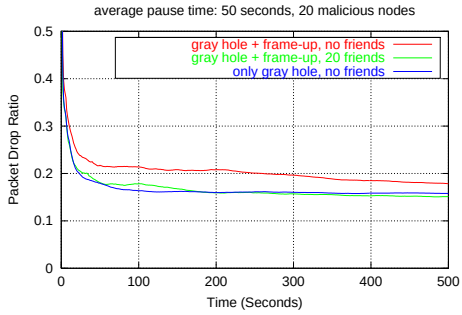


Figure 3.6: Effects of friendship mechanism

watchdog because we have made many assumptions which favor watchdog, such as no collusion attacks, only receiver collisions, perfect MAC protocol. For HADOF, no extra assumptions are needed.

Effectiveness of Friendship: In the previous simulations, friendship has not been introduced and the source only trusts the destination. Next we show the results after introducing friendship mechanism to combat frame-up attacks. We conduct simulations under the situations that each source has 20 friends which are randomly chosen among all good nodes in the network. Fig. 3.6 shows the simulation results using HADOF with the configuration of average pause time 50 seconds, 20 malicious nodes, half of them launching both gray hole and frame-up attacks, and half of them only launching gray hole attacks. From these results we can see that the effects of frame-up attacks can be overcome when trustiness has been established among certain number of users. For example, with 20 friends, the packet drop ratio, which is 15%, is even lower than the situation that no frame-up attacks are launched, which is 16%.

3.3.2 Overhead Comparisons

Routing Discovery Overhead: For each simulation, we have counted the total

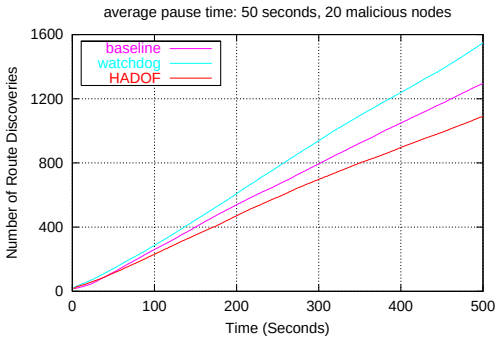


Figure 3.7: Route discovery overhead comparison

number of route discoveries that have been initiated by all the sources. Fig. 3.7 shows the results under the configuration of average pause time 50 seconds, 20 malicious nodes, and only gray hole attacks. From these results we can see that though HADOF needs to initiate route discoveries preventively, it still has the lowest routing discovery overhead. In the baseline system, only one route is returned for each route discovery, which explains why baseline needs to initiate more route discoveries. This also verifies the effectiveness of path diversity. Surprisingly, watchdog has the highest route discovery overhead, which comes from its high false alarm ratio, since a new route discovery will be initiated once no route has average reputation larger than 0.

Energy Consumption Overhead: One major drawback of watchdog is that it consumes much more energy than HADOF, because each node has to keep monitoring its neighbors' transmission activities. We use Fig. 3.1 to illustrate why watchdog needs to consume extra energy. For example, after B sends a packet to C and asks C to forward the packet to D, B has to keep listening C's transmission. If C is a malicious node, C can launch resource consumption attacks to consume B's energy by putting off forwarding packets for B. Even if C is a good node, B still needs to consume extra energy to receive, decode, and compare the packets

transmitted by C with the packets stored in B's buffer. This consumes a lot of extra energy. By requiring nodes to keep monitoring their neighbors, watchdog not only reduces network capacity, but also consumes extra energy. On the other hand, HADOF has no such drawbacks.

Encryption Overhead: As we discussed before, all packets should be encrypted and signed to ensure data confidentiality and integrity. Otherwise, outside attackers can easily intercept those messages through eavesdropping. Compared with the baseline system, HADOF introduces some encryption overhead which comes from encrypting the reports. In most situations only the destination needs to submit reports, and the source and the destination already share a secrete key for data encryption. Thus, the reports from the destination can just be encrypted by this secrete key, which introduces little overhead. In addition, if the amount of data for reporting packet forwarding statistics is much less than the amount of data, which is generally true, the overhead of encrypting reports of intermediate nodes on the route will become negligible compared with data encryption overhead.

Complexity Overhead: In HADOF, each source needs to launch a route traffic observer to maintain and update traffic statistics, and maintain records to track cheating behavior. However, both can be implemented using simple data structures, and consume little memory. The computation overhead comes from updating traffic statistics, route quality, and cheating records. These operations will not introduce a lot computation burden. In watchdog, each node also needs to keep a reputation database and need to calculate route quality. Moreover, each node in watchdog needs to keep an extra buffer to store the packets that it has requested its neighbors to forward but not yet confirmed, which consumes a lot extra memory, and introduces extra computation overhead to compare the packets.

3.4 Summary

In this chapter we proposed HADOF to defend against routing disruption attacks launched by inside attackers, which can be implemented upon the existing source routing protocols. HADOF is capable of adaptively adjusting routing strategies according to the network dynamics and nodes' past records and current performance. It can handle various attacks launched by malicious nodes, such as black hole, gray hole, frame-up, rushing attacks, and wormhole attacks. Moveover, HADOF introduces little overhead to the existing routing protocols, and is fully distributed. Extensive simulation studies have also confirmed the effectiveness of HADOF.

Chapter 4

Secure Ad Hoc Networks Against Injecting Traffic Attacks

Chapter 3 has demonstrated how to defend against routing disruption attacks. In this chapter, we will study another class of powerful attacks: injecting traffic attacks. Specifically, attackers inject an overwhelming amount of traffic into the network in attempt to consume valuable network resources, and consequently degrade the network performance. Since in ad hoc networks, nodes need to cooperatively forward packets for other nodes, such networks are extremely vulnerable to injecting traffic attacks, especially those launched by insider attackers.

In this chapter, we first proposed a set of fully distributed defense mechanisms which can effectively detect injecting data packets attacks, even when attackers can use advanced transmission techniques, such as directional antennas, in attempt to avoid being detected. We have also derived the theoretical upper-bounds for the probability that attackers can successfully launch injecting data packets attacks without being detected. The results show that from attackers' point of view, the best injecting data packets strategy is to conform to their original data packets

injection rate. That is, the best strategy is not to launch injecting data packets attacks. To further decrease the incurred overhead, we have then proposed a centralized defense mechanisms with de-centralized implementation. By letting some nodes under strong protection to perform attack detection, the detection performance can be further improved and the average overhead can be dramatically decreased. Besides injecting data packet attacks, the query-flooding attacks have also been studied and the tradeoff between limiting query rate and system performance is exploited.

This chapter is organized as follows. Section 4.1 describes the system model and investigates the possible types of injecting traffic attacks. Section 4.2 describes the proposed fully distributed defense mechanisms. The theoretical analysis of the proposed distributed defense mechanisms are presented in Section 4.3. In Section 4.4, a centralized detection mechanism with de-centralized implementation is described. Simulation results are presented in Section 4.5. Finally, Section 4.6 summarizes this chapter.

4.1 Injecting Traffic Attacks

Similar as in Chapter 3, in this chapter we also focus on cooperative ad hoc networks, where nodes are classified into two types: *good* and *malicious*. We focus on the scenario that nodes use omnidirectional transmission techniques, such as omnidirectional antennas. However, attackers are allowed to use directional transmission techniques, such as directional antennas [52] or adaptive beamforming [33], to improve their attacking capabilities.

According to the system goal, each node may schedule to generate and send a sequence of packets to certain destinations. We call a source-destination pair to

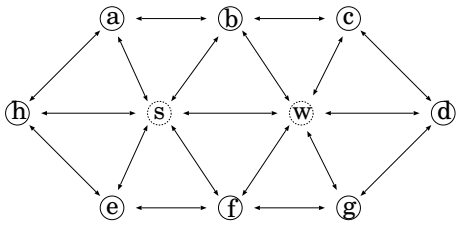


Figure 4.1: An example of long route attack

be *legitimate* if this pair is needed to achieve the system goal. For each legitimate source-destination pair (s, d) in the network, we assume that the number of packets that is scheduled to inject by this pair until time t is $T_{s,d}(t)$. In general, the exact value of $T_{s,d}(t)$ may not be known *a priori* by other nodes in the network. In this chapter we assume that a loose upper-bound of $T_{s,d}(t)$, denoted by $f_{s,d}(t)$, can be estimated by some nodes in the network, which is referred to as the *traffic injection upper-bounds* associated to pair (s, d). Without loss of generality, we simply assume that all data packets have the same size.

As mentioned before, in this chapter our focus is on defending against injecting traffic attacks. Roughly speaking, injecting traffic attacks can be classified into two types: *query-flooding attack* and *injecting data packets attack* (IDPA). Due to the changing topology or traffic pattern, nodes in ad hoc networks may need to frequently perform route updates, which may require broadcasting query messages. Then attackers can broadcast query message with a very high frequency to consume valuable network resources. We call such attack as *query-flooding attack*. Besides query-flooding attacks, attackers can also inject an overwhelming amount of data packets into the network to request other nodes to forward. When other nodes process and forward these packets, their resources (e.g., energy) are wasted. We call such attack as *injecting data packets attack* (IDPA). Since in general the size

of data packet is much larger than the size of query messages, and the injection rate of data packets is much higher than the injection rate of query message, the damage that can be caused by injecting data packets attack is usually more severe than by query-flooding attacks.

We first consider the possible ways that IDPA can be launched by attackers s and d with s being the source and d being the destination. The simplest way, which is called **simple IDPA**, is that s picks a route R to d and injects an overwhelming amount of packets into the network, which is much higher than the legitimate bound $f_{s,d}(t)$.

In the second way, which is called **long route IDPA**, s picks a very long route to inject data packets into the network. For example, as in Fig. 4.1, s can pick the route "swcbahefgd" to send packets from s to d with the number of injected packets conforming to the legitimate bound $f_{s,d}(t)$. By doing this way, s and d can achieve the same effect as increasing its traffic injecting rate.

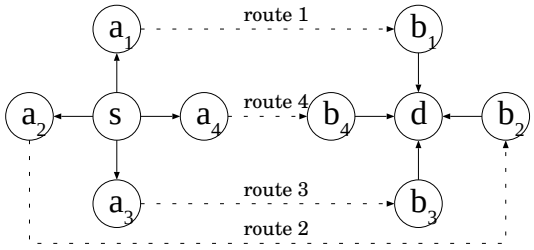


Figure 4.2: An example of multiple route attack

In the third and advanced way, which is called **multiple routes IDPA**, s picks multiple routes to d and simultaneously injects traffic into the network via these routes. For example, as shown in Fig. 4.2, s uses four routes "$sa_1 \ldots b_1 d$", "$sa_2 \ldots b_2 d$", "$sa_3 \ldots b_3 d$" and "$sa_4 \ldots b_4 d$" to inject packets into the network. By doing in this way, the traffic can be distributed among multiple routes such that

for each route the packet injection rate conforms to the legitimate bound $f_{s,d}(t)$, though the total number of injected packets can be much higher than $f_{s,d}(t)$. Moreover, the attackers can also take advantage of advanced transmission techniques, such as directional antenna and beamforming, to avoid being detected.

Besides injecting data packets, attackers can also inject an overwhelming amount of query messages into the network to request other nodes to forward, which is called *query-flooding attacks*. The advantage of query-flooding attacks lies in that for each query, more nodes in the network may be involved to process and forward packets than IDPA. Although query messages are usually much smaller than data packets, when the query frequency is high, the damage can still be severe.

4.2 Defense Mechanisms

In order to detect whether a node has launched injecting traffic attacks, we have to base on the observed behavior of that node. In other words, a node can be marked as launching injecting traffic attacks only if it has been observed by some other nodes that it has injected too much traffic (higher than the legitimate bound), or it has sent traffic to illegitimate destinations. Therefore, the following mechanisms will be required by the defense system:

- A robust packet delivery mechanism where for any packet injected by node x, x cannot deny that this packet is form it and no other nodes can generate the same packet without colluding with x. This is addressed in Section 4.2.1.
- A robust traffic monitoring mechanism to count the number of packets injected by each node in the network. This is addressed in Section 4.2.2.
- A robust detection mechanism to detect injecting traffic attacks based on observed information. This is addressed in Section 4.2.3.

4.2.1 Route Discovery and Packet Delivery

In this chapter we adopt DSR [47] as the underlying routing protocol to perform route discovery. Meanwhile, to defend against possible attacks, the following security enhancements will also be incorporated into the DSR protocol:

- When node s initiates a route discovery to destination d, besides the source destination pair, the query packet should also includes a unique ID associated to this query and the sequence number corresponding to the last data packet that s has sent to d. In this chapter, the following format is used for each query packet:

$$\{s, d, id_s(s,d), seq_s(s,d), sign_s(s, d, id_{s,d}, seq_s(s,d))\}$$

  Here $id_s(s,d)$ is the sequence number of this query packet, which has an initial value of 1 and is required to be increased by 1 after each query has been issued by the pair (s,d). $seq_s(s,d)$ is the sequence number of the last packet that has been injected into the network by the pair (s,d). $sign_s(s,d,id_{s,d},seq_s(s,d))$ is the signature generated by s based on the message $\{s,d,id_{s,d},seq_s(s,d)\}$.

- When a good node x receives a route request packet with s being the source and d being the destination, x first checks whether the following conditions can be satisfied:

  1. the source-destination pair (s,d) is legitimate;
  2. all signatures are valid;
  3. $id_x(s,d) < id_s(s,d)$, where $id_x(s,d)$ is the maximum query sequence number corresponding to the pair (s,d) that x has seen before;

4. $seq_x(s,d) \leq seq_s(s,d)$, where $seq_x(s,d)$ is the maximum data packet sequence number corresponding to the pair (s,d) that x has seen before;
5. no nodes appended to the request packet have been detected as malicious by x;
6. less than T_{maxhop} relay nodes have been appended to the query packet, where T_{maxhop} is a system-level parameter indicating the maximum number of relays that a route can have.
7. x has not forwarded any request for the pair (s,d) in the last $T_x(s,d)$ interval, where $T_x(s,d)$ is the minimum query forwarding interval specified by x to indicate that x will forward at most 1 route request for (s,d) in any $T_x(s,d)$ interval.

If all of the above conditions can be satisfied, we call such a request as a *valid* request. In this situation, x will assign the value of $id_s(s,d)$ to $id_x(s,d)$, assign the value of $seq_s(s,d)$ to $seq_x(s,d)$, append its own address to the route request packet and sign the whole packet, and rebroadcast the new query. If only the conditions from 1 to 4 are satisfied, x will only assign the value of $id_s(s,d)$ to $id_x(s,d)$, assign the value of $seq_s(s,d)$ to $seq_x(s,d)$. In all other situations, x will discard this route request, and perform necessary attacker detection. Assume *request* is the received valid query message that x has decided to forward, then the following format will be used for x to append its own address:

$$\{request, x, sign_x(request, x)\}$$

Once a source has decided to send a packet to a certain destination using a certain route, a data packet delivery transaction should be started. The proposed

data packet delivery mechanism works as follows. Suppose that node s is to send a packet with payload msg and sequence number $seq_s(s,d)$ to destination d through the route R. s first generates two signatures sig_h and sig_b, with sig_h being generated based on the message $\{R, seq_s(s,d)\}$ and sig_b being generated based on the message $\{R, seq_s(s,d), MD(msg)\}$ where $MD()$ is a digest function such as SHA-1 [1]. The format of the packet to be sent is as follows:

$$\{R, seq_s(s,d), sig_h, msg, sig_b\}. \tag{4.1}$$

We refer to $\{R, seq_s(s,d), sig_h\}$ as the *header* of the packet, and refer to $\{msg, sig_b\}$ as the *body* of the packet. Next, s transmits this packet to the next node on route R, and is required to increase the value of $seq_s(s,d)$ by 1. The advantage of generating two signatures will be demonstrated later.

When a node (e.g., x) detects that a certain packet is to be transmitted by a certain node (e.g., y), a first decodes and checks the header of the packet. Assume $\{R, seq_s(s,d), sig_h\}$ is the header of the transmitted packet, a needs to continue receiving and decoding the body of the packet only if all of the following conditions are satisfied:

1. the signature sig_h is valid;
2. x is on the route R and is the target of this transmission;
3. no nodes on route R has been detected as malicious by x;
4. $seq_s(s,d) > seq_x(s,d)$;
5. route R has no more than T_{maxhop} relays;
6. x has agreed to participate on this route before and the route has not expired, where each route will be set an expiration time.

If all of the above conditions are satisfied, x will continue receiving and decoding the body of the packet, assuming it is $\{msg, sig_b\}$. If the signature sig_b is valid, x will forward the packet to the next node on the route, and assign the value of $seq_s(s,d)$ to $seq_x(s,d)$.

4.2.2 Traffic Monitoring

Traffic monitoring is an indispensable component to detect possible injecting traffic attacks. In the chapter, each node will keep monitoring its neighbors' transmission activities using the proposed *header watcher* mechanism. Specifically, when a node x detects that a neighbor y is transmitting a data packet, no matter whether x is the receiver of this transmission or not, x will try to receive and decode the packet header sent by y. Actually this is needed by most wireless networks: without decoding the header, how can a node know whether this packet target at it or not? The uniqueness of the proposed header watcher mechanism lies in that each node will also check the validity of the signature for the packet header. If the signature of the packet header is valid, x will put the packet header into the set $List(s,d,x)$ in x's records, which will be used later to detect whether s has launched injecting traffic attacks.

Unlike the "watchdog" mechanism introduced in [58], which requires a node to buffer all the packets that it has sent or forwarded and to keep monitoring its neighbors' transmission activities in order to check whether those packets have been forwarded by them, the "header watcher" mechanism proposed in this chapter only requires a node to monitor the packet headers around its neighborhood. Since only packet header needs to be received and decoded, and the header of a packet is much shorter than the body of a packet, a lot of effort can be saved comparing to

the watchdog mechanism which requires receiving, decoding, and comparing the whole packets.

In general, if all packet headers received by node x are recorded, with the increase of x's staying time in the network, more and more storage will be required. Actually, in our scheme, for each legitimate source-destination pair (s, d), only those packet headers received after the last valid route request issued by (s, d) need to recorded by x; in another words, only those packet headers whose sequence numbers are larger than the sequence number broadcast by s in its last valid route request packet. With this modification, the storage requirement become very small and does not increase over x's staying time in the network. In Section 4.4, we will also show how to further decrease the storage requirement.

4.2.3 Injecting Traffic Attack Detection

In this chapter each good node in the network will perform injecting traffic attack detection based on the observed behaviors. Specifically, for each source-destination pair (s, d) with $List(s, d, x)$ being non-empty in good node x's records, the following detection rules will be used by x to check whether s has launched injecting traffic attacks:

- Rule 1: If $List(s, d, x)$ is not empty and the source-destination pair (s, d) is illegitimate, x will mark s as malicious.
- Rule 2: x received a request issued by an illegitimate source-destination (s, d), x will mark s as malicious.
- Rule 3: For any packet header $\{R, seq_s(s, d), sig_h\}$ which belongs to $List(s, d, x)$, if route R has more than T_{maxhop} relays, x will mark s as malicious.

- Rule 4: If x detects that there exist two valid packet headers $\{R, seq_s(s,d), sig_h\}$ and $\{R', seq'_s(s,d), sig'_h\}$ in the set $List(s,d,x)$ with $seq_s(s,d) = seq'_s(s,d)$ but $R \neq R'$, x will mark s as malicious.

- Rule 5: Let $seq_{max}(s,d)$ be the maximum possible sequence number corresponding to the source-destination pair (s,d) at time t, that is, $seq_{max}(s,d) = f_{s,d}(t)$ at time t. If x detects that there exists a sequence number $seq_s(s,d)$ in $List(s,d,x)$ with $seq_s(s,d) > seq_{max}(s,d)$, x will mark s as malicious.

The first two rules imply that only legitimate source-destination pairs can inject packets into the network. Rule 3 implies that no routes should have more than T_{maxhop} relays. Rule 4 handles multiple route attack. Rule 5 handles attackers which inject more packets than they should. In summary, rule 4 and 5 are used to prevent attackers from injecting more packets than they are allowed by associating each packet with a unique sequence number. That is, no any two packets for the same traffic pair should have the same sequence number, and the sequence number has to be increased monotonically.

Once x detects that s is launching injecting traffic attacks, x will also inform the other nodes in the network by broadcasting an ALERT message which includes evidence such as the corresponding packet headers. When other good nodes have received the ALERT message, after necessary verification (i.e., signatures are valid), they will also mark s as malicious.

Next we analyze the effects of possible impersonation attacks that can be launched by attackers. In the proposed mechanisms, the only way that an attacker m can impersonate a good node s whose has not been compromised is to first record the packets that s has transmitted, then later forwards/broadcasts these packets. Specifically, there are two situations:

- Situation 1: m recorded a query packet issued by s and rebroadcast it later. However, since this query packet has been seen by all other nodes in the network due to the flooding nature of query message, no nodes will further process this query packet.

- Situation 2: m recorded a data packet issued by s and forwarded it later. However, since nodes on the route associated to this data packet will only process this packet at most one time, forwarding this packet at time t_1 by m cannot cause damage to other nodes.

In summary, impersonation attack cannot cause further damage to good nodes in the network. Furthermore, it can be readily checked that as long as s is good and has not been compromised, the probability that x will mark s as malicious is 0. That is, the false alarm ratio of the above detection rules is 0.

4.2.4 Overhead Analysis

Now we analyze the overhead associated with the above defense mechanisms. According to the above description, there is no extra communication overhead. Meanwhile, the computation overhead comes from generate and verify the signatures. More specifically, the extra computation overhead comes from generating and verifying the signatures for packet headers. Comparing to packet body, the length of packet header is much smaller, so the extra computation overhead is also small. Now we analyze the storage overhead. In the proposed defense mechanism, each node needs to store the set of packet headers between two consecutive route query requests. In mobile ad hoc networks, due to dynamic topology change, the time interval between two consecutive route query requests is usually short. Therefore,

the number of packet headers that need to be stored is also small. Section 4.4 will discuss how to further reduce the storage overhead.

4.3 Theoretical Analysis

According to the secure route discovery procedure described in Section 4.2.1, a good node x will only forward at most 1 route request in any time interval $T_x(s, d)$ for any legitimate source-destination (SD) pair (s,d), and will not forward route requests for any illegitimate SD pairs, therefore the total damage that can be caused by attackers launching query flooding attacks is bounded. Next we analyze the effects of IDPA. Assume that node s is malicious and tries to launch IDPA with d being the destination of the packets injected by s. To avoid being detected immediately, the SD pair (s, d) must be legitimate and d must be malicious too, otherwise, s can be easily detected by d as malicious. According to Section 4.1, there are three possible ways to launch IDPA: simple IDPA, long-route IDPA and multiple-route IDPA.

We first consider *simple IDPA*. According to Section 4.2.1, in order for good nodes to forward packets for s, s has to increase the sequence number $seq_s(s, d)$ by 1 after each packet delivery. Unless all nodes on the selected route are malicious, which makes no sense, the good nodes on route R can easily detect that s is launching IDPA by comparing the received packets' sequence number with $f_{s,d}(t)$ defined in Section 4.2.3. That is, when launching simple IDPA, the attackers can be immediately detected and can cause negligible damage.

If s launches *long-route IDPA*, since much more good nodes will be involved, s can cause similar damage as launching simple IDPA. However, as described in Section 4.2.1, the maximum allowable number of hops per route is bounded by

T_{maxhop}, and good nodes will drop all packets with the associated number of hops more than T_{maxhop}. Therefore the damage is upper-bounded by $f_{s,d}(t)T_{maxhop}$.

Finally we consider the *multiple-route IDPA*. To avoid being detected immediately, the packet injection rate to each route must conform to $f_{s,d}(t)$, and the selected routes must be node-disjoint, that is, no selected routes should share any common good node except s and d, otherwise, if a good node x lies in more than one route from s to d, it can easily detect whether s and d have launched multiple-route IDPA. Meanwhile, the packets passing through the same route should have different sequence numbers in order for good nodes on the route to forward them. Based on whether s allows packets in different routes to share the same sequence numbers and what transmission techniques s will use, there are three cases:

- Case 1: s dose not allow packets on different routes to share the same sequence numbers. Since $seq_s(s,d) \leq f_{s,d}(t)$ is required to let s avoid being detected immediately, in this case s has no extra gain by comparing with launching simple IDPA.

- Case 2: s allows packets on different routes to share the same sequence numbers, and transmits packets omnidirectionally. Since s's neighbors will keep monitoring s's packets transmission, they can easily detect that some packets sent by s through different routes use the same sequence number, which indicates that s is launching IDPA. Therefore if s can only transmit packets omnidirectionally, s should not launch multiple-route IDPA.

- Case 3: s allows packets on different routes to use the same sequence numbers, and can transmit packets using directional transmission techniques. Since now s's neighbors cannot receive s' transmission not targeting on them, they have little chance to directly detect that s is launching IDPA. However,

since good nodes in the network use omnidirectional transmission techniques, the probability that s can successfully launch multiple-route IDPA without being detected still approaches 0, as to be shown next.

Next we derive the upper-bounds for the probability that s is able to successfully pick n node-disjoint routes to inject data packets without being detected immediately, as illustrated in Case 3. We consider the most general situation that the destination d does not know the exact locations of those nodes within its transmission range, and all d's neighbors are good nodes. Given a node x and a certain area S, we say that x is randomly deployed inside S according to the 2D uniform distribution if for any subarea $S_1 \subset S$ we have $P(x \in S_1 | x \in S, S_1 \subset S) = S_1/S$. Then we have the following theorem.

Theorem 4.3.1 *Suppose that N good nodes are independently deployed inside a large area of S according to the 2D uniform distribution. Suppose that all of these N nodes use omnidirectional transmission techniques and r is their common maximum transmission distance. Suppose that the SD pair (s, d) collude to launch IDPA with s using directional transmission technique and s and d not knowing the exact location of the nodes inside d's receiving range (which is r). If the defending mechanisms described in Section 4.2 are used by good nodes, then the probability $P(n, r, N)$ that the two attackers can successfully pick n node-disjoint routes to launch multiple-route IDPA without being detected immediately is upper-bounded by*

$$P(n, r, N) \leq \left(\frac{3\sqrt{3}}{4\pi}\right)^{\binom{n}{2}} \sum_{k=n}^{N} P_1(k, N) \left(n \left(\frac{3\sqrt{3}}{4\pi}\right)^{\binom{n-1}{2}}\right)^{k-n}. \tag{4.2}$$

where $P_1(k, N)$ is defined as follows:

$$P_1(k, N) = \binom{N}{k} \left(\frac{\pi r^2}{S}\right)^k \left(1 - \frac{\pi r^2}{S}\right)^{N-k}. \tag{4.3}$$

Before proving Theorem 4.3.1, we first prove the following lemmas.

Lemma 4.3.2 *Assume N nodes are independently deployed inside an area of S according to the 2D uniform distribution. For any node x inside subarea $S_1 \subset S$ and for any subarea $S_2 \subset S_1$, we have*

$$P(x \in S_2 | x \in S_1, S_2 \subset S_1 \subset S) = \frac{S_2}{S_1} \tag{4.4}$$

Proof

$$P(x \in S_2 | x \in S_1, S_2 \subset S_1 \subset S) = \frac{P(x \in S_2, x \in S_1 | S_2 \subset S_1 \subset S)}{P(x \in S_1 | S_2 \subset S_1 \subset S)} = \frac{P(x \in S_2 | S_2 \subset S)}{P(x \in S_1 | S_1 \subset S)} = \frac{S_2}{S_1}. \tag{4.5}$$

That is, the conditional distribution of x in S_1 is independent of S, which is also the 2D uniform distribution. ■

Lemma 4.3.3 *Assume nodes x and y are independently deployed inside a certain area S according to the 2D uniform distribution. Given $x \in S_1 \subset S$ and $y \in S_1 \subset S$, and given any subareas $S_x \subset S_1$ and $S_y \subset S_1$, we have*

$$\begin{aligned} &P(x \in S_x, y \in S_y | x \in S_1, y \in S_1, S_x \subset S_1, S_y \subset S_1) \\ &= P(x \in S_x | x \in S_1, S_x \subset S_1) P(y \in S_y | y \in S_1, S_y \subset S_1) \end{aligned} \tag{4.6}$$

Proof Since the deployment of x and y are independent of each other, we have

$$\begin{aligned} & P(x \in S_x, y \in S_y | x \in S_1, y \in S_1, S_x \subset S_1, S_y \subset S_1) \\ = \; & P(x \in S_x | x \in S_1, S_x \subset S_1, y \in S_y \subset S_1) * \\ & P(y \in S_y | y \in S_1, S_y \subset S_1, x \in S_1, S_x \subset S_1) \\ = \; & P(x \in S_x | x \in S_1, S_x \subset S_1) P(y \in S_y | y \in S_1, S_y \subset S_1) \end{aligned}$$

That is, the distribution of x and y inside S_1 are independent of each other. ■

Lemma 4.3.4 *Let S be a circular area with o being the center and R being the radius. Assume that node x lies in S and $P(A \in S_1 | A \in S, S_1 \subset S) = \frac{S_1}{S}$. Let $d(x)$ denote the random variable of the distance from x to o, then*

$$P(d(x) = r | x \in S) = \begin{cases} \frac{2r}{R^2} & 0 \leq r \leq R \\ 0 & r > R \end{cases} \tag{4.7}$$

Proof For any $0 < r \leq R$, we have

$$P(d(x) = r | x \in S) = \lim_{\Delta \to 0} \frac{\pi r^2 / \pi R^2 - \pi (r - \Delta)^2 / \pi R^2}{\Delta} = \frac{2r}{R^2} \tag{4.8}$$

For any $r > R$, we have $x \notin S$, which implies $P(d(x) = r | x \in S) = 0$. ■

Lemma 4.3.5 *Let S be a circular area with o being its center and R being its radius. Given that two nodes a and b independently deployed in S according to the 2D uniform distribution, we have*

$$P(|ab| > R | a \in S, b \in S) = \frac{3\sqrt{3}}{4\pi}, \tag{4.9}$$

where $|ab|$ denote the distance between a and b.

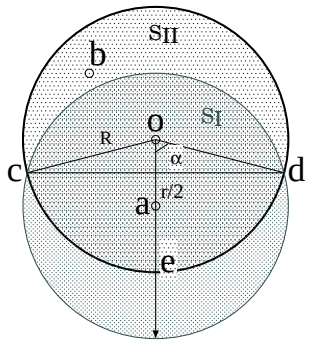


Figure 4.3: Illustration

Proof We use Figure 4.3 to help illustrating the proof. Let r denote the distance from a to o, let C_o denote the circle with o being the center and R being the radius,

and let C_a denote the circle with a being the center and R being the radius. Let c and d be the intersecting points between the two circles C_o and C_a, and let $\alpha = \angle coa = \angle doa$. Let $S_I(r)$ denote the intersecting area inside both circles C_o and C_a with $|oa| = r$, and let $S_{II}(r)$ denote the area of S subtracted by $S_I(r)$. Then we have

$$P(|ab| > R \big| a \in S, b \in S) = \int_0^R \frac{2r}{R^2} \frac{S_{II}(r)}{S} \mathbf{d}r, \tag{4.10}$$

where (4.10) comes from Lemma 4.3.5. We first calculate $S_I(r)$:

$$S_I(r) = 2\left(R^2 \arccos\frac{r}{2R} - \frac{r}{2}\sqrt{R^2 - (\frac{r}{2})^2}\right), \tag{4.11}$$

where $\alpha = \arccos(\frac{r}{2R})$. Then $S_{II}(r)$ can be calculated as

$$S_{II}(r) = R^2\left(\pi - 2\arccos\frac{r}{2R} - \frac{r}{R^2}\sqrt{R^2 - (\frac{r}{2})^2}\right). \tag{4.12}$$

By integrating (4.12) into (4.10), we have $P(|ab| > R \big| a \in S, b \in S) = \frac{3\sqrt{3}}{4\pi}$. ■

Lemma 4.3.6 *Assume that n nodes $A = \{a_1, \ldots, a_n\}$ are independently deployed inside a circular area S according to the 2D uniform distribution with R being the radius, then we have*

$$P(|a_i a_j| > R : \forall a_i, a_j \in A) \leq P(|a_1 a_2| > R)^{\binom{n}{2}} \tag{4.13}$$

Proof

$$\begin{aligned}
& P(|a_i a_j| > R : \forall a_i, a_j \in A) \\
= \; & P(|a_1 a_2| > R, \ldots |a_1 a_n| > R, \ldots, |a_{n-1} a_n| > R) \\
= \; & P(|a_1 a_2| > R \big| |a_1 a_3| > R, \ldots |a_{n-1} a_n| > R) P(|a_1 a_3| > R, \ldots, |a_{n-1} a_n| > R) \\
= \; & P(|a_1 a_2| > R \big| |a_1 a_i| > R, |a_2 a_i| > R : \forall 3 \leq i \leq n) P(|a_1 a_3| > R, \ldots, |a_{n-1} a_n| > R)
\end{aligned}$$

Given $|a_1 a_i| > R$ and $|a_2 a_i| > R$ for any $3 \leq i \leq n$, we can draw a circle with a_i being the center and R being the radius. To conform to the statement that

"$\forall a_i, a_j \in A, |a_i a_j| > R$", both a_1 and a_2 cannot lie inside the intersecting area between this circle and the circle with o being the center. That is, a_1 and a_2 are now restricted in an area of $S' \subset S$ smaller than S. So the probability that $|a_1 a_2|$ is larger than R under such restrictions will become smaller than without such restrictions. That is,

$$P(|a_1 a_2| > R \big| |a_1 a_i| > R, |a_2 a_i| > R) \leq P(|a_1 a_2| > R : \forall 3 \leq i \leq n). \quad (4.14)$$

Following the same arguments we can have

$$P(|a_i a_j| > R : \forall a_i, a_j \in A) \leq \prod_{1 \leq i < j \leq n} P(|a_i a_j| > R). \quad (4.15)$$

Since there are total $\binom{n}{2}$ items in the product, and nodes in A are symmetric, we can conclude that (4.13) holds. ■

Lemma 4.3.7 *Assume $n+m$ nodes $\{a_1, \ldots, a_n, b_1, \ldots, b_m\}$ are independently deployed inside a circular area S according to 2D uniform distribution with R being the radius. Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$, then we have*

$$P(|a_i b_l| > R \text{ or } |a_j b_l| > R : \forall a_i, a_j \in A, b_l \in B, i \neq j) \leq \left(nP(|a_1 b_1| > R)^{n-1}\right)^m \quad (4.16)$$

Proof Let $A_i = A - \{a_i\}$. Given any $b \in B$, to say "$|a_i b| > R$ or $|a_j b| > R : \forall a_i, a_j \in A, a_i \neq a_j$" is equivalent to say "there exists at least one A_i with $|xb| > R$ for any $x \in A_i$", that is,

$$\begin{aligned}
& P(|a_i b| > R \text{ or } |a_j b| > R : \forall a_i, a_j \in A, a_i \neq a_j) \\
= \;& P((|xb| > R : \forall x \in A_1) \text{ or } \ldots \text{ or } (|xb| > R : \forall x \in A_n)) \\
\leq \;& \sum_{i=1}^{n} P(|xb| > R : \forall x \in A_i) \\
= \;& nP(|xb| > R : \forall x \in A_1) \\
\leq \;& nP(|a_1 b| > R)^{n-1}
\end{aligned}$$

Due to the symmetry and independence of the m nodes in B, we can concludes that (4.16) holds. ■

Now Theorem 4.3.1 can be proved as follows:

Proof Let C_d denote the circle with d being the center and r being the radius. For s and d to be able to successfully pick n node-disjoint routes to launch multiple-route IDPA without being detected immediately, they need to pick at least n distinct nodes inside C_d, one for each route, to act as the last intermediate nodes on these routes. Since s and d do not know the exact locations of the nodes inside C_d, these n nodes can only be randomly selected. It is easy to see that the following **three necessary conditions** must be satisfied in order for the attackers to succeed:

C1. There exist at least n nodes inside C_d, otherwise, s and d can never have n node-disjoint routes between them.

C2. Given that there are $k \geq n$ nodes inside C_d, and that s and d are to randomly select n nodes among them to act as the last intermediate node for these n node-disjoint routes, then for any two nodes among the n nodes selected by s and d, no node should lie in the other nodes' transmission range. Otherwise, if any two of the n nodes lie in each other's transmission range, they can easily detect that s is launching multiple-route IDPA.

C3. Given that the n nodes have been selected by s and d, there should exist no other good nodes (nodes excluding the selected n good nodes) which can simultaneously lie in any two of these n nodes' transmission range. Otherwise, if there exist one such node, then it can easily detect that s is launching multiple-route IDPA.

Let $P_1(k, N)$ denote the probability that there are k nodes inside C_d, $P_2(n, r, k)$ denote the probability that the condition C2 can be satisfied given that the n nodes are randomly selected among $k \geq n$ nodes inside C_d, and $P_3(n, r, k, N)$ denote the probability that the condition C3 can be satisfied given there are $k \geq n$ nodes inside C_d and the n nodes have been determined by s and d. It is easy to see that

$$P(n, r, N) \leq \sum_{k=n}^{N} P_1(k, N) P_2(n, r, k) P_3(n, r, k, N). \tag{4.17}$$

Since nodes are independently deployed inside S according to the 2D uniform distribution, we can immediately have

$$P_1(k, N) = \binom{N}{k} \left(\frac{\pi r^2}{S}\right)^k \left(1 - \frac{\pi r^2}{S}\right)^{N-k}. \tag{4.18}$$

Given that k nodes lie in C_d, according to Lemma 4.3.2 and Lemma 4.3.3, it is equivalent to say that these k nodes are independently deployed inside C_d according to the 2D uniform distribution. According to Lemma 4.3.5 and Lemma 4.3.6, we can have

$$P_2(n, r, k) = \left(\frac{3\sqrt{3}}{4\pi}\right)^{\binom{n}{2}}. \tag{4.19}$$

To simplify the analysis, we consider a modified version of condition C3: given any two nodes among the selected n nodes, there should exist no other good nodes *inside C_d but not belonging to these n nodes* which can simultaneously lie in these two nodes' transmission range. That is, only a small subset of the applicable nodes are considered. Let $P_3'(n, r, k, N)$ denote the probability that the modified condition C3 can be satisfied given there are $k \geq n$ nodes inside C_d and the n nodes have been determined by s and d, then we must have $P_3(n, r, k, N) \leq P_3'(n, r, k, N)$. According to Lemma 4.3.5 and Lemma 4.3.7, the probability that the modified

condition C3 can be satisfied is upper-bounded by

$$P_3'(n, r, k, N) \leq \left(n \left(\frac{3\sqrt{3}}{4\pi} \right)^{\binom{n-1}{2}} \right)^{k-n} \tag{4.20}$$

By combining the above results, we can conclude that (4.2) as well as Theorem 4.3.1 holds. ■

Theorem 4.3.8 *The probability that two colluding attackers s and d can successfully pick 6 or more node-disjoint routes to launch multiple-route IDPA without being detected immediately is 0.*

Proof For the attackers s and d (assuming s is the source and d is the destination) to simultaneously pick 6 routes to launch multiple-route IDPA, it needs to pick 6 nodes within d's receiving range, that is, the circular area C_d with d being the center and r the radius. Let $A = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ denote the set of 6 selected nodes by s and d that lies inside C_d. One necessary condition for the attackers to succeed is that for any $a_i, a_j \in A$, we must have $|a_i a_j| > r$ for any $a_j \in A$ and $a_j \neq a_i$. Now we show that it is not achievable. If there exist $a_i, a_j \in A$ with $\angle a_i d a_j = 0$, then we must have $|a_i a_j| \leq r$. Next we only need to consider the situations that for any $a_i, a_j \in A$, $\angle a_i d a_j \neq 0$. For each node $a_i \in A$, we draw a radial originated from d and passing a_i, and let a_i' be the intersecting point between the radial da_i and the circumference of the circle C_d. Any two radials will partition the circular area C_d into two sectors. We say a sector is *singleton* if none of the nodes in A lie inside this sector (including the arc but excluding the two radials). It is easy to say that the 6 nodes will partition the circle into 6 singleton sectors. To satisfy the above necessary condition, the angle of each singleton sector should be more than $\pi/3$: if the angle of a singleton section is no more than $\pi/3$, let a_i be the node on one side of this sector, and a_j be the node on the other side of this

sector, then for any point x that lies in the segment da'_i and any point y that lies in the segment da'_j, we must have $|xy| \leq r$. Since we have 6 singleton sectors, and each singleton sector has an angle of more than $\pi/3$, the summed angle is more than 2π, which contradicts the fact that a circle is 2π. Given this conclusion, it is trivial to show that more than 6 routes is also not achievable. ■

We have also evaluated through experiments the upper-bounds of the success ratio for two colluding attackers s and d to launch multiple-route IDPA with s using directional transmission technique. Given a rectangular area of $20r \times 20r$, we put d in the center of the area. At each round of experiment, we independently deploy $400r^2\rho$ nodes inside the area according to 2D uniform distribution and randomly pick n nodes inside d's receiving range, where ρ is referred to as the node density. We say (s, d) may succeed only if all of the three necessary conditions presented in the proof of Theorem 4.3.1 are satisfied. For each configuration of route number n and node density ρ, 10^7 experiments have been conducted, and the upper-bounds are obtained as the ratio of total success number over the total number of experiments.

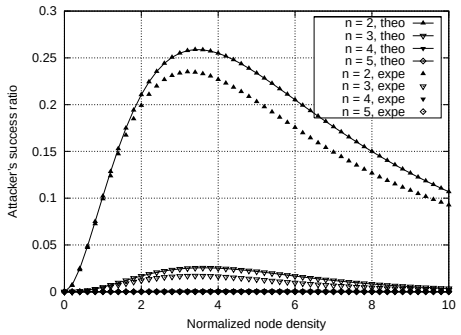


Figure 4.4: Upper bounds of attackers' success probability

Both experimental and theoretical upper-bounds are plotted in Figure 4.4, where "theo" denotes the theoretical upper-bounds obtained using (4.2), "expe"

denotes the experimental upper-bounds obtained through experiments described above, and "n" denotes the number of node-disjoint routes to be picked by the malicious SD pair (s, d). In Figure 4.4, the normalized node density is defined as the average number of nodes inside an area of πr^2. Since both the theoretical and experimental upper-bounds corresponding to $n = 4$ and $n = 5$ are almost equal to 0 across all illustrated node densities (e.g., for $n = 4$, all values are less than 2×10^{-3}), the four curves associated to $n = 4, 5$ have almost overlapped into one single curve, which is the lowest curve illustrated in Figure 4.4. For $n = 2, 3$, we can see that the success ratio increases first with the increase of node density until it arrives at a peak, then decreases with the further increase of node density, which is consistent with (4.2). The reason is as follows: with the increase of the node density, the probability P_1 that the condition C1 can be satisfied increases monotonically from 0 to 1, the probability P_2 that the condition C2 can be satisfied keeps unchanged, while the probability P_3 that the condition C3 can be satisfied decreases monotonically from 1 to 0, and when ρ is small, the value of P_1 dominates the bound, while when ρ is large, the value of P_3 dominates the bound. From Figure 4.4 we can also see that there exist gaps between theoretical results and experimental results. The reason is that when we calculate the probability of condition C3 being satisfied, only a subset of applicable nodes have been considered, which make the theoretical upper-bounds a little bit looser (higher) than the experimental upper-bounds.

The above upper bounds are evaluated based on a fixed topology, that is, the set of links $E(t)$ keeps unchanged for all time index t. However, due to node mobility, $E(t)$ will change over time t, therefore s needs to frequently update routes. Then after several route updates, the probability that s still has not been detected as

malicious will be very small. For example, assume that each route update is independent, after 5 times of route updates, even for $n = 2$, the probability that s has not been detected as malicious is less than 0.06%. That is, attackers has negligible chance to flee. In summary, when the malicious SD pair (s, d) tries to launch IDPA, to avoid being detected and to maximize the damage, the optimal strategy is to use only one route to inject data packets by conforming to both the maximum hop number L_{maxhop} and the legitimate rate $\lambda_{s,d}$, which is equivalent to say that the optimal strategy is not to launch IDPA.

Besides injecting traffic by themselves, attackers may also impersonate good nodes to launch injecting traffic attacks in attempt to avoid being detected as well as let those impersonated good nodes being mistakenly detected as malicious. Next we analyze the effects of possible impersonation attacks that can be launched by attackers. In the proposed mechanisms, the only way that an attacker m can impersonate a good node s who has not been compromised is to first record the packets that s has transmitted, then later forwards/broadcasts these packets. Specifically, there are two situations:

- Situation 1: m recorded a query packet issued by s and rebroadcast it later. However, since this query packet has been seen by all other nodes in the network due to the flooding nature of query message, no nodes will further process this query packet.

- Situation 2: m recorded a data packet issued by s and forwarded it later. However, since nodes on the route associated to this data packet will only process this packet at most one time, forwarding this packet at time t_1 by m cannot cause damage to other nodes.

In summary, impersonation attack cannot cause further damage to good nodes in

the network. Further, it is ready to check that as long as a good node s has not been compromised, with no chance it will be marked as malicious. That is, the false alarm ratio of the above detect rules is 0.

4.4 Centralized Detection with De-centralized Implementation

The defense system described in Section 4.2 are fully distributed. However, the drawback this system is that it may have relatively high storage complexity. Meanwhile, each node needs to have prior knowledge of the set of legitimate traffic pairs, which may not be available to all nodes in general. Next we describe a modified version of the proposed defense system. In the modified version, instead of performing attacker detection by itself, each good node will report the observed information to certain nodes which we called centralized detectors, then the centralized detector will perform attacker detection based on the collected traffic information. In general, the centralized detectors will be under stronger protection than normal nodes and may have more powerful computation capability and more storage, such as base station.

The detailed description of the modified defense system is as follows. First, the route discovery and packet delivery procedure is the same as described in Section 4.2.1. Second, the monitoring mechanism is still header watcher as described in Section 4.2.2 with the following modification: for each good node, instead of storing all listened valid packet headers locally, most time it does not need to any packet headers locally, but only needs to store the following three-tuple (traffic pair, sequence number, route) that is associated to each listened valid packet

headers. A good node needs to record the full packet headers only if it has been notified by the centralized detectors to do so, as explained next. Furthermore, instead of reporting each listened packet header information separately, each good node will report the listened packet header information in a batch mode, that is, each report consists of many listened packet header information.

For the centralized detector, its job is to perform injecting traffic attack detection by applying similar detection rules as described in Section 4.2.3. The major difference lies in that when the centralized detector performs injecting traffic attack detection, the procedure are two steps. In the first step, the detector will check whether a node has injected two packets with the same sequence number or whether a sequence number is larger than specified upper-bound based only on the collected partial packet header information, that is, without checking the packet header signatures. If any of the two conditions has been satisfied, the detector then will request those nodes who report such information to submit full packet headers. That is, the centralized detector needs concrete evidence to charge the attacker.

Now we use an example to illustrate the modified detection procedure. Assume that node a has reported a sequence number seq_1 and route R_1 associated to traffic pair (s, d), and node b has reported a sequence number seq_2 and route R_2 associated to traffic pair (s, d). After the centralized detector has received these reports, it will find that $seq_1 = seq_2$ but $R_1 \neq R_2$. Then the detector has reason to suspect that s has launched injecting traffic attacks. When this happens, the detector will ask node a and b to report the full packet headers next time such that it can collect concrete evidence to charge s.

From the above description we can see that although the attacker detection

is performed in a centralized, the monitoring is still fully distributed. Now we analyze the detection performance of the modified defense system. It is easy to see that either simple IDPA or long-route IDPA can be easily detected. Meanwhile, for the multi-route IDPA, requiring packets sent via different route to use different sequence number has not gain from the attacker's point of view, and allowing packets sent via different route to use same sequence number will be detected immediately when omnidirectional transmission technique is used. Now we focus on the scenario that the attackers allow packets sent via different route to use same sequence number will be detected immediately, and directional transmission technique is used to avoid being detected.

Given that an attacker s picks n node-disjoint routes to simultaneously inject packets and packets on different routes will share the same set of sequence numbers, as long as at least two nodes on the selected routes are good, it is easy to check that with zero probability that s can avoid being detected. In other words, attackers have no chance to launch IPDA without being detected. Comparing to the fully distributed defense system described in Section 4.2, the storage overhead of the modified defense system can be dramatically reduced, but some extra communication overhead is introduced due to that each node needs to report to the centralized detector. However, since the size of each report is very small comparing to the data packet, the extra communication overhead is negligible. For example, if the average packet size is 1000 bytes, and the report size is 20 bytes, then the increased overall traffic is only 2%.

Until now we have assumed that each good node will keep listening all the packet transmission in its neighborhood. Next we show how to further decrease the overhead by letting nodes selectively listen packet transmissions, with negligible

degradation of the detection performance. Specifically, each node can selectively listen its neighbors' transmission with a certain probability p, which we called *probabilistic monitoring*. That is, a packet transmission event happens in a good node's neighborhood, with only probability p this node will monitor this transmission and report the observation to the centralized detector. Now when an attacker has injected $n \geq 2$ packets with the same sequence number via n node-disjoint routes, with no more than probability $p(n) = (1-p)^n + p(1-p)^{n-1}$ the attacker can avoid being detected. Furthermore, after the attacker has injected k packets, the probability that it will not be detected will be decreased to $p(n)^k$, which goes to 0 with the increase of k. By applying probabilistic monitoring, the communication overhead can be further decreased by $1-p$, while the detection performance only suffers negligible degradation.

One possible drawback of such centralized detection mechanism is that the detector itself can also become attackers' target. Besides increasing the protection level, one can also increase the number of centralized detectors. For example, if there are 2 detectors in the network, even one has been compromised, the other still work well. In this case, for each node, it can either submit report to both detectors, or each time randomly pick one to submit, where the later is equivalent to reducing p by half.

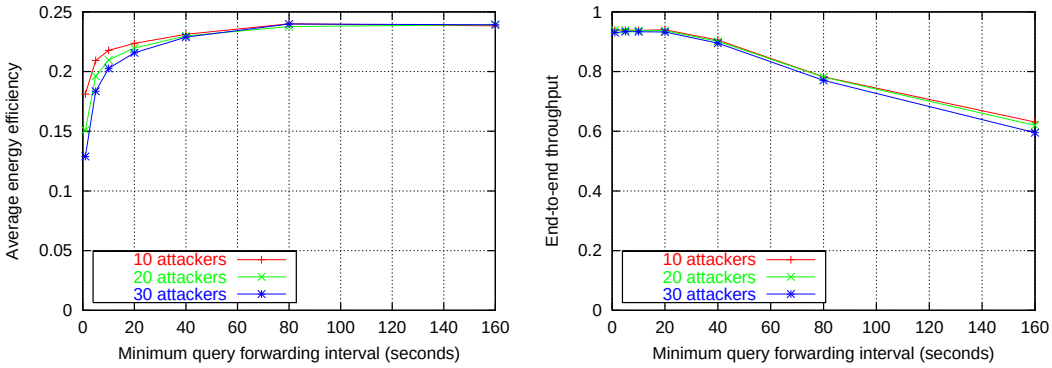
4.5 Simulation Studies

In the simulations, 100 good nodes and various number of malicious nodes are randomly deployed inside a rectangular area of 1500m*1500m, and the maximum transmission range for each node is 300 m. As mentioned before, random waypoint mobility model is used, with the average pause time being 300s, and $v_{max} = 10m/s$.

In the simulations, 50 good nodes are selected as the packet generators, and each will randomly pick a good node to send packets, therefore the total number of SD pairs are 50. For each malicious node who launches injecting traffic attacks, it will randomly pick another malicious node who also launches injecting traffic attacks as the destination to inject packets. For each malicious node who launches routing disruption attacks, it will not inject traffic to the network. All SD pairs (good or malicious) are set to be legitimate, and for each pair, packets are generated according to a Poisson process with a pre-specified traffic rate known by all nodes, where the average packet inter-arrival time is 1 second. For malicious nodes who launch injecting traffic attacks, they will increase the average packet injection rate by 10 times. Also, all data packets have the same size of 1024 bytes.

In our simulations, each configuration has been run 20 independent rounds using different random seeds, and the result are averaged over all the 20 rounds. For each round, the simulation time is set to be 5000 seconds. When we calculate the energy efficiency, only transmission energy consumption has been considered, one reason is that transmission energy consumption plays a major role in overall energy consumption, and another reason is that receiving energy consumption may vary dramatically over different communication systems due to their different implementations. However, both data and route request packets have been considered. We assume the transmission energy needed per data packet is normalized to be 1.

We first investigate the tradeoff between limiting the route request rate and system performance. Although the performance also depends on other factors such as the mobility pattern, the number of nodes in the network, the average number of hops per route, etc., to better illustrate the tradeoff between limiting the route request rate and system performance, the other parameters are set to be



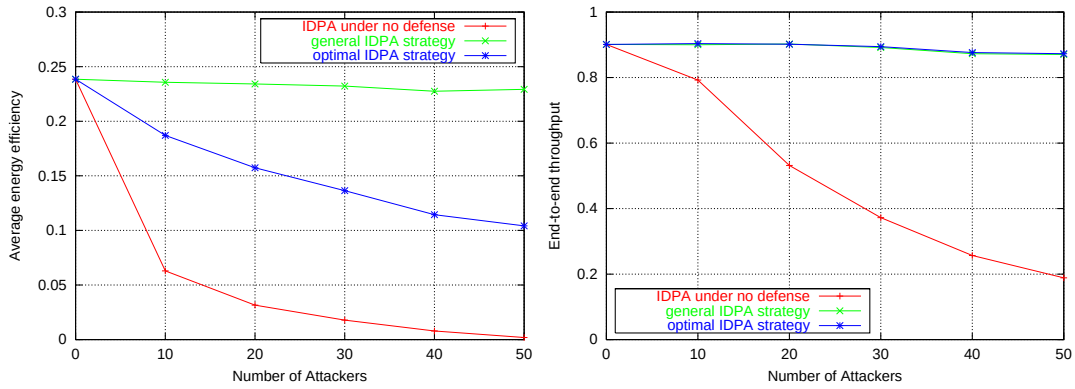
(a) Energy efficiency

(b) End-to-end throughput

Figure 4.5: Limiting route request rate vs. system performance

fixed. However, similar results can also be obtained by changing these parameters.

Fig. 4.5 illustrates the tradeoff between limiting the route request rate and network performance. In this set of simulations, all malicious nodes will only inject route request packets and will not inject any data packets or launch routing disruption attacks. We assume that all good nodes have the same minimum route request forwarding interval denoted by T^{min}, but all malicious nodes will set their route request rate to be 1 per second. From Fig. 4.5(a) we can see that with the increase of T^{min} from 1 to 80 seconds, the energy efficiency of good nodes also increases, and keeps almost unchanged from 80 to 160 seconds. The reason is that when T^{min} is small, attackers can waste good nodes' energy through injecting a lot of route request packets to request others to forward. Fig. 4.5(b) shows that with the increase of T^{min} from 1 second to 20 seconds, the end-to-end throughput of good nodes keeps almost unchanged, while with the increase of T_{min} from 80 seconds to 160 seconds, the end-to-end throughput of good nodes drops almost linearly. These results also motivate us to pick T^{min} to be 40 seconds in the following simulations.



(a) Energy efficiency

(b) End-to-end throughput

Figure 4.6: Effects of IDPA under different configurations

Fig. 4.6 shows the simulation results under various types of IDPA. In Fig. 4.6, "IDPA under no defense" denotes the case that attackers launched simple IDPA and the underlying system has not launched any defending mechanism; "general IDPA strategy" denotes the case that attackers launch IDPA but the mechanisms described in Section 4.2 have been launched, where both multiple-route IDPA and long-route IDPA have been simulated; "optimal IDPA strategy" denotes the case that attackers will use only one route to inject data packets which conforms both to the maximum hop number $T_{maxhop} = 10$ and to the legitimate maximum packet injection rate and the mechanisms described in Section 4.2 have been launched.

From Fig. 4.6(a) we can see that when there is no defending mechanisms for IDPA, even simple IDPA can dramatically degrade the energy efficiency of good nodes. When the defending mechanisms described in Section 4.2 are employed, from attackers' point of view, launching IDPA has no any gain in decreasing the energy efficiency of good nodes. However, if attackers apply the optimal IDPA strategy, they can still degrade the energy efficiency of good nodes. From Fig. 4.6(b) we can see that without employing necessary defending mechanisms, with the in-

crease of the number of attackers, even simple IDPA can dramatically degrade the end-to-end throughput of good nodes due to the congestion they caused. When the defending mechanisms described in Section 4.2 are employed, launching IDPA has almost no effects on the performance of good nodes' end-to-end throughput.

4.6 Summary

In this chapter we have studied the possible injecting traffic attacks that can be launched in mobile ad hoc networks, and proposed a set of mechanisms to defend against such attacks. Both query flooding attacks and injecting general data packets attacks have been investigated. Furthermore, for injecting general data packets attacks, the situations that attackers may use some advanced transmission techniques, such as directional antennas or beamforming, to avoid being detected have also been studied. Two set of defense mechanisms have been proposed, one is fully distributed, while the other is centralized with de-centralized implementation. Our theoretical analysis has shown that when the proposed mechanisms are used, the best strategy for attackers is not to launch injecting traffic attacks. Extensive simulation studies have also agreed with our theoretical analysis.

Chapter 5

Game Theoretic Analysis of Security in Cooperative Ad Hoc Networks

Although in Chapter 3 and Chapter 4 several mechanisms have been proposed to defend cooperative ad hoc networks against various insider attacks, there still exist some important issues which have not been fully addressed. A significant one is the optimality measure of defense mechanisms. For example, what metrics should be used to measure the optimality of the defense mechanism? Under certain optimality metrics, what are the optimal defending strategies, *especially when the environment is noisy and the monitoring is not perfect*? What strategies should the attackers use to maximize the damage to the network, and consequently what is the maximum possible damage that the attackers can cause? In this chapter we will formally address the above issues. Specifically, we will try to derive the optimal defending strategies as well as the maximum possible damage that can be caused by insider attackers under noise and imperfect monitoring.

This chapter is organized as follows. Section 5.1 describes the secure routing and packet forwarding game model with incomplete type information. Section 5.2 presents the devised defending strategies by incorporating statistical attacker detection mechanisms, as well as possible attacking strategies. The optimality of the devised strategies is analyzed in Section 5.3. Simulation results are presented in Section 5.4. Section 5.5 summarizes this chapter.

5.1 Game Model

In this chapter we jointly consider routing and packet forwarding in cooperative ad hoc networks, and model the interactions between good nodes and attackers as games, referred to as *secure routing and packet forwarding games*. We adopt Nash equilibrium as a basic optimality metric. In order to fully address the above issues, we focus on the following scenario: initially good nodes do not know who are attackers while attackers can know who are good nodes. This scenario can be regarded as the worst-case scenario from the defenders' point of view, that is, if a strategy can work well under this scenario, they can work well under any scenarios.

We consider cooperative ad hoc networks deployed in adversarial environments. According to their objectives, nodes in such networks can be classified into two types: *good* and *malicious*. The objective of good nodes is to optimize the overall system performance, while the objective of malicious nodes is to maximize the damage that they can cause to the system. In such networks, each node may generate (or collect) some data scheduled to be delivered to certain destinations, and the data rate from each node is determined by the common system goal, which is usually application-specific.

In this chapter we will still focus on insider attacks. Since we focus on packet

forwarding, we will mainly consider the following two representative attack models: *dropping packets* and *injecting traffic*. By dropping other nodes' packets, all the resources spent to transmit these packets are wasted, and the network's performance is degraded. Attackers can also inject an overwhelming amount of packets into the network: once the others have forwarded these packets but cannot get payback, those resources spent to forward these packets are wasted. Meanwhile, the attackers are allowed to *collude* to increase their attacking capability.

As we have known in Chapter 4, in cooperative ad hoc networks, without knowing any information about node's legitimate data generation rate, the detection of injecting traffic attacks will become extremely hard (or impossible). Fortunately, since cooperative ad hoc networks are designed to fulfill certain common goals, it holds in general that a node's legitimate data generation rate can be known or estimated by some other nodes in the network. For example, in ad hoc sensor networks designed to do environment surveillance, each node needs to send collected information to the centralized data collector, and the amount of data that each node can send is usually pre-determined by the system goal, and can be known or estimated by some other legitimate nodes. Similar as in Chapter 4, we assume that for each node s in the network, the number of packets that it will generate by time t is $T_s(t)$, which is usually different for different node[1]. In general, the exact value of $T_s(t)$ may not be known by other nodes in the network. In this chapter we assume that the upper-bound of $T_s(t)$, denoted by $f_s(t)$, can be known or estimated by some nodes in the network.

To formally analyze the security in cooperative ad hoc networks, we model

[1]In general, the number of packets that each node s will generate by time t can be modeled as a random variable, and $T_s(t)$ can be regarded as a specific realization.

the dynamic interactions between good nodes and attackers as a **secure routing and packet forwarding game** with incomplete type information and imperfect observation:

- **Players:** The set of players is denoted by N, which is the set of legitimate nodes in the network.

- **Types:** Each player $i \in N$ has a type $\theta_i \in \Theta$ where $\Theta = \{good, malicious\}$. Initially, each attacker knows any other player's type, while each good player assumes all nodes are good. That is, good nodes have incomplete information of the others' type. Let N_g denote the set of good players and $N_m = N - N_g$ the set of attackers.

- **Strategy space:**

  1. **Route participation stage**: For each player, after receiving a message requesting it to be on a certain route, it can either *accept* this request, denoted by A, or *not accept* this request, denoted by NA.

  2. **Route selection stage**: For each player who has a packet to send, after discovering a valid route[2], its decision can be either *request/use* this route to send the packet, denoted by R, or *not request/use* this route to send the packet, denoted by NR.

  3. **Packet forwarding stage**: For each relay node, once it has received a packet requesting it to forward, its decision can be either *forward* this packet, denoted by F, or *drop* this packet, denoted by D.

[2]A valid route means that all nodes on this route have agreed to be on this route and each node on this route lies inside the transmission range of its previous player on this route.

- **Cost:** For any player $i \in N$, transmitting a packet, either for itself or for the others, will incur cost c_i.

- **Gain:** For each good player $i \in N_g$, if a packet originated from it can be successfully delivered to its destination, it can get gain g_i.

- **Imperfect execution:** Due to noise, with probability p_e each decision F can be mistakenly executed as D.

- **Imperfect observation:** With probability p_m each forwarding outcome can be observed as dropping by the source (i.e., miss probability), and with probability p_f each dropping outcome can be observed as forwarding by the source (i.e., false positive probability). Meanwhile, when a node has injected a packet, with probability p_s it can avoid being detected by those who know its legitimate traffic injection rate.

- **Utility:** For each player $i \in N$, let $S_i(t)$ denote the number of i's packets that have been scheduled to send and have successfully arrived at their destinations by time t, let $F_i(j,t)$ denote the number of packets that i has forwarded for player $j \in N$ by time t, and let $W_i(j,t)$ denote the total times of wasted packet transmissions that i has caused to j by time t due to i dropping those packets that have been transmitted by j. Let t_f denote the lifetime of this network. Then we can model the players' utility (payoff) functions as follows:

    1. **Good players:** Since all good players belong to the same authority and pursue the common goals, they will share the same utility function

as follows:

$$U_g(t_f) = \frac{\sum_{i \in N_g} \left(S_i(t_f) g_i - F_i(t_f) c_i \right)}{\sum_{i \in N_g} T_i(t_f)}, \tag{5.1}$$

where

$$F_i(t) = \sum_{j \in N} F_i(j, t). \tag{5.2}$$

2. **Malicious players:** Since malicious players are allowed to collude, we assume they will also share the same utility function, defined as follows:

$$U_m(t_f) = \frac{\sum_{i \in N_m} \left(\sum_{j \in N_g} (W_i(j, t_f) + F_j(i, t_f)) c_j - \alpha F_i(t_f) c_i \right)}{t_f}. \tag{5.3}$$

Here parameter α is introduced to determine the relative importance of attackers' cost comparing to good players' cost. That is, from the attackers' point of view, it is worth spending cost c to cause the damage worth c' to good players as long as $\alpha < \frac{c'}{c}$.

The objective of good players is to maximize U_g, while the objective of attackers is to maximize U_m. If the game will be played for an infinite duration, then their utility functions will become $U_g = \lim_{t \to \infty} U_g(t)$ and $U_m = \lim_{t \to \infty} U_m(t)$, respectively.

Remark 1: On the right-hand side of (5.1), the numerator denotes the net profit (i.e., total gain minus total cost) that the good nodes obtained, and the denominator denotes the total number of packets that good nodes need to send. The utility function (5.1) represents the average net profit that good nodes can obtain per packet that needs to be delivered. Since good nodes do not have any prior knowledge of the other nodes' types, each good node may not know its exact payoff by time t, which introduce extra difficulty for optimal strategy design.

Remark 2: In (5.3), $W_i(j, t)c_j$ represents the total damage (or wasted energy) that i has caused to j by time t due to i launching dropping packets attacks,

$F_j(i,t)c_j$ represents the total damage that i has caused to j by time t due to i launching injecting traffic attacks, and $F_i(t)c_i$ represents the total cost incurred to i by launching both injecting traffic and dropping packets attacks by time t. In summary, the numerator of the right-hand side of (5.3) represents the net damage that the attackers caused to the good nodes. Since this value may increase monotonically, it is normalized by dividing the network lifetime t_f. Now this utility function represents the average net damage that the attackers cause to the good nodes per time unit. From (5.3) we can see that in this game setting the attackers' goal is to waste the good nodes' energy as much as possible. Alteratively, attackers can also have other types of goals, such as minimizing the good nodes' payoff. In Section 5.3 we will show that the performance of the proposed defending strategy is not sensitive to the attackers' specific goal, and in most situations maximizing (5.3) has the same effect as minimizing the good nodes' payoff under the proposed defending strategies.

Remark 3: The above game can be divided into many subgames as explained below. Once a player wants to send a packet to a certain destination, a subgame will be initiated which consists of three stages: in the first stage, the source will request some players to be on a certain route to the destination; in the second stage, the source will decide whether it should use this route to send the packet; in the third stage, each relay will decide whether it should help the source to forward this packet once receiving it. We refer to each subgame as a *single routing and packet forwarding subgame*. It is worth noting that a subgame may terminate immediately after finishing the first or the second stage.

To simplify our illustration, we assume that $g_i = g$ for all $i \in N_g$ and $c_i = c$ for all $i \in N$. Like many other routing protocols for ad hoc networks, in the

above game, the maximum number of hops per route will be upper-bounded by L_{max}, which is a pre-determined system parameter. Without loss of generality, we assume that $(1-p_e)^{L_{max}}g > L_{max}c$, otherwise the expected gain may be less than the expected cost. Since in ad hoc networks energy is usually the most precious resource, we can directly relate the cost to energy. The physical meaning of gain g may vary according to specific applications. However, as to be shown in Section 5.3, as long as g is reasonably large, it will affect the strategy design.

According to the above game model, in each single routing and packet forwarding subgame, for the initiator of this subgame, its strategy space is {R, NR}, while for each relay node, its strategy space is {(A, F), (A, D), (NA, F), (NA, D)}. Here (A, F) means that a relay node agrees to be on a certain route in the route participation stage and will forward the packet from the source in the packet forwarding stage, (A, D) means that a relay node agrees to be on a certain route in the route participation stage but will drop the packet from the source in the packet forwarding stage, (NA, F) means that a relay node does not agree to be on a certain route but will forward the packet from the source in the packet forwarding stage, and (NA, D) means that a relay node does not agree to be on a certain route and will drop the packet from the source in the packet forwarding stage.

In the above game we have assumed that some necessary monitoring mechanisms will be launched to detect possible packet dropping. We have also assumed that when a node transmits a packet, its neighbors can know who is the source of this packet and who is currently transmitting this packet. However, we do not assume any perfect monitoring, and each node makes its decision only based on local private and imperfect observation. In general, p_f, p_m, and p_s are determined by the underlying monitoring mechanism. We assume that p_e, p_f and p_m can be

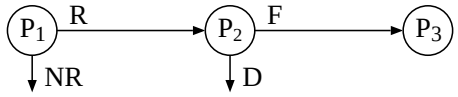


Figure 5.1: A single routing and packet forwarding subgame.

known or estimated by each node, but p_s may not be known by those detectors.

5.2 Defense Strategies with Statistical Attacker Detection

We first briefly study a simple subgame with complete type information and perfect observation: P_1 requests P_2 to forward a packet to P_3 through the route "$P_1 \rightarrow P_2 \rightarrow P_3$", and P_2 has agreed to be on this route, as illustrated in Fig. 5.1. Since the type information is complete, all players know each other's type. This is a two-stage extensive game with P_1 moving first. If P_1's action is NR, then the game will be terminated immediately, otherwise P_2 will take its action accordingly. The payoff profiles for this game under different scenarios are shown in Fig. 5.2, where the first value in each payoff profile corresponds to P_1's payoff and the second corresponds to P_2's payoff. Based on the types of P_1 and P_2, there are four different scenarios:

- Scenario 1: P_1 is good and P_2 is bad. Then the only Nash equilibrium is (NR, D) with payoff profile (0, 0).

- Scenario 2: P_1 is bad and P_2 is good. Then the only Nash equilibrium is (NR, D) with payoff profile (0, 0).

- Scenario 3: Both players are good. In this scenario, if $g > 2c$, the only Nash equilibrium is (R, F) with payoff profile $(g - 2c, g - 2c)$; if $g < 2c$, the only

Nash equilibrium is (NR, F) with payoff profile (0, 0); while if $g = 2c$, there are two Nash equilibria (NR, F) and (R, F), both have the same payoff profile (0, 0).

- Scenario 4: Both players are bad. Then the only Nash equilibrium is (NR, D) with payoff profile (0, 0).

Based on the above analysis we can conclude that in a two-hop subgame with complete type information:

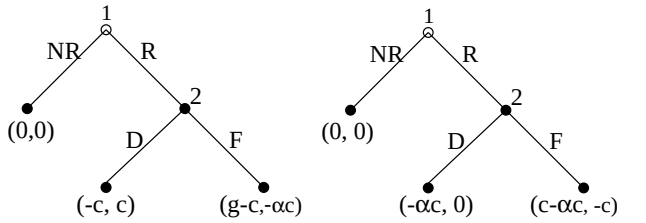
1. A good node should neither forward any packet for attackers nor request any attackers to forward packets. Meanwhile, good nodes should always forward packets for other good nodes provided $g > 2c$.

2. A malicious node should not forward any packet and should not request other nodes to forward packets.

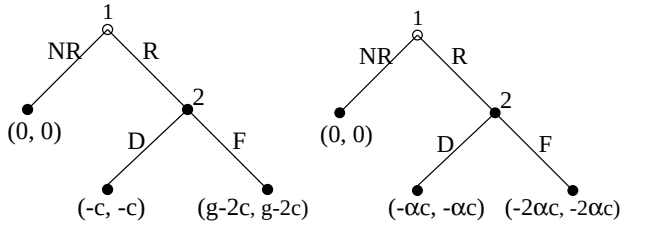
This can be easily generalized to the multi-hop scenario, that is, no good nodes should work with malicious nodes.

However, defending against insider attacks in realistic scenarios is much more challenging due to the following reasons. First, good nodes cannot know who are attackers *a priori*. Second, owing to noise, decision execution may not be perfect. Third, monitoring errors will be very common because of the fully distributed nature and limited available resources. Consequently, the attackers can easily take advantage of such information asymmetry and imperfectness to cause more damage and to avoid being detected.

To handle incomplete type information, certain attacker detection mechanisms should be applied. In general, one can base on what being observed to detect malicious behavior. For example, if a node has agreed to forward a packet but



(a) P_1 is good, P_2 is bad (b)P_1 is bad, P_2 is good



(c) both players are good (d) both players are bad

Figure 5.2: The payoff profiles under different scenarios.

later drops it, other nodes (either its neighbor or the source of the packet) who has observed this inconsistency (i.e., agreeing to forward but dropping) can mark this node as malicious. If there is no decision execution error and the observation is perfect, such method can detect all intentional packet dropping.

However, noise always exists and the monitoring is impossible to be perfect. Under such realistic circumstances, detecting malicious behavior will become extremely hard due to that an observed misbehavior may either be caused by intention, or by unintentional execution error, or simply due to observation error. Now a node should not be marked as malicious just *simply because* it has been observed dropping some packets. Accordingly, the attackers can take advantage of noise and observation errors to cause more damage without being detected.

5.2.1 Statistical Dropping Packet Attack Detection

To combat insider attacks under noise and imperfect observation, we first study what should be normal observation when no attackers are present. In this case, when a node has made a decision to forward a packet (i.e., decision F), the probability p_F that the outcome observed by the source is also forwarding can be calculated as follows:

$$p_F = (1 - p_e)(1 - p_f) + p_e p_m. \tag{5.4}$$

Let $R_i(j,t)$ denote the number of times that node j has agreed to forward for node i by time t, and $\widetilde{F}_j(i,t)$ denote the number of times that i has observed that j has forwarded a packet for it by time t. Based on the Central Limit Theorem [49], for any $x \in \mathcal{R}^+$ we can have

$$\lim_{R_i(j,t)\to\infty} Prob\left(\frac{\widetilde{F}_j(i,t) - R_i(j,t)\cdot p_F}{\sqrt{R_i(j,t)\cdot(1-p_F)\cdot p_F}} \geq -x\right) = \Phi(x), \tag{5.5}$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}dt. \tag{5.6}$$

In other words, when $R_i(j,t)$ is reasonably large, $\widetilde{F}_j(i,t) - R_i(j,t)p_F$ can be approximately modeled as a Gaussian random variable with mean 0 and variance $R_i(j,t)p_F(1-p_F)$.

Let $isDPA_i(j)$ denote i's belief about whether j has launched dropping packets attack, where $isDPA_i(j) = 1$ indicates that i believes j has launched dropping packets attack, while $isDPA_i(j) = 0$ indicates that i believes j has not launched dropping packets attack. Let B_{th} be a reasonably large constant (e.g., 200). Then the following hypothesis testing rule can be used by i to judge whether j has

maliciously dropped its packets:

$$isDPA_i(j) = \begin{cases} 1 & if\ \widetilde{F}_j(i,t) < R_i(j,t)p_F - x\sqrt{R_i(j,t)p_F(1-p_F)} \ \text{ and } \ R_i(j,t) > B_{th} \\ 0 & otherwise. \end{cases} \tag{5.7}$$

If (5.7) is used to detect dropping packets attack, the false alarm ratio would be no more than $1 - \Phi(x)$. It is worth mentioning that even for a small positive x, the value of $\Phi(x)$ can still approach to 1 (e.g, $\Phi(5) > 0.999$).

5.2.2 Statistical Injecting Traffic Attack Detection

In Section 5.2.1 we focus on dropping packets attacks. Attackers can also try to inject an overwhelming amount of traffic to waste the good nodes' resources. Let $isITA_i(j)$ denote i's belief about whether j has launched injecting traffic attack, where $isITA_i(j) = 1$ indicates that i believes j has launched injecting traffic attack, while $isITA_i(j) = 0$ indicates that i believes j has not launched injecting traffic attack. Let $\widetilde{T}_j(t)$ denote the number of packets that have been injected by j and have been observed by those nodes who know j's legitimate traffic injection rate. Then a simple detection rule can be as follows:

$$isITA_i(j) = \begin{cases} 1 & \text{if} \quad \widetilde{T}_j(t) > f_j(t) \\ 0 & otherwise. \end{cases} \tag{5.8}$$

Under this detection rule, the maximum number of packets that attacker j can inject without being detected will be no more than $f_j(t)/(1-p_s)$. This detection rule is very conservative since only those observed packet injection events are used. If p_s can also be known by good nodes, we can modify (5.8) to further limit the number of packets that j can inject without being marked as malicious, such as changing the threshold from $f_j(t)$ to $\frac{f_j(t)}{1-p_s}$. Since p_s is usually not known and may

change across different nodes, in this chapter when performing injecting traffic attack detection, we will not incorporate p_s into the detection rule.

The detection rule (5.8) can work well only when no retransmission is allowed. Next we show how to detect injecting traffic attack when retransmission is allowed upon unsuccessful delivery. We first make a simple assumption that all selected routes have the same number of hops, denoted by L, and let $q = (1 - p_e)^L$. Then for each packet, the total number of tries needed to successfully deliver this packet to its destination can be modeled as a geometric random variable with mean $\frac{1}{q}$ and variance $\frac{1-q}{q^2}$. For any node $j \in N$, if $p_s = 0$ and j has never intentionally retransmitted a packet that has been successfully delivered to its destination, according to the Central Limit Theorem, for any $x \in \mathcal{R}^+$ we should have

$$\lim_{\widetilde{T}_j(t)\to\infty} Prob\left(\frac{\widetilde{T}_j(t) - T_j(t)/q}{\sqrt{T_j(t)(1-q)/q^2}} \leq x\right) = \Phi(x). \tag{5.9}$$

In other words, when $\widetilde{T}_j(t)$ is reasonably large, $\widetilde{T}_j(t) - T_j(t)/q$ can also be approximately modeled as a Gaussian random variable with mean 0 and variance $T_j(t)(1-q)/q^2$. Then a modified detection rule can be as follows:

$$isITA_i(j) = \begin{cases} 1 & if\ \widetilde{T}_j(t) > \frac{f_j(t)}{q} + \frac{x\sqrt{f_j(t)(1-q)}}{q} \ \text{ and } \ \widetilde{T}_j(t) > B_{th} \\ 0 & otherwise. \end{cases} \tag{5.10}$$

Similarly, when the above detection rule is used, the false alarm ratio would be no more than $1 - \Phi(x)$. In this case, the number of packets that attacker j can inject without being marked as malicious is upper-bounded by $\frac{f_j(t)+x\sqrt{f_j(t)(1-q)}}{q(1-p_s)}$. Comparing to the case that no retransmission is allowed, when retransmission is allowed, attackers can inject more packets without being detected, though good nodes can also enjoy higher throughput.

In general the number of hops per selected route varies according to the loca-

tions of the source and destination and the network topology. Let $\bar{L}_{min}$ denote the average number of hops per selected route. When calculating q used in (5.10), an alternative way is to let $q = (1-p_e)^{\bar{L}_{min}}$. However, this may lead to higher false positive probability since some nodes may experience longer routes due to their locations. In this chapter we adopt a more conservative way by letting $q = (1-p_e)^{L_{max}}$. As a consequence, even when $p_s = 0$, the resulted false positive probability will be far less than $1 - \Phi(x)$, with the penalty that the attackers can also inject more packets without being detected. For example, for $L_{max} = 10$, $\bar{L}_{min} = 4$, $p_e = 0.02$, the extra increase would be about 12.9% (i.e., $(1-p_e)^{\bar{L}_{min}-L_{max}} - 1$). Accordingly, the good nodes' payoff will also be decreased.

5.2.3 Secure Routing and Packet Forwarding Strategy

Based on the above analysis, we can arrive at the following strategy to secure routing and packet forwarding in cooperative ad hoc networks under noise and imperfect monitoring:

Secure Routing and Packet Forwarding Strategy: *In the secure routing and packet forwarding game under noise and imperfect monitoring, initially each good node will assume all other nodes are good. For each single routing and packet forwarding subgame, assuming that P_0 is good and is the source who wants to send a packet to P_n at time t, and a route "$P_0 \rightarrow P_1 \rightarrow \cdots \rightarrow P_n$" has been discovered by P_0. After P_0 has sent requests to all the relays on this route asking them to participate, for each good node on this route the following strategies should be taken in different stages:*

1. *In the route participation stage: A good relay P_i takes action A if and only if no nodes on this route have been marked as malicious and $n \leq L_{max}$;*

otherwise, it takes NA.

2. *In the route selection stage: P_0 will take action R if and only if all the following conditions can be satisfied: a) the packet is valid (i.e., it is scheduled to be sent by P_0), b) $n \leq L_{max}$, c) no nodes on this route have been marked as malicious by P_0, d) all relays have agreed to forward packets in the route participation stage, and e) this route has the minimum number of hops among all good routes to P_n known by P_0; otherwise, P_0 should take action NR.*

3. *In the packet forwarding stage: For each relay P_i, it will take action F if and only if it has agreed to be on this route in the route participation stage; otherwise, it should take action D.*

Let x be a positive constant. For any node j, it will be marked as malicious by node i if it has been detected by any following rules: (5.7), (5.8) if retransmission is not allowed, and (5.10) if retransmission is allowed, where in (5.10) $q = (1 - p_e)^{L_{max}}$. Meanwhile, node j will also be marked as malicious if it has requested to send a packet through a route with the number of hops greater than L_{max}.

In the above defense strategy, each good node needs to know or estimate the following parameters: p_e, p_f, p_m, and L_{max}. Meanwhile, it also needs to set the two constants that are used in (5.7) and (5.10): B_{th} and x. L_{max} is a system-level parameter and is known by all nodes in the network. The packet dropping probability p_e can be either trained off-line, or estimated online by each node through evaluating its own packet dropping ratio. In general different node may experience different p_e at different time or location. Under such circumstances, to reduce the false positive when performing attacker detection using (5.7) and (5.10), a node may set p_e to be a little bit larger than the one experienced by itself. The two observation error related parameters p_f and p_m can be provided by the

underlying monitoring mechanism. Similarly, different node may also experience different p_f and p_m at different situations. Therefore, when a node uses (5.7) to perform attacker detection, to limit the false positive, it may use the upper-bounds of p_f and p_m provided by the underlying monitoring mechanisms. The effects of these parameters will be further studied in later sections.

5.2.4 Attacking Strategy

Since this chapter focuses on insider attackers, it is reasonable to believe that attackers can know the defending strategies employed by the system. This can be regarded as the worst-case scenario from the defenders' point of view. In other words, if the proposed defending strategy can work well in this scenario, it can also work well in any scenarios. This subsection studies what strategies the attackers should use to maximize their utility (or the damage to the network) when the proposed secure routing and packet forwarding strategy is used by the good nodes.

We first study dropping packets attack. According to the proposed secure routing and packet forwarding strategy, once a node i has been marked as malicious by another node j, i will not be able to cause damage to j again. Therefore, an attacker should avoid being detected in order to continuously cause damage to the good nodes. One simple strategy is to always apply the strategy (A, D). However, when applying this strategy, the maximum number of good nodes' packets that an attacker can drop without being detected will be no more than $|N_g| \cdot B_{th}$, while the penalty is that it will be detected as malicious and cannot cause damage to the good nodes any more.

Intuitively, attackers can selectively drop packets to avoid being detected and still cause continuous damage to good nodes. According to the proposed secure

routing and packet forwarding strategy, the number of a good node i's packets that an attacker j can drop without being detected is upper-bounded by $np_e + \frac{x\sqrt{n}}{1+p_m-p_f}$[3], where n is the number of packets that j has agreed to forward for i. In other words, j has to forward at least $n(1-p_e) - \frac{x\sqrt{n}}{1+p_m-p_f}$ packets for i in order to avoid being marked as malicious. However, recall that even there are no attackers, in average $n(1-p_e)$ packets will be dropped due to noise. That is, the extra number of i's packets that j can selectively drop without being detected is upper-bounded by $\frac{x\sqrt{n}}{1+p_m-p_f}$, while the cost needed to forward packets for j is at least $n(1-p_e)c - \frac{x\sqrt{n}}{1+p_m-p_f}c$. Since we have $\lim_{n\to\infty} \frac{\frac{x\sqrt{n}}{1+p_m-p_f}}{n(1-p_e)} = 0$, selectively dropping i's packets can bring almost no gain to j if the game will be played for long enough time.

According to the secure routing and packet forwarding strategy, a good node will not start performing dropping packet attack detection before having enough interactions with another node (e.g., B_{th}). Therefore, the following dropping packet attack strategy can be used by an attacker j when acting as relay nodes: for each good node i, it can drop the first $B_{th}-1$ i's packets by playing (A, D), then start playing (NA, D) forever. With this strategy, the damage that j can cause to i is upper-bounded by $B_{th}c$ without introducing any cost to j. It is easy to see that the relative damage (normalized by time) $\frac{B_{th}c}{t_f}$ decreases monotonically with the increase of the network lifetime t_f.

Until now we have assumed that all nodes will experience the same p_e, p_f, and p_m. However, such assumption may not hold in general. For example, attackers may be able to decrease p_f and/or increase p_m experienced by it. Let p'_f and p'_m

[3]It is ready to check that $(np_e + \frac{x\sqrt{n}}{1+p_m-p_f})p_m + (n(1-p_e) - \frac{x\sqrt{n}}{1+p_m-p_f})p_f = np_F - x\sqrt{n} < np_F - x\sqrt{np_F(1-p_F)}$.

denote the actual false positive probability and miss probability experienced by an attacker j. When j tries to drop i's packets, in order to avoid being detected, the actual packet drop ratio p'_e that j will apply to drop i's packet should satisfy the following condition:

$$(1-p'_e)(1-p'_f)+p'_e p'_m > p_F - \frac{x\sqrt{np_F(1-p_F)}}{n}, \tag{5.11}$$

where n is the number of packets that j has agreed to forward for i. It is easy to check that to satisfy (5.11) for all possible n, the maximum packet dropping ratio p'_e that j can apply is upper-bounded by

$$p'_e \leq \frac{p_e(1-p_f-p_m)+(p_f-p'_f)}{1-p'_f-p'_m} \tag{5.12}$$

From (5.12) we can see that increasing the miss probability p'_m and/or decreasing the false positive probability experienced by j can also increase p'_e, and consequently increase the damage to i. Let L_{avg} denote the average number of wasted packet transmissions caused by j when it drops i's packets, according to the payoff definition (5.3), as long as $\frac{p'_e-p_e}{1-p_e} \leq \alpha L_{avg}$, launching dropping packet attack with p'_e can introduce no gain to j. However, if $\frac{p'_e-p_e}{1-p_e} > \alpha L_{avg}$, j should launch dropping packet attacks by selectively dropping the good nodes' packets with dropping probability calculated based on (5.12).

Now we study injecting traffic attack. According to the secure routing and packet forwarding strategy, to avoid being marked as launching injecting traffic attack, an attacker j should be sure that $\widetilde{T}_j(t) \leq f_j(t)$. However, j may not know the exact value of $\widetilde{T}_j(t)$, and needs to estimate $\widetilde{T}_j(t)$ by itself. Recall that for each packet injected by j, with probability p_s it can avoid being detected. It is readily to show that $\widetilde{T}_j(t) - F_j(j,t)(1-p_s)$ can be approximately modeled as a Gaussian random variable with mean 0 and variance $F_j(j,t)p_s(1-p_s)$, where $F_j(j,t)$ is the total number of packets injected by j until time t.

Based on the above analysis, when no retransmission is allowed, a good injecting traffic strategy is as follows: j should try to limit the number of injected packets $F_j(j,t)$ to satisfy the following condition:

$$F_j(j,t)(1-p_s) + y\sqrt{F_j(j,t)p_s(1-p_s)} < f_j(t), \tag{5.13}$$

where y is a large positive constant. By using this strategy, the probability that j will be detected is upper-bounded by $1-\Phi(y)$. When retransmission is allowed, according to the secure routing and packet forwarding strategy, the condition should be changed as follows:

$$F_j(j,t)(1-p_s) + y\sqrt{F_j(j,t)p_s(1-p_s)} < \frac{f_j(t) + x\sqrt{f_j(t)(1-q)}}{q}, \tag{5.14}$$

where y is a large positive constant and x and q are defined in the secure routing and packet forwarding strategy.

In summary, we can arrive at the following attacking strategy, referred to as **optimal attacking strategy:**

1. **Dropping packet attack:** *For any attacker j, if the maximum possible p'_e calculated using (5.12) is larger than p_e and $\frac{p'_e - p_e}{1-p_e} \leq \alpha L_{avg}$, it should try to selectively drop the good nodes' packets with probability p'_e; otherwise, it should apply the following strategy: for any good node i, j should try to drop the first $B_{th} - 1$ i's packets by playing (A, D), then start playing (NA, D) forever when acting as relay node for i.*

2. **Injecting traffic attack:** *For any attacker j, if no retransmission is allowed, it should try to inject traffic by following (5.13); otherwise, it should try to inject traffic by following (5.14). Meanwhile, when j has decided to inject a packet, it should pick a route with the following properties: a) the*

number of hops is no more than L_{max}, b) all relays are good nodes, c) among all the routes known by j which satisfy (a) and (b), this route has the maximum number of hops.

5.3 Optimality Analysis

In this section we analyze the optimality of the proposed strategy profile, where all good nodes follow the strategy described in Section 5.2.3 and all attackers follow the strategy described in Section 5.2.4. We will focus on the worst-case scenario from the good nodes' point of view: when a malicious node wants to send a packet to another node, it can always find a route with L_{max} hops and all relay nodes being good. This also is the best-case scenario from the attackers' point of view. We focus on the scenario that all nodes experience the same p_e, p_f, and p_m. The scenario that different node will experience different p_e, p_f, and p_m will be discussed at the end of this section.

Theorem 5.3.1 *In the secure routing and packet forwarding game in noiseless environment with perfect observation (i.e., $p_e = p_f = p_m = p_s = 0$), the proposed strategy profile with $B_{th} = 1$ form a Nash equilibrium.*

Proof To show that the proposed strategy profile forms a Nash equilibrium, we only need to show that no player can increase its payoff by unilaterally changing its own strategy:

- **P_0's actions when it is good**: According to the secure routing and packet forwarding strategy, P_0 will take action R if and only if 1) the packet to be sent is valid, 2) $n \leq L_{max}$, 3) no nodes on this route have been marked as malicious by P_0, 4) all relay nodes have agreed to be on this route, and

5) this route has the minimum cost among all good routes to P_n known by P_0. First, if P_0 takes action R when the packet to be sent is not valid, the good nodes' payoff cannot be increased, or may even be decreased. Second, if P_0 takes action R when $n > L_{max}$, P_0 will be marked as malicious by other good nodes and cannot send any packets again, which will decrease the good nodes' payoff. Third, if P_0 takes action R when some nodes have been marked as malicious by P_0 or some nodes do not agree to be the route, then the packet will be dropped by a certain relay node, and consequently all cost spent to transmit this packet will be wasted, and the good nodes' payoff will be decreased. Fourth, if P_0 takes action R when the selected route does not have the minimum cost among all good routes to P_n known by P_0, then comparing to the situation that the good route with the minimum cost is used, some extra cost will be wasted if this route is used instead, which will decrease the good nodes' payoff. Finally, if all the above conditions are satisfied but P_0 takes action NR, the good nodes' payoff will not increase too, since not sending the packet or sending the packet using non-minimum cost route can bring no gain or can only bring less gain.

- P_0**'s decision when it is malicious**: According to the optimal attacking strategy, P_0 will take action R if and only if 1) $F_{P_0}(P_0, t) < f_{P_0}(t)$, 2) $n = L_{max}$, 3) all relay nodes are good, and 4) all relay nodes have agreed to be on this route. First, if P_0 takes action R when $F_{P_0}(P_0, t) \geq f_{P_0}(t)$ or $n > L_{max}$, P_0 will be marked as malicious by good nodes and cannot inject any packets again, which will surely decrease the attackers' payoff. Second, if P_0 takes action R when $n < L_{max}$ or some relay nodes are malicious or some relay nodes do not agree to be on this route, since P_0 can always find a route with

L_{max} hops and with all relay nodes being good, using a suboptimal route surely cannot increase P_0's attack efficiency. Third, if all those conditions are satisfied but P_0 takes action NR, since the maximum possible damage that can be caused by each packet injecting is $(L_{max} - 1)c$, the attackers' payoff cannot be further increased either.

- P_i**'s decision** $(0 < i < n)$ **when it is good**: According to the secure routing and packet forwarding strategy, P_i will take action (A, F) if all the other nodes on this route have not been marked as malicious by it and $n \leq L_{max}$; otherwise, it will take action (NR, D). When no nodes on this route have been marked as malicious by it and $n \leq L_{max}$, since refusing to be on this route may cause the source to select a route with higher cost and dropping packet will waste other good nodes' cost, both will cause P_i' payoff to be decreased. When some nodes on this route have been marked as malicious by P_i or $n > L_{max}$, if P_i agrees to be on this route or does not drop the packet, since the packet will finally be dropped by malicious node, all effort that has been spent by good nodes in this subgame will be wasted, which surely cannot increase P_i's payoff either.

- P_i**'s decision** $(0 < i < n)$ **when it is malicious**: According to the optimal attacking strategy, P_i will always take action (NA, D). We first consider the situation that P_0 is good. If P_i takes action (A, D), it will be detected as malicious immediately and cannot cause damage to P_0 any more, which surely cannot increase the attackers' payoff. If P_i takes action (A, F), this can only contribute to good nodes by helping good nodes forward packets, and cannot increase the attackers' payoff. Meanwhile, taking action (NA, F) surely cannot cause damage to the good nodes, since good nodes will not use

P_i to forward packets. Now let's consider the situation that the initiator P_0 is malicious. It is also easy to check that taking action (NA, D) is always a best strategy from the malicious nodes' point of view since P_0 can always find a better route, that is, a route with L_{max} hops and with all relay nodes being good.

Based on the above analysis we can see that no player can increase its payoff by unilaterally changing its own strategy. ■

Now we analyze nodes' possible payoff under the proposed strategy profile. Let $f_i^{avg} = \frac{f_i(t_f)}{t_f}$ when t_f is finite, and $f_i^{avg} = \lim_{t\to\infty} \frac{f_i(t)}{t}$ when t_f is infinite. According to the secure routing and packet forwarding strategy, a good node will not work with any node that has been marked as malicious by itself. First, as we have shown in Section 5.2.4, playing (A, D) cannot increase the attackers' payoff provided t_f is infinite. Second, it is easy to see that playing (NA, F) and (A, F) cannot increase the attackers' payoff either, since when an attacker plays (NA, F), no good nodes will request it to forward packets, while when an attacker plays (A, F), it can only make contribution to the good nodes. Third, when an attacker tries to inject packets, similar to the analysis in the proof of Theorem 5.3.1, it should always use the route with all relay nodes being good and having agreed to be on the route. Meanwhile, from an attacker's point of view, injecting more packets than specified will make it to be marked as malicious and cannot cause any more damage to the good nodes, and consequently decrease its payoff. Therefore, when no retransmission is allowed, based on (5.3), the attackers' payoff will be

upper-bounded by

$$\begin{aligned} U_m &\leq \lim_{t_f \to \infty} \frac{1}{t_f} \sum_{i \in N_m} \left(\frac{f_i(t_f)}{1-p_s}(L_{max} - 1 - \alpha)c + |N_g| B_{th} L_{avg} c \right) \\ &= \sum_{i \in N_m} \frac{f_i^{avg}}{1-p_s}(L_{max} - 1 - \alpha)c. \end{aligned} \tag{5.15}$$

Here $\frac{f_i(t_f)}{1-p_s}$ is the number of packets that attacker i can inject to the network by time t_f without being marked as malicious, $(L_{max} - 1)c$ is the maximum possible damage that can injected packet can cause to good nodes, αc is the cost incurred to attackers by forwarding a packet, and $|N_g| B_{th} L_{avg} c$ is the damage that j can caused by launching dropping packet attack.

When retransmission is allowed upon unsuccessful delivery, from the attackers' point of view, the only difference is that they can inject more packets without being detected. Now the attackers' payoff will be upper-bounded by

$$\begin{aligned} U_m &\leq \lim_{t_f \to \infty} \frac{1}{t_f} \sum_{i \in N_m} \left(\frac{f_i(t_f)}{(1-p_s)q} + \frac{x\sqrt{f_i(t_f)(1-q)}}{q} \right) (L_{max} - 1 - \alpha)c \\ &\quad + \lim_{t_f \to \infty} \frac{|N_m||N_g| B_{th} L_{avg} c}{t_f} \\ &= \sum_{i \in N_m} \frac{f_i^{avg}}{(1-p_s)q}(L_{max} - 1 - \alpha)c \end{aligned} \tag{5.16}$$

Now we analyze the good nodes' payoff. Recall that $\bar{L}_{min}$ denotes the average number of hops among those routes selected by good nodes. We first consider the situation that the environment is noisy and no retransmission is allowed. In this case, some good nodes' packets will be dropped due to noise, and $\lim_{t \to \infty} \frac{S_i(t)}{T_i(t)} = (1-p_e)^{\bar{L}_{min}}$. According to (5.1), for each $i \in N_g$, $F_i(t)$ comes from two parts: forwarding packets for the good nodes and forwarding packets for the attackers. The total number of packets that the good nodes have forwarded for themselves is $\sum_{i \in N_g} T_i(t) \bar{L}_{min}$ by time t, and the total number of packets that the good

nodes have forwarded for the attackers is no more than $\sum_{i\in N_m}\frac{f_i(t)}{(1-p_s)}(L_{max}-1)$. Meanwhile, for any given positive value x adopted in the secure routing and packet forwarding strategy, the overall false positive probability will be upper-bounded by $1-\Phi(x)$, that is, at most $1-\Phi(x)$ percentage of good nodes will be mistakenly marked as malicious. Let $T_i^{avg}=\frac{T_i(t_f)}{t_f}$ when t_f is finite and $T_i^{avg}=\lim_{t\to\infty}\frac{T_i(t)}{t}$ when t_f is infinite. Then the good nodes' payoff will be lower-bounded by

$$\begin{aligned} U_g &\geq \lim_{t\to\infty}\frac{\sum_{i\in N_g}(S_i(t)g-T_i(t)\bar{L}_{min}c)-\sum_{j\in N_m}\frac{f_j(t)}{(1-p_s)}(L_{max}-1)c}{\sum_{i\in N_g}T_i(t)} \\ &= \Phi(x)g(1-p_e)^{\bar{L}_{min}}-\left(\bar{L}_{min}+\frac{\sum_{j\in N_m}f_j^{avg}\cdot(L_{max}-1)}{(1-p_s)\sum_{j\in N_g}T_i^{avg}}\right)c. \end{aligned} \quad (5.17)$$

When the environment is noiseless or when the retransmission is allowed, all good nodes' packets can be successfully delivered to their destinations with $\lim_{t\to\infty}\frac{S_i(t)}{T_i(t)}=1$ for $i\in N_g$. Meanwhile, the total number of packets that the good nodes have forwarded for themselves by time t is no more than $\sum_{i\in N_g}\frac{T_i(t)}{(1-p_e)^{\bar{L}_{min}}}\bar{L}_{min}$, and the total number of packets that the good nodes have forwarded for the attackers is no more than $\sum_{i\in N_m}\frac{f_i(t)}{q(1-p_s)}(L_{max}-1)$. Thus in this case the good nodes' payoff can be lower-bounded by

$$U_g\geq\Phi(x)g-\left(\frac{\bar{L}_{min}}{(1-p_e)^{\bar{L}_{min}}}+\frac{\sum_{j\in N_m}f_j^{avg}\cdot(L_{max}-1)}{q(1-p_s)\sum_{j\in N_g}T_i^{avg}}\right)c. \quad (5.18)$$

On the other hand, when the proposed optimal attacking strategy is used by attackers, from the good nodes' point of view, when no retransmission is allowed, the maximum possible payoff can also be upper-bounded by

$$U_g\leq g(1-p_e)^{\bar{L}_{min}}-\left(\bar{L}_{min}+\frac{\sum_{j\in N_m}f_j^{avg}\cdot(L_{max}-1)}{(1-p_s)\sum_{j\in N_g}T_i^{avg}}\right)c. \quad (5.19)$$

While when retransmission is allowed, the maximum possible payoff can also be upper-bounded by

$$U_g\leq g-\left(\frac{\bar{L}_{min}}{(1-p_e)^{\bar{L}_{min}}}+\frac{\sum_{j\in N_m}f_j^{avg}\cdot(L_{max}-1)}{q(1-p_s)\sum_{j\in N_g}T_i^{avg}}\right)c. \quad (5.20)$$

From the above payoff analysis we can see that the good nodes' payoff can be lower-bounded by certain value, no matter what strategies the attackers use and what kind of goals the attackers have. In other words, the attackers' goal has little effect on good nodes' payoff when the proposed secure routing and packet forwarding strategy is used by good nodes. From the above payoff analysis we can also see that as long as the gain g is reasonably large, it will not play an important role in the strategy design.

Theorem 5.3.2 *In the infinite duration secure routing and packet forwarding game under noise and imperfect observation, the proposed secure routing and packet forwarding strategy is asymptotically optimal from the good nodes' point of view in the sense that for any $\epsilon > 0$, we can always find a $x^* > 0$ such that no other equilibrium strategies can further increase the good nodes' payoff by more than ϵ as long as the attackers also play optimally.*

Proof We first consider the situation that no retransmission is allowed. Based on the above analysis we can see that from the attackers' point of view, to maximize their payoff, the optimal attacking strategy is to inject no more packets to the network than they are allowed and will not forward any packet for the good nodes. In this case the good nodes' maximum possible payoff is defined in (5.19). According to (5.17), the difference between the actual payoff and maximum possible payoff is $(1 - \Phi(x))(1 - p_e)^{\bar{L}_{min}} g$. Since $\Phi(x) \to 1$ as $x \to \infty$, for any $\epsilon > 0$, we can always find a constant x^* such that the actual payoff is within ϵ of the maximum possible payoff. Similarly, we can also prove this under the situation that retransmission is allowed. ■

Theorem 5.3.3 *In the infinite duration secure routing and packet forwarding*

game, the proposed strategy profile is strongly Pareto optimal[4].

Proof To show the proposed strategy profile is strongly Pareto optimal, we only need to show that no other strategy profiles can further increase some players' payoff without decreasing any other player's payoff.

We first show that the good nodes' payoff cannot be further increased without decreasing the attackers' payoff. According to (5.17), to further increase the good nodes' payoff, one can either decrease $\bar{L}_{min}$, or decrease f_j^{avg}. First, since the minimum-hop routes have been used, $\bar{L}_{min}$ cannot be further decreased. Second, according to (5.3) and (5.15), decreasing f_j^{avg} always decreases the attackers' payoff.

Next we show that the attackers' payoff cannot be further increased without decreasing the good nodes' payoff. According to (5.3), to increase the attackers' payoff, one can either try to increase $\sum_{i\in N_m, j\in N_g} W_i(j,t)$ and $\sum_{i\in N_m, j\in N_g} F_j(i,t)$, or try to decrease $\sum_{i\in N_m} F_i(t)$. First, $\sum_{i\in N_m, j\in N_g} F_j(i,t)$ comes completely from injecting traffic attacks, which has been maximized and cannot be further increased. Since $\sum_{i\in N_m, j\in N_g} W_i(j,t)$ comes from launching dropping packet attacks, increasing $\sum_{i\in N_m, j\in N_g} W_i(j,t)$ will also decrease the good players' payoff. Now we consider $\sum_{i\in N_m} F_i(t)$. According to the above packet forwarding strategy, attacker i will not forward packets for others, so $F_i(t)$ comes totally from transmitting packets for itself. Therefore, $F_i(t)$ cannot be further decreased without decreasing the attackers' payoff. ■

[4]A strategy profile is said to be Pareto optimal if there is no other strategy profile which can simultaneously increase all players' payoff; a strategy profile is said to be strongly Pareto optimal if there is no other strategy profile which can increase at least one player' payoff without decreasing any other players' payoff [64].

Until now we have focused on the scenario that p_e, p_f, and p_m keep being the same for all nodes at all times. However, as we have mentioned, this may not hold in general. Next we study the consequence when different nodes may experience different p_e, p_f, and p_m. First, from the good nodes' point of view, such variation may increase false positive probability when performing attacker detection. For example, for a node experiencing lower packet dropping ratio, when it uses this ratio to perform dropping packet attacker detection, with much higher probability those nodes experiencing higher packet dropping ratio can be mistakenly marked as malicious (e.g., higher than $1 - \Phi(x)$). As mentioned in Section 5.2.3, to avoid high false positive probability, a good node may need to set a higher p_e than the one experienced by itself when performing attacker detection. Meanwhile, a good node may also need to increase B_{th} and x to handle possible bursty packet dropping effect, which is normal in wireless networks due to fading. Similarly, when nodes experience different p_f and p_m, a good node may need to use the upper-bounds of p_f and p_m to avoid high false positive probability when performing attacker detection. As a penalty, these variations can be taken advantage of by attackers to inject more packets and drop more packets without being marked as malicious, which consequently leads to the decrease of good nodes' performance. However, our simulation studies indicate that even in such realistic scenarios, the proposed secure routing and packet forwarding strategy can still work very well.

5.4 Simulation Studies

We have conducted a series of simulations to evaluate the performance of the proposed strategies in both static and mobile ad hoc networks. In each ad hoc network, nodes are randomly deployed inside a rectangular area of 1000m $\times$ 1000m. For

Table 5.1: Mobility patterns

Pattern 1:	$v_{max} = 10$m/s, $v_{min} = 1$m/s, pause time = 500 s
Pattern 2:	$v_{max} = 15$m/s, $v_{min} = 5$m/s, pause time = 300 s
Pattern 3:	$v_{max} = 15$m/s, $v_{min} = 5$m/s, pause time = 100 s
Pattern 4:	$v_{max} = 30$m/s, $v_{min} = 10$m/s, pause time = 100 s

mobile ad hoc networks, nodes move randomly according to the *random waypoint* model. The following physical layer model is used: two nodes can directly communicate with each other only if they are in each other's transmission range, but it can be easily extended to more realistic model where the error probability is a function of distance. Based on the above models, the static ad hoc networks can be regarded as the noiseless case, while the mobile ad hoc networks can be regarded as the noisy case where the decision execution error (i.e., the decision is F but the outcome is D) is only caused by link breakage. For each node, the transmission power is fixed, and the maximum transmission range is 200m.

In the simulations, each good node will randomly pick another good node as the destination. Similarly, each attacker will also randomly pick another attacker as the destination. In both cases, packets are scheduled to be sent to this destination according to a constant rate. The total number of good nodes is set to be 100 and the total number of attackers varies from 0 to 40. For each good or malicious node, the average packet inter-arrival time is 1 second, that is, $T_i(t) = \lfloor t \rfloor$ for any time t and any node $i \in N$. Further, each good node $i \in N_g$ will set $f_i(t) = \lfloor t \rfloor + 2$ for any other node $i \in N$. All data packets have the same size.

Since the link breakage ratio p_e plays an important role in the strategy design, we first study the characteristics of link breakages in mobile ad hoc networks under different mobility patterns. In this set of simulations only good nodes will

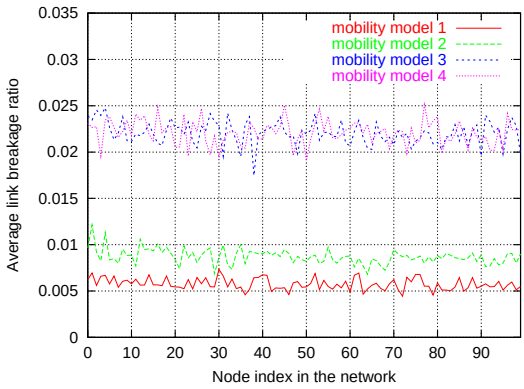
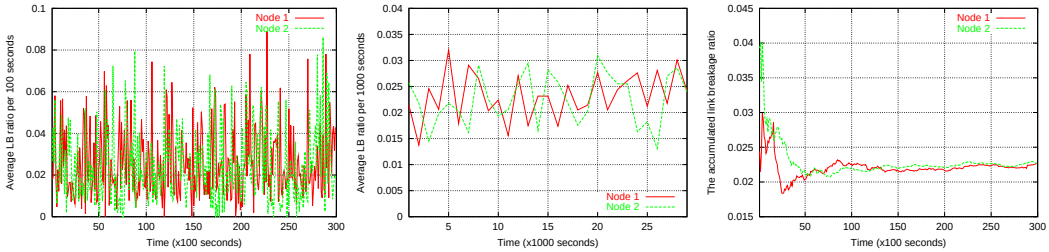


Figure 5.3: The average link breakage ratio in mobile ad hoc networks



(a) (b) (c)

Figure 5.4: The evolution of p_e in mobile ad hoc networks

be considered. For each node, the average link breakage ratio experienced by it is calculated as the ratio between the total number of link breakages it experienced as the transmitter and the total number of packet transmissions it has tried as the transmitter. The total simulation time is 30000s. Fig. 5.3 shows the link breakage ratios experienced by different nodes under four different mobility patterns listed in Table 5.1. First, from these results we can see that the average link breakage ratio will change under different mobility patterns. Second, under the same mobility pattern, the average link breakage ratio experienced by each node is almost the same.

Fig. 5.4(a)-(c) show the evolution of the average link breakage ratios over time

when mobility pattern 4 is used. In this set of simulations, 2 nodes are randomly selected among the 100 nodes in the network. Fig. 5.4(a) shows the link breakage ratio averaged over every 100 seconds, Fig. 5.4(b) shows the link breakage ratio averaged over every 1000 seconds, and Fig. 5.4(c) shows the accumulated average link breakage ratio. From these results we can see that the link breakage ratio experienced by each node may vary dramatically in a short period, but will become stable in a long period. These results suggest that when performing attacker detection, if t_f is not large enough, p_e should be set higher than the long-term average to avoid high false positive probability, while if t_f is large or goes to infinity, the average link breakage ratio can be used when performing attacker detection, with a reasonably large B_{th}.

Now we study the performance of the proposed strategies in different scenarios. We use "noiseless scenario" to denote static ad hoc networks, and use "noisy scenario" to denote mobile ad hoc networks. In both cases, all good nodes follow the secure routing and packet forwarding strategy described in Section 5.2.3, and all (insider) attackers follow the optimal attacking strategy described in Section 5.2.4 with the only modification being that no attacker will intentionally drop packets. The total simulation time t_f is set to be 10000 seconds, and all results are averaged over 20 independent rounds. The following parameters are used: $g = 20$, $c = 1$, $\alpha = 1$, $L_{max} = 10$, $p_f = 0.05$, $p_m = 0.05$, $p_s = 0.05$. The acceptable false alarm ratio is set to be 0.1%. For mobile ad hoc networks, the mobility pattern 4 listed in Table 5.1 is used. Since t_f is not very large, p_e is set to be 3%, which is obtained through off-line training. For static ad hoc networks, we focus on the case that the attackers can always find routes with L_{max} hops to inject packets. For mobile ad hoc networks, four scenarios are considered, as listed in Table 5.2, and DSR [47] is

Table 5.2: Noisy scenarios

Scenario 1:	Retransmission is allowed, and attackers can always find a L_{max}-hop route with all relays good.
Scenario 2:	No retransmission is allowed, and attackers can always find a L_{max}-hop route with all relays good.
Scenario 3:	Retransmission is allowed, and attackers may not find a L_{max}-hop route with all relays good.
Scenario 4:	No retransmission is allowed, and attackers may not find a L_{max}-hop route with all relays good.

used as the underlying routing protocol to perform route discovery. The simulation results are illustrated in Fig. 5.5.

Fig. 5.5(a) compares the good nodes' payoff under different scenarios. First, we can see that when no attackers are present, the noiseless scenario has the highest payoff, and the noisy scenario 2 & 4 (no retransmission is allowed upon unsuccessful packet delivery) have the lowest payoff. The reason is that the good nodes' payoff is determined not only by their transmission cost, but also by the packet delivery ratio. Under noisy environments, when no retransmission is allowed upon unsuccessful packet delivery, the packet delivery ratio will also be decreased, as illustrated in Fig. 5.5(a), where in this case the packet delivery ratio is only about 89% (illustrated in Fig. 5.5(c)). Second, we can see that the allowance of retransmission upon unsuccessful packet delivery can increase the good nodes' payoff in these scenarios (noisy scenario 1 vs. noisy scenario 3, and noisy scenario 2 vs. noisy scenario 4). However, with the increase of the number of attackers, the performance gap between the two scenarios (with or without retransmission) will also decrease (noisy scenario 1 vs. noisy scenario 2, and noisy scenario 3 vs.

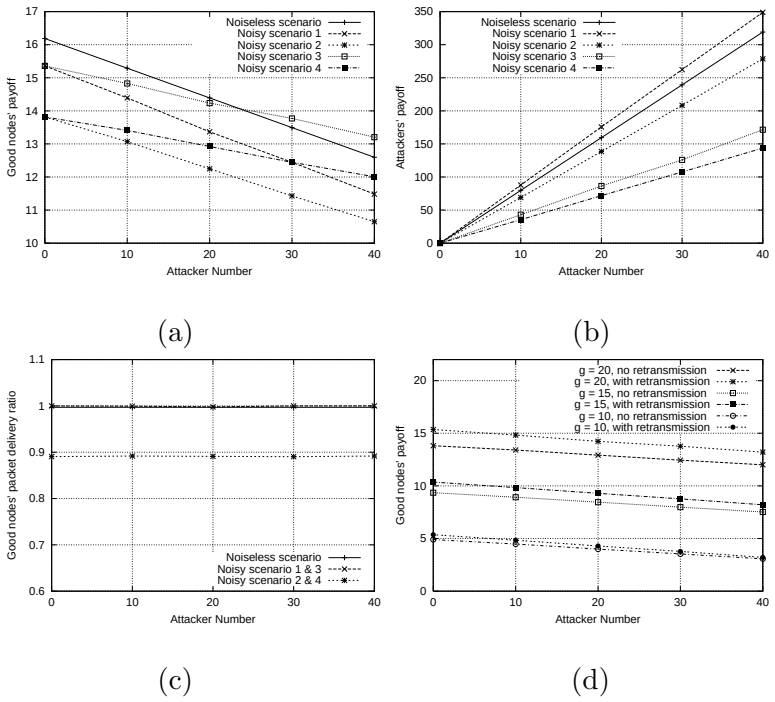


Figure 5.5: Payoff comparison when no attackers will drop packets

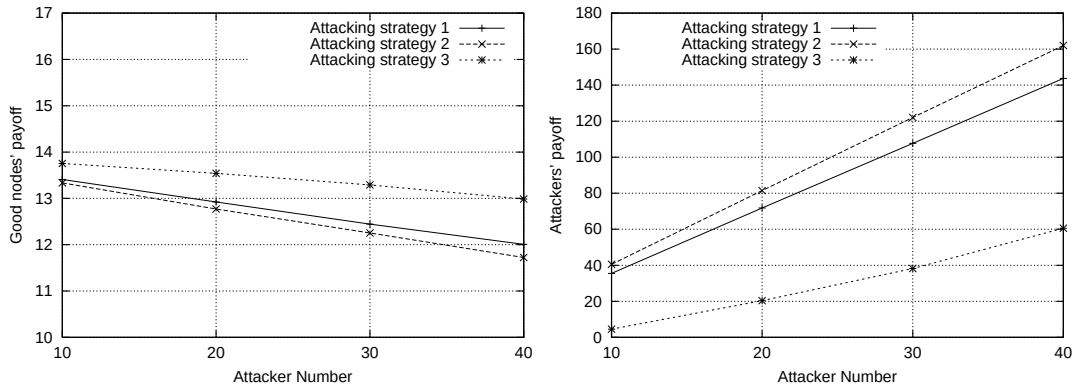
noisy scenario 4). Third, in general, noise will decrease the good nodes' payoff, however, the noisy scenario 3 can achieve higher payoff than the noiseless scenario when the attacker number is no less than 30. The reason is that in the noiseless scenario, attackers can always find L_{max}-hop routes, while in the noisy scenario 3, the average hop number per route selected by the attackers is much less than L_{max}, and the caused damage is less than in noiseless scenario.

Fig. 5.5(b) demonstrates the attackers' payoff under different scenarios. First, as shown in the case of noisy scenario 3 & 4, when the attackers cannot always use L_{max}-hop routes to inject packets, their payoff will be decreased a lot comparing to the cases that they can, as shown in the case of noisy scenario 1 & 2. Second, the allowance of retransmission upon unsuccessful packet delivery can also increase the attackers' payoff, since now more packets can be injected by the attackers. Third, since the attackers' packets may also be dropped under the noisy scenar-

ios, without allowing retransmission, the attackers' payoff will also be decreased comparing to the noiseless scenario, as shown by the noisy scenario 2. However, when retransmission is allowed, comparing to the noiseless scenario, the attackers' payoff can still be increased even under the noisy scenarios, as illustrated by the noisy scenario 1.

Finally, Fig. 5.5(d) illustrates the good nodes' payoff under different g values, where now only the noisy scenario 3 & 4 are considered. First, from these results we can see that with the increase of the number of attackers, the performance gap between these two scenarios will also decrease. The reason is that the attackers can take advantage of retransmission to cause more damage to the good nodes. Second, with the decrease of g, the performance gap between these two scenarios will also decrease. For example, when $g = 10$ and the number of attackers is 40, there is almost no difference. In summary, the gain introduced by the allowance of retransmission becomes less and less with the increase of the number of attackers or with the decrease of g. However, it is worth mentioning that g does not change the underlying strategy design as long as it is reasonably large.

Thus far we have only considered the situations that no attackers will intentionally drop packets. Next we study the situation when the attackers will also try to drop the good nodes' packets. In this set of simulations, three attacking strategies will be studied: in "attacking strategy 1", no attackers will intentionally drop the good nodes' packets. In "attacking strategy 2", each attacker will only drop the first B'_{th} packets for any good node that has requested it to forward, then will stop participating route discoveries initiated by that good node, where dropping B'_{th} packets will not be detected as malicious. In these simulations, we set $B'_{th} = 20$. In "attacking strategy 3", each attacker will always keep participating the route



(a) (b)

Figure 5.6: Payoff comparison when some attackers will drop packets

discoveries initiated by the good nodes and will drop the good nodes' packets in such a way that it will not be detected as malicious, which can be regarded as selective dropping.

Fig. 5.6(a) illustrates the good nodes' payoff under different attacks. First, comparing to the attacking strategy 1, the attacking strategy 3 even increases the good nodes' payoff, though the attackers can drop some good nodes' packets. The reason is that when the attacking strategy 3 is used, the attackers also need to keep forwarding packets for the good nodes, which will increase the number of nodes that the good nodes can use and reduce the value of $\bar{L}_{min}$. Since the number of packets that the attackers can drop without being detected as malicious is very limited, the extra damage that they can cause is also very limited, and the good nodes' payoff will be increased consequently. Second, comparing to the attacking strategy 1, the attacking strategy 2 can decrease the good nodes' payoff a little bit due to the extra number of packets that they have dropped. However, since the number of packets that the attackers can drop is always bounded, with the increase of time, the effect of such packet dropping becomes less and less noticeable.

Fig. 5.6(b) illustrates the attackers' payoff. First, attacking strategy 2 can increase the attackers' payoff comparing to attacking strategy 1. The reason is that the attackers can drop some extra packets without being detected when attacking strategy 2 is used. However, attacking strategy 3 can dramatically decrease the attackers' payoff comparing to attacking strategy 1, the reason is that forwarding packets for the good nodes will also incur a lot of cost, while the number of packets that they can drop without being detected as malicious is very limited. In summary, from the attackers' point of view, when the network lifetime is finite, attacking strategy 2 should be used, while its advantage over attacking strategy 1 is very limited, and will decrease with the increase of network lifetime.

5.5 Summary

In this chapter we have formally investigated how to secure cooperative ad hoc networks against insider attacks under realistic scenarios. We model the dynamic interactions between good nodes and attackers in such networks as securing routing and packet forwarding game. The optimal defense strategies have been devised, which are optimal in the sense that no other strategies can further increase the good nodes' payoff under attacks. The maximum possible damage that can be caused by the attackers have also been analyzed. By focusing on the worst-case scenario from the good nodes' point of view, that is, the good nodes have no prior knowledge of the other nodes' types while the insider attackers can know who are good nodes, the devised strategies can work well under any scenarios. Extensive simulations have also been conducted, which demonstrate that the proposed defending strategies can effectively secure cooperative ad hoc networks under noise and imperfect monitoring.

Chapter 6

Attack-Resistant Cooperation Stimulation in Autonomous Ad Hoc Networks

In Chapter 3, Chapter 4, and Chapter 5 we have mainly focused on designing defense mechanisms to secure cooperative ad hoc networks. From now on we will focus on designing attack-resistant cooperation strategies for autonomous ad hoc networks where nodes belong to different authorities and pursue different goals. In this chapter we will present a reputation-based self-organized system for autonomous ad hoc networks such that cooperation among selfish nodes can be effectively stimulated even under attacks. This chapter is organized as follows. Section 6.1 describes the system model and formulates the problem. Section 6.2 describes the proposed ARCS system. Section 6.3 presents the performance analysis of the system under various attacks. Simulation studies are presented in Section 6.4. Finally Section 6.5 summarizes the contribution of this chapter.

6.1 System Model

We consider autonomous ad hoc networks where nodes belong to different authorities and have different goals. We assume that each node is equipped with a battery with limited power supply, and may act as a service provider: packets are scheduled to be generated and delivered to certain destinations with each packet having a specific delay constraint. If a packet can be successfully delivered to its destination within the specified delay constraint, the source of the packet will get some payoff, otherwise, it will be penalized.

According to their objectives, the nodes in such networks can be classified into two types: selfish and malicious. The objective of selfish nodes is to maximize the payoff they can get using their limited resources, and the objective of malicious nodes is to maximize the damage that they can cause to the network. Since energy is usually the most stringent and valuable resource for battery-supplied nodes in ad hoc networks, we restrict the resource constraint to the energy. However, the proposed schemes are also applicable to other types of resource constraints. For each node in the network, the energy consumption may come from many aspects, such as processing, transmitting, and receiving packets. In this chapter, we focus on the energy consumed in communication-related activities. We focus on the situation that all nodes in the network are legitimate, no matter selfish or malicious.

Before formulating the problem, we first introduce some notations to be used, as listed in Table 6.1. We assume that all data packets have the same size, and the transmitting power is the same for all nodes. We use "packet delivery transaction" to denote sending a packet from its source to its destination. We say a transaction is "successful" if the packet has successfully reached its destination within its delay constraint; otherwise, the transaction is "unsuccessful".

Table 6.1: Notations used in the problem formulation

E	The energy needed to transmit and receive a data packet and a receipt.
$E_{S,max}$	S's total available energy when it enters the network.
α_S	The payoff that source S gets for each successfully delivered data packet.
β_S	The penalty that source S receives for each unsuccessfully delivered data packet.
E_S	The amount of energy that S has spent until now.
$N_{S,succ}$	# successful data packet deliveries until now with S being the source.
$N_{S,fail}$	# unsuccessful data packet deliveries until now with S being the source.
$E_{S,waste}$	The energy that has been wasted until now due to S's its malicious behavior.
$E_{S,contribute}$	The energy that S has spent until now on successfully transmitting packets for others.

For each node S, if it is selfish, its total profit $P_{rofit}(S)$ is defined as follows:

$$P_{rofit}(S) = \alpha_S N_{S,succ} - \beta_S N_{S,fail}. \tag{6.1}$$

Then the objective of each selfish nodes S can be formulated as follows:

$$\max P_{rofit}(S) \quad \text{s.t.} \quad E_S \leq E_{S,max}. \tag{6.2}$$

If S is malicious, then the total damage D_S that S has caused to other selfish nodes until the current moment is calculated as

$$D_S = E_{S,waste} - E_{S,contribute}. \tag{6.3}$$

Since in the current system model malicious nodes are allowed to collude, in this chapter we only formulate the overall objective of malicious nodes, which is as follows:

$$\max \sum_{\text{S is malicious}} D_S. \tag{6.4}$$

Table 6.2: Records kept by node S

$C_{redit}(A, S)$	The energy that A has spent until now on successfully transmitting packets for S.
$D_{ebit}(A, S)$	The energy that S has spent until now on successfully transmitting packets for A.
$W_{by}(A, S)$	The wasted energy that A has caused to S until now.
$W_{to}(A, S)$	The wasted energy that S has caused to A until now.
$LB_{with}(A, S)$	The wasted energy caused to S until now due to the link breakages between A and S.
$B_{lacklist}(S)$	The set of nodes that S believes are malicious and S does not want to work together with.
$B_{lacklist}(A, S)$	The subset of A's blacklist known by S until now.

6.2 Description of ARCS System

This section describes the proposed ARCS system for autonomous ad hoc networks. In the ARCS system, each node S keeps a set of records indicating the interactions with other nodes, as listed in Table 6.2. In a nutshell, when a node has a packet scheduled to be sent, it first checks whether this packet should be sent and which route should be used. Once an intermediate node on the selected route receives a packet forwarding request, it will check whether it should forward the packet. Once a node has successfully forwarded a packet on behalf of another node, it will request a receipt from its next node on the route and submit this receipt to the source of the packet to claim credit. After a packet delivery transaction finishes, all participating nodes will update their own records to reflect the changed relationships with other nodes and to detect possible malicious behavior. For each selfish node S, all the records listed in Table 6.2 will be initiated to be 0 when S first enters the network.

6.2.1 Cooperation Degree

In [28], Dawkins illustrates that reciprocal altruism is beneficial for every ecological system when favors are granted simultaneously, and gives an example to explain the survival chances of birds grooming parasites off each other's head which they cannot clean themselves. In that example, Dawkins divides the birds into three categories: *suckers*, which always help; *cheats*, which ask other birds groom parasites off their heads but never help others; and *grudgers*, which start out being helpful to every bird but refuse to help those birds that do not return the favor. The simulation studies have shown that both cheats and suckers extinct finally, and only grudgers win over time. Such selfish behavior and cooperation are also developed at length in [10, 11].

In order to best utilize their limited resources, selfish nodes in autonomous ad hoc networks should also act like the grudgers. In the ARCS system, each selfish node S keeps track of the *balance* $B(A, S)$ with any other node A known by S, which is defined as:

$$B(A, S) = (D_{ebit}(A, S) - W_{to}(A, S)) - (C_{redit}(A, S) - W_{by}(A, S)). \tag{6.5}$$

That is, $B(A, S)$ is the difference between what S has contributed to A and what A has contributed to S in S's point of view. If $B(A, S)$ is a positive value, it can be viewed as the relative damage that A has caused to S; otherwise, it is the relative help that S has received from A.

Besides keeping track of the balance, each node S will also set a threshold $B_{threshold}(A, S)$ for each known node A in the network, which we called *cooperation degree*. A necessary condition for S to help A, e.g., forwarding packet for A, is

$$B(A, S) < B_{threshold}(A, S). \tag{6.6}$$

Setting $B_{threshold}(A, S)$ to be ∞ means that S will always help A no matter what A has done, as the *suckers* act in the example. Setting $B_{threshold}(A, S)$ to be $-\infty$ means that S will never help A, as the *cheats* act in the example. In the ARCS system, each selfish node will set $B_{threshold}(A, S)$ to be a relatively small positive value, which means that initially S is helpful to A, and will keep being helpful to A unless the relative damage that A has caused to S exceeds $B_{threshold}(A, S)$, as the *grudgers* act in the example where they set the threshold to be 1 for any other bird. By specifying positive cooperation degrees, cooperation among selfish nodes can be enforced, while by letting the cooperation degrees to be relatively small, the possible damage caused by malicious nodes can be bounded.

6.2.2 Route Selection

In the ARCS system, *source routing* is used, that is, when sending a packet, the source lists in packet header the complete sequence of nodes through which the packet is to traverse. Due to insufficient balance, malicious behavior and possible node mobility, not all packet delivery transactions can succeed. When a node has a packet scheduled to be sent, it needs to decide whether it should start the packet delivery transaction and which route should be used.

In the ARCS system, each route is specified an expiring time indicating that after that time the route will become invalid, which is determined by the intermediate nodes during the route discovery procedure. Assume that S has a packet scheduled to be sent to D, route $R =$ "$R_0 R_1 \ldots R_M$" is a valid route known by S with $R_0 = S$, $R_M = D$, and M being the number of hops. Let $P_{drop}(R_i, S)$ denote the probability that node R_i will drop S's packet, and let $P_{delivery}(R, S)$ denote the probability that a packet can be successfully delivered from S to D through route

R at the current moment. S then calculates $P_{delivery}(R,S)$ as following:

$$P_{delivery}(R,S) = \begin{cases} 0 & (\exists R_i \in R)\ B(R_i,S) < -B_{max}(S,R_i) \\ 0 & (\exists R_i, R_j \in R)\ R_i \in B_{lacklist}(R_j,S) \\ \prod_{i=1}^{M-1}(1 - P_{drop}(R_i,S)) & otherwise \end{cases} \tag{6.7}$$

That is, a packet delivery transaction has no chance to succeed unless S has enough balance to request help from all intermediate nodes on the route and no node has been marked as malicious by any other node on the route. Once a valid route R with non-zero $P_{delivery}(R,S)$ is used to send a packet by S, the expected energy consumption can be calculated as:

$$E_{avg}(R,S) = EMP_{delivery}(R,S) + E_{fail}(R,S) \sum_{n=1}^{M-1} nE \left(\prod_{k=1}^{n-1} (1 - P_{drop}(R_k,S)) \right) P_{drop}(R_n,S), \tag{6.8}$$

and the expected profit of S is

$$P_{rofit}(R,S) = \alpha_S P_{delivery}(R,S) - \beta_S (1 - P_{delivery}(R,S)). \tag{6.9}$$

Let $Q(R,S)$ be the expected profit per unit energy when S uses R to send a packet to D at the current moment, referred to as the expected *energy efficiency*. that is,

$$Q(R,S) = \frac{P_{rofit}(R,S)}{E_{avg}(R,S)}. \tag{6.10}$$

Then in the ARCS system, which route should be selected is decided as follows:

Route Selection Decision: *Among all routes $\mathcal{R}$ known by S which can reach D, route R^* will be selected if and only if $P_{delivery}(R^*,S) > 0$ and $Q(R^*,S) \geq Q(R,S)$ for any other $R \in \mathcal{R}$.*

The above decision is optimal in the sense that no other known routes can provide better expected energy efficiency than route R^*. Since the accurate value of $P_{drop}(R_i,S)$ is usually not known, in the ARCS system, $P_{drop}(R_i,S)$ is estimated

as the ratio between the number of S's failed transactions caused by R_i and S's total transactions passing R_i.

After the route with the highest expected energy efficiency has been found by the sender S, suppose it is route R^*, in the next step S should decide whether it should use R^* to start a data packet delivery transaction. If the route quality is too low, simply dropping the packet without trying may be a better choice. Let $Q_{avg}(S)$ be S's average energy efficiency over the past:

$$Q_{avg}(S) = \frac{\alpha_S N_{S,succ} - \beta_S N_{S,fail}}{E_S}. \tag{6.11}$$

Then in the ARCS system, the following decision rule is used:

Packet Delivery Decision: *S will use route R^* to start a data packet delivery transaction if and only if the following condition holds:*

$$P_{rofit}(R^*, S) \geq Q_{avg}(S) E_{avg}(R^*, S) - \beta_S. \tag{6.12}$$

The left hand side of (6.12) is the expected profit when S uses R^* to start a packet delivery transaction, and the right hand side of (6.12) is the predicted profit by simply dropping the packet without trying, where β_S is the penalty due to dropping a packet and $Q_{avg}(S)E_{avg}(R^*, S)$ is the gain that S predicts to get with energy $E_{avg}(R^*, S)$ based on its past performance. If $Q_{avg}(S)$ is stationary over time, the above decision is optimal in the sense that the total profits can be maximized under the energy constraint.

6.2.3 Data Packet Delivery Protocol

In the ARCS system, a data packet delivery consists of two stages: *forwarding data packet stage* and *submitting receipts stage*. In the first stage, the data packet is delivered from its source to its destination, while in the second stage, each

Table 6.3: Notations used in the data packet delivery protocols

$sign_S(m)$	S generates a signature based on the message m.
$verify_S(m, s)$	Other nodes verify whether s is the signature generated by node S based on the message m.
$v \leftarrow m$	Assign the value of m to the variable v.
$MD()$	A message digest function, such as SHA-1 [1].
$seq_S(S, D)$	The sequence number of the current packet being processed with S being the source and D being the destination.

participating node on the route will submit a receipt to the source to claim credit. Table 6.3 lists some notations to be used.

Forwarding Data Packet Stage: Suppose that node S is to send a packet with payload m and sequence number $seq_S(S, D)$ to destination D through the route R. The sender S first computes a signature $s = sign_S(MD(m), R, seq_S(S, D))$. Next, S transmits the packet $(m, R, seq_S(S, D), s)$ to the next node on the route, increases $seq_S(S, D)$ by 1, and waits for receipts to be returned by the following nodes on route R. Once a selfish node A has received the packet $(m, R, seq_S(S, D), s)$, A first checks whether it is the destination of the packet. If it is, after necessary verifications, A returns a receipt to its previous node on the route to confirm the successful delivery; otherwise, A checks whether the packet should be forwarded. A is willing to forward the packet if and only if all the following conditions are satisfied: 1) A is on the route R; 2) $seq_S(S, D) > seq_A(S, D)$, where $seq_A(S, D)$ is the sequence number of the last packet that A has forwarded with S being source and D being the destination; 3) the signature is valid; 4) $B(S, A) < B_{threshold}(S, A)$; 5) no node on route R has been marked as malicious by node A.

Once A has successfully forwarded the packet $(m, R, seq_S(S, D), s)$ to the next node on the route, it will specify a time to wait for a receipt being returned by

Protocol 1 Forwarding data packet stage

▷ A is the current node, S is the sender, D is the destination. $(m, R, seq_S(S, D), s)$ is the received data packet from A's previous node if $A \neq S$; otherwise, $(m, R, seq_S(S, D), s)$ is the data packet generated by A.

if (A = S) **then**

S forwards $(m, R, seq_S(S, D), s)$ to next node, increases $seq_S(S, D)$ by 1, and waits for receipts to be returned.

else if $((A = D)$ and $(verify_S((m, R, seq_S(S, D)), s) =$ true) and $(seq_S(S, D) > seq_A(S, D)))$ **then**

A assigns the value of $seq_S(S, D)$ to $seq_A(S, D)$, and returns a receipt to its previous node.

else

if $((A \notin R)$ or $(verify_S((m, R, seq_S(S, D)), s) \neq$ true) or $(seq_S(S, D) \leq seq_A(S, D))$ or $(\exists R_i \in R, R_i \in B_{lacklist}(A)))$ **then**

A simply drops this packet.

else if $((B(S, A) > B_{threshold}(S, A))$ or (the link to A's next node is broken)) **then**

A drops the packet, and returns a receipt to its previous node which also includes the dropping reason.

else

A assigns the value of $seq_S(S, D)$ to $seq_A(S, D)$, forwards $(m, R, seq(S, D), s)$ to its next node, and waits for a receipt to be returned by the next node.

end if

end if

the next node before that time to confirm the successful transmission, which A will use to claim credit from S. In the ARCS system, a selfish node sets its waiting time to be the value of T_{link} multiplied by the number of hops following this node, where T_{link} is a relatively small interval to account for the necessary processing and waiting time (e.g., time needed for channel contention) per hop. Since in general the waiting time is small enough, we can assume that if a node can return a receipt to its previous node in time, the two nodes will still keep connected. The protocol execution of each participating selfish node in this stage is described in Protocol 1.

Submitting Receipts Stage: In autonomous ad hoc networks, nodes may not be willing to forward packets on behalf of other nodes. So after a node (e.g., A) has forwarded a packet $(m, R, seq_S(S, D), s)$ for another node (e.g., S), A will try to claim corresponding credit from S, which A can use later to request S to return the favor. To claim credit from S, A needs to submit necessary evidence to convince S that it has successfully forwarded packets for S. In the ARCS system, in order for A to show that it has successfully forwarded a packet for S, A only needs to submit a valid receipt generated by any node following A on the route (e.g., B) indicating that B has successfully received the packet. One possible format of such a receipt is

$$\{MD(m), R, seq_S(S, D), B, sign_B(MD(m), R, seq_S(S, D), B)\}.$$

That is, the receipt consists of the message $\{MD(m), R, seq_S(S, D), B\}$ and the signature generated by node B based on this message. For each selfish node, if it has dropped the packet or cannot get a receipt from its next node in time, or the received receipt is invalid, it will generate a receipt by itself and return it to its previous node; otherwise, it will simply send the received receipt back to its previous node on the route. The protocol execution of each participating selfish

node in this stage is described in Protocol 2.

Protocol 2 Submitting receipt stage

▷ A is the current node, $(MD(m), R, seq_S(S, D), B, s)$ is the successfully received packet to be processed.

if ($(A = D)$ or (no valid receipts have been returned by the next node after waiting enough time)) **then**

$s \leftarrow sign_A(MD(m), R, seq_S(S, D), A)$.

Send the receipt $\{MD(m), R, seq_S(S, D), A, s\}$ to A's previous node on R.

else

$receipt = \{MD(m), R, seq_S(S, D), B, s\}$, which is the returned receipt from the next node on the route.

if $(verify_B((MD(m), R, seq_S(S, D), B), s) = \text{true})$ **then**

Send *receipt* to A's previous node on R.

else

$s \leftarrow sign_A(MD(m), R, seq_S(S, D), A)$.

Send the receipt $\{MD(m), R, seq_S(S, D), A, s\}$ to A's previous node on R.

end if

end if

6.2.4 Update Records

In the ARCS system, after a packet delivery transaction has finished, no matter whether it is successful or not, each participating node will update its records to keep track of the changing relationships with other nodes and to detect possible malicious behavior. Next we use Fig. 6.1 to illustrate the records updating procedure, where S is this initiator of the transaction, D is the destination, and $R =$ "$S \ldots AMB \ldots D$" is the associated route.

For sender S, according to different situations, it updates its records as follows:

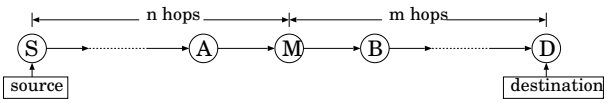


Figure 6.1: Update records

- *Case 1*: S has received a valid receipt signed by D which means that this transaction has succeeded. Then for each intermediate node X, S updates $C_{redit}(X, S)$ as follows:

$$C_{redit}(X, S) = C_{redit}(X, S) + E. \tag{6.13}$$

- *Case 2*: S has successfully sent a packet to its next node, but cannot receive any receipt in time. In this case, let X be S's next node, S then updates its records as follows:

$$W_{by}(X, S) = W_{by}(X, S) + E, \tag{6.14}$$

$$B_{lacklist}(S) = B_{lacklist}(S) \bigcup \{X\}. \tag{6.15}$$

  That is, refusing to return a receipt will be regarded as malicious behavior.

- *Case 3*: If S has received a valid receipt which is not signed by D, but signed by an intermediate node (e.g., M), which means that either M has dropped the packet or a returned receipt has been dropped by a certain node following M (including M) on the route in the submitting receipt stage. In this case, for each intermediate node X between S and M, S still updates $C_{redit}(X, S)$ using (6.13). Since node M's transmission cannot be verified by S, S has enough evidence to suspect that the packet is dropped by M. To reflect this suspect, S updates $W_{by}(M, S)$ as follows:

$$W_{by}(M, S) = W_{by}(M, S) + nE. \tag{6.16}$$

where nE accounts for the amount of energy that has been wasted in this transaction with n being the number of hops between S and M.

If a transaction fails, S also keeps a record of $(MD(m), R, seq_S(S, D), s)$ for this transaction as well as a copy of the returned receipt if there exists.

For each intermediate node (e.g., node M in Fig. 6.1) that has participated in the transaction, if it is selfish, it updates its records as follows:

- *Case 1*: M has successfully sent the packet to node B, and has got a receipt from B to confirm the transmission. In this case, M only needs to update $D_{ebit}(S, M)$ as follow:

$$D_{ebit}(S, M) = D_{ebit}(S, M) + E. \tag{6.17}$$

- *Case 2*: M has successfully sent the packet to node B, but cannot get a valid receipt from B. In this case, M updates its records as follows:

$$W_{to}(S, M) = W_{to}(S, M) + nE, \tag{6.18}$$

$$W_{by}(B, M) = W_{by}(B, M) + (n + 1)E, \tag{6.19}$$

$$B_{lacklist}(M) = B_{lacklist}(M) \bigcup \{B\}. \tag{6.20}$$

- *Case 3*: M has dropped the packet due to link breakage between M and B. Although this packet dropping is not M's fault, since M cannot prove it to S, M will take the responsibility. However, since this link breakage may be caused by S who has selected a bad route, or caused by B who tries to emulate link breakage to attack M, M should also record this link breakage. In this case, M updates its records as follows:

$$W_{to}(S, M) = W_{to}(S, M) + nE, \tag{6.21}$$

$$LB_{with}(B, M) = LB_{with}(B, M) + nE, \tag{6.22}$$

$$LB_{with}(S, M) = LB_{with}(S, M) + nE. \tag{6.23}$$

In the ARCS system, each selfish node (e.g., M) will also set a threshold $LB_{threshold}(S, M)$ with any other node (e.g., S) to indicate the damage that M can tolerate which is caused due to the link breakages happened between M and S. In this case, if $LB_{with}(B, M)$ exceeds $LB_{threshold}(B, M)$, B will be put into M's blacklist. Similarly, if $LB_{with}(S, M)$ exceeds $LB_{threshold}(S, M)$, S will be put into M's blacklist.

- *Case 4*: M has dropped the packet due to the reason that the condition in (6.6) is not satisfied or some nodes on R are in M's blacklist. In this case M does not need to update its records.

After finishing updating its records, M will also keep a copy of the submitted receipt for possible future usage, such as resolving inconsistent records update problem, as will be described in Section 6.2.6. From the above update procedure we can see that a selfish node should always return a receipt to confirm a successful packet reception, since refusing to return receipt is regarded as malicious behavior and cannot provide any gain.

6.2.5 Secure Route Discovery

In the ARCS system, DSR [47] is used as the underlying routing protocol to perform route discovery, which is an on-demand source routing protocol. However, without security consideration, the routing protocol can easily become an attacking target. For example, malicious nodes can inject an overwhelming amount of route request packets. In the ARCS system, besides necessary identity authenti-

cation, the following security enhancements have also been incorporated into the route discovery protocol:

1. When node S initiates a route discovery, it also appends its blacklist in the request packet. After an intermediate node A has received the request packet, it will update its own record $B_{lacklist}(S, A)$ using the received blacklist.

2. When an intermediate node A receives a route request packet which originates from S and A is not this request's destination, A first checks the following conditions: 1) A has never seen this request before; 2) A is not in S's blacklist; 3) $B(S, A) < B_{threshold}(S, A)$; 4) no nodes that have been appended to the request packet are in A's blacklist; 5) A has not forwarded any request for S in the last $T_{interval}(S, A)$ interval, where $T_{interval}(S, A)$ is the minimum interval specified by A to indicate that A will forward at most one route request for S in each $T_{interval}(S, A)$ interval. A will broadcast the request if and only if all of the above conditions can be satisfied, otherwise, A will discard the request.

3. During a discovered route is being returned to the requester S, each intermediate node A on the route appends the following information to the returned route: the subset of its blacklist that is not known by S, the value of $B_{threshold}(S, A)$ if not known by S, the value of $D_{ebit}(S, A)$, and node A's expected staying time at the current position. After S has received the route, for each node A on the discovered route, it updates the corresponding blacklist $B_{lacklist}(A, S)$, updates the value of $B_{threshold}(S, A)$, determines the expiring time of this route which can be approximated as the expected minimum staying time among all nodes on the route, and checks the consistency between $D_{ebit}(S, A)$ and $C_{redit}(A, S)$.

6.2.6 Resolve Inconsistent Records Update

In some situations, after a node (e.g., A) has successfully forwarded a packet for another node (e.g., S) and has sent a receipt back to S, the value of $C_{redit}(A, S)$ may not be increased immediately by S due to some intermediate node dropping the receipt returned by A. In this case, the value of $D_{ebit}(S, A)$ will be larger than the value of $C_{redit}(A, S)$, which we referred to as *inconsistent records update*. As a consequence, S may refuse to forward packets for A even the actual value of $B(A, S)$ is still less than $B_{threshold}(A, S)$, or S may continue requesting A to forward packets for it when the true value of $B(S, A)$ has exceeded $B_{threshold}(S, A)$. Next we describe how the inconsistent records update problem is resolved.

In the route discovery stage, after route R has been returned to S, S will check whether there exists inconsistency. If S finds that a node A on route R has reported a larger value of $D_{ebit}(S, A)$ than the value of $C_{redit}(A, S)$, when calculating route quality, S should use the value of $D_{ebit}(S, A)$ to temporarily substitute the value of $C_{redit}(A, S)$. In the packet delivery stage, when route R is picked by S to send packets, for each intermediate node A on route R, the value of $C_{redit}(A, S)$ will also be appended to the payload of the data packet.

When A receives an appended value of $C_{redit}(A, S)$ from S, and finds $C_{redit}(A, S) < D_{ebit}(S, A)$, A will submit those receipts that target on S but have not been confirmed by S to claim corresponding credits, where we say a receipt received by A at time t_1 and targeting on S has been *confirmed* if there existed at least one moment $t_2 > t_1$ before now at which A and S have agreed that $C_{redit}(A, S) = D_{ebit}(S, A)$. Once S has received an unconfirmed receipt returned by A, S will check whether there is a failed transaction record associated to this receipt. If no such record exists, either the receipt is faked, or the corresponding credit has been issued to A. If

there exists such a record, let B be the node who has signed the receipt associated to this transaction record, that is, all nodes between S and B have been credited by S. Let C be the node who has signed the receipt submitted by A. If B is in front of C on the route, S should use the new receipt singed by C to replace the previous receipt signed by B, and for each intermediate node X between B and C on the route, S should update $C_{redit}(X, S)$ using (6.13), also, if C is not the destination of the associated packet, S should update $W_{by}(C, S)$ using (6.16).

6.2.7 Parameter Selection

In the ARCS system, for each selfish node S, it needs to specify three types of thresholds regarding to any other node A in the network: the cooperation degree $B_{threshold}(A, S)$, the maximum tolerable damage due to link breakage $LB_{threshold}(A, S)$ and the minimum route request forwarding interval $T_{interval}(A, S)$, which are determined in the following way.

For each known node A, S initially sets $T_{interval}(A, S)$ to be a moderate value, such as a value equal to its own average pause time. During staying in the network, S will keep estimating a good route discovery frequency for itself, and will set $T_{interval}(A, S)$ to be the inverse of its own route discovery frequency. Similarly, S initially sets all link breakage thresholds using a (relatively small) constant value LB_{init}, and keeps estimating its own average link breakage ratio over time, assuming $P_{S,LB}$. For each node A, let $N_{trans}(A, S)$ be the total number of transactions that simultaneously revolve S and A with A either being S's next node or being the initiator of the transactions, then S may set

$$LB_{threshold}(A, S) = L_{hop} P_{S,LB} N_{trans}(A, S) E + LB_{init}, \tag{6.24}$$

where L_{hop} is the average number of hops per route.

For $B_{threshold}(A, S)$, if favors can be granted simultaneously, a small value (for example 1, as grudgers do in the ecological example) can work perfectly. However, in many situations favors cannot be granted immediately. For example, after S has helped A several times, S may not get similar amount of help from A due to that S does not need help from A currently or A has moved. Many factors can affect the selection of $B_{threshold}(A, S)$, among them some are unknown to S, such as other nodes' traffic patterns and behaviors, and some are unpredictable, such as mobility, which make selecting an optimal value for $B_{threshold}(A, S)$ hard or impossible. However, our simulation studies in Section 6.4 have shown that in most situations a relatively small constant value can achieve good tradeoff between energy efficiency and robustness to attacks.

6.3 Analysis of the ARCS System Under Attacks

In this section we analyze the performance of the ARCS system under the following types of attacks: dropping packet, emulating link breakage, injecting traffic, collusion and slander. Since the attacks of preventing good routes from being discovered are mainly used to increase attackers' chance of being on the discovered routes, they can be regarded as part of dropping packets or emulating link breakage attacks, and will not be analyzed separately. Similarly, modifying or delaying packets attacks can also be regarded as specific types of dropping packets attacks, and will not be analyzed separately. The results show that the damage that can be caused by malicious nodes is bounded, and the system is collusion-resistant.

Dropping Packet Attacks: In the ARCS system, malicious nodes can waste other nodes' energy by dropping their packets, which can happen either in the forwarding data packet stage or in the submitting receipts stage. We use Fig. 6.2

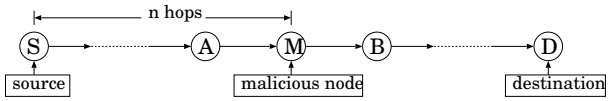


Figure 6.2: Dropping Packets Attacks

as an example to study the possible dropping packet attacks that can be launched by malicious node M. Based on in which stage M drops packets and whether M will return receipts, there are four possible attacking scenarios:

- Scenario 1: M drops a packet in the forwarding data packet stage, but creates a receipt to send back to A to confirm successful receiving from A. In this scenario, after S gets the receipt, S will increase $W_{by}(M, S)$ by nE, which equals to the total amount of energy that has been wasted by M. That is, in this scenario, the damage caused by M has been recorded by S and needs to be compensated by M later if M still wants to get help from S.

- Scenario 2: M drops a packet in the forwarding data packet stage, and refuses to return a receipt to A. In this scenario, although A will be mistakenly charged by S which increases $W_{by}(A, S)$ by $(n-1)E$, A will mark M as malicious and will stop working with M further. That is, M can never get help from A and cause damage to A in the future.

- Scenario 3: M drops the receipt returned by B, but creates a receipt to send back to A. In this scenario, M will be charged nE by S, but the nodes after M who have successfully forwarded the packet will not be credited by S immediately. That is, by taking some charge (here nE), M can cause inconsistent records update. However, as described in Section 6.2.6, this inconsistency can be easily resolved and will not cause further damage. That

is, M can only cause temporary records inconsistency with the extra payment of $(n+1)E$.

- Scenario 4: M drops the receipt returned by B, and refuses to return a receipt to A. This scenario is similar to scenario 3 with the only difference being that in this scenario A will be mistakenly charged by S, but M will be marked as malicious by A and cannot do any further damage to A in the future.

From the above analysis we can see that when a malicious node M launches dropping packet attacks, either it will be marked as malicious by some nodes, or the damage caused by it will be recorded by other nodes. Since for each node A, the maximum possible damage that can be caused by M is bounded by $B_{threshold}(M, A)$, the total damage that M can cause is also bounded.

Emulating Link Breakage Attacks: Malicious nodes can also launch emulating link breakage attacks to waste other nodes' energy. For example, in Fig. 6.2, when node A has received a request from S to forward a packet to M, M can just keep silent to let A believe that the link between A and M is broken. By emulating link breakage, M can cause a transaction to fail and waste other nodes' energy.

In the ARCS system, each selfish node handles the possible emulating link breakage attacks as follows: For each known node M, S keeps a record $LB_{with}(M, S)$ to remember the damage that has been caused due to link breakage between M and S, and if $LB_{with}(M, S)$ exceeds the threshold $LB_{threshold}(M, S)$, S will mark M as malicious and will never work with M again. That is, the damage that can be caused to S by malicious node M who launched emulating link breakage attacks is bounded by $LB_{threshold}(M, S)$.

Injecting Traffic Attacks: Besides dropping packets, attackers can also inject an excessive amount of traffic to overload the network and to consume other nodes'

valuable energy. Two types of packets can be injected: general data packets and route request packets. In the ARCS system, according to the route discovery protocol, the number of route request packets that can be injected by each node is bounded by 1 in each time interval $T_{interval}$. For general data packets, since an intermediate node A will stop forwarding packets for node M if $B(M, A) > B_{threshold}(M, A)$, the maximum damage that can be caused to node A by node M launching injecting general data traffic attacks is bounded by $B_{threshold}(M, A)$. In summary, by launching injecting traffic attacks, the maximum damage that can be caused by a malicious node M to node A is bounded.

Collusion Attacks: In order to increase their attacking capability, malicious nodes may choose to collude. Next we show that in the ARCS system colluding among malicious nodes cannot cause more damage to the network than working alone, that is, the ARCS system is collusion-resistant. First, it is easy to see that two nodes collude to launch injecting extra traffic attacks cannot increase the damage due to the existence of balance threshold (cooperation degree), and two nodes colluding to launch emulating link breakage attacks makes no sense, since each link breakage event has only two participants. Next we consider two malicious nodes colluding to launch dropping packets attacks.

Given a packet delivery transaction, we first consider the case that the two colluding nodes are neighbor of each other. For example, as in Fig. 6.2, assume that M and B collude. When M drops the packet, M can still get (or generate by itself, since M may know B's private key) the receipt showing that M has successfully forwarded the packet. However, this cannot increase their total attacking capability, since B needs to take the charge for the damage caused by this packet dropping. That is, in this case M is released from the charge by sacrificing B.

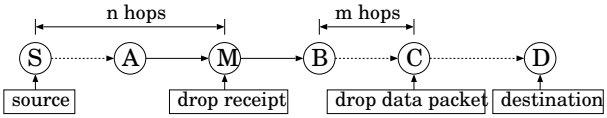


Figure 6.3: Collusion Attacks

If two colluding nodes are not neighbor of each other, the only way that they can collude is that one node drops the data packet in the forwarding data packet stage, and the other node drops the receipt in the submitting receipt stage, as shown in Fig. 6.3, where node C drops the data packet and node M drops the receipt. By colluding in this way, if C has returned a receipt to its previous node, C will not be charged by S temporarily, and all the nodes between M and C cannot get credits from S immediately. For node M, if M will return a receipt to A, S will increase $W_{by}(M, S)$ by nE, and if M refuses to return a receipt to A, M will be marked as malicious by A. That is, in this case, temporary inconsistent records update can be caused, but the colluding nodes will be overcharged by nE. However, according to Section 6.2.6, the inconsistency can be easily resolved.

Slander Attacks: In ARCS, each selfish node may propagate its blacklist to the network, which may give attackers chances to slander others. Next we show that instead of causing damage, such attacks can even benefit selfish nodes in some situations. Suppose that a malicious node M propagates information to the network to say that node X is malicious. For any selfish node S, this information will only be used in the situation where S wants to calculate a certain route's successful packet delivery probability according to (6.7) and X lies on this route. In this situation, the successful packet delivery probability of this route will always be calculated as 0 according to (6.7), and this route will not be used by S, which is just one goal of secure route discovery: preventing attackers from being on the

route. In all other situations, such information will not affect S's decision.

In summary, given that in the ARCS system there are L selfish nodes $\{S_1, \ldots, S_L\}$ and K malicious nodes $\{M_1, \ldots, M_K\}$. Let $B_{threshold}(M_k, S_l)$, $LB_{threshold}(M_k, S_l)$ and $T_{interval}(M_k, S_l)$ be the cooperation degree, the link breakage threshold, and the minimum route request forwarding interval that $S_l$set for M_k, respectively. Let T_{S_l} be node S_l's staying time in the system, and let $E_{request}$ (which is far less than E) be the consumed energy per route request forwarding. Based on the above analysis, we can see that the total damage D_{amage} that can be caused by all the malicious nodes is bounded by

$$D_{amage} \leq \sum_{k=1}^{K} \sum_{l=1}^{L} (B_{threshold}(M_k, S_l) + LB_{threshold}(M_k, S_l) + \frac{T_{S_l} * E_{request}}{T_{interval}(M_k, S_l)}) \tag{6.25}$$

That is, the damage that can be caused by malicious nodes is bounded, which is determined by the specified thresholds.

6.4 Simulation Studies

In our simulations 100 good nodes and various number of attackers are randomly deployed inside a rectangular area of 1000m $\times$ 1000m. Each node moves randomly according to the *random waypoint* model with $v_{max} = 10m/s$ and the average Pause time being 100 seconds. The physical layer assumes a fixed transmission range model, where two nodes can directly communicate with each other successfully only if they are in each other's transmission range. The MAC layer protocol simulates the IEEE 802.11 Distributed Coordination Function (DCF) with a four-way handshaking mechanism [44]. The maximum transmission Range for each node is fixed to be 250m.

In the simulations, each selfish node acts as a service provider which randomly picks another selfish node as the receiver and packets are scheduled to be generated according to a Poisson process. Similarly, each malicious node also randomly picks another malicious node as the receiver to send packets. The total number of malicious nodes varies from 0 to 50. Among those malicious nodes, 1/3 launch dropping packets attacks which drop all packets passing through them whose sources are not malicious, 1/3 launch emulating link breakage attacks which emulate link breakage once receiving packet forwarding request from selfish nodes, and 1/3 launch injecting traffic attacks. For each selfish or malicious node that does not launch injecting traffic attacks, the average packet inter-arrival time is 2 seconds, while for malicious nodes launching injecting traffic attacks, the average packet inter-arrival time is 0.1 second. In the simulations, all data packets have the same size.

Based on selfish nodes' forwarding decision, three types of systems have been implemented in the simulations: the proposed ARCS system, which we called "ARCS"; the ARCS system without balance constraint (i.e., cooperation degree is set to be infinity for all selfish nodes), which we called "ARCS-NBC"; and a fully-cooperative system, which we called "FULL-COOP". In "ARCS", all selfish nodes behave in the way as described in Section 6.2. In "ARCS-NBC", the same strategies as in "ARCS" have been used to detect launching dropping packets attacks and emulating link breakage attacks, but now (6.6) is not a necessary condition to forward packets for other nodes, and a selfish node will unconditionally forward packets for those nodes which have not been marked as malicious by it. In "FULL-COOP", all selfish nodes will unconditionally forward packets for other nodes, and no malicious nodes detection and punishment mechanisms have been used. In all three systems, the same route discovery procedure is used as described

in Section 6.2.5.

We use $\{S_1, \ldots, S_L\}$ to denote the L selfish nodes and use $\{M_1, \ldots, M_K\}$ to denote the K malicious nodes in the network. In this section the following performance metrics are used:

- *Energy efficiency of selfish nodes*, which is the total profits gained by all selfish nodes divided by the total energy spent by all selfish nodes until the current moment.

- *Average damage received per selfish node*: which is the total damage received by all selfish nodes until the current moment divided by the total number of selfish nodes, that is,

$$D_{avg} = \frac{1}{L} \sum_{i=1}^{L} \left(\sum_{l=1}^{L} B(S_l, S_i) + \sum_{k=1}^{K} B(M_k, S_i) \right). \tag{6.26}$$

- *Balance variation of selfish nodes*, which is the standard deviation of selfish nodes' overall balance with the assumption that $\sum_{l=1}^{L} B(S_l) = 0$, that is,

$$V_{ariation} = \sqrt{\frac{1}{L} \sum_{l=1}^{L} B(S_l) * B(S_l)}. \tag{6.27}$$

  By assuming $\sum_{l=1}^{L} B(S_l) = 0$, this definition has incorporated the effects caused by malicious nodes, which will make $\sum_{l=1}^{L} B(S_l)$ deviate from 0. This definition also reflects the fairness for selfish nodes, where $V_{ariation} = 0$ implies absolute fairness, and the increase of $V_{ariation}$ implies the increase of possible unfairness for selfish nodes.

In our simulations, each configuration has been run 10 independent rounds using different random seeds, and the result are averaged over all the rounds. In the simulations, we set $\alpha_S = 1$, $\beta_S = 0.5$, and $T_{interval}$ to be 100 seconds for each

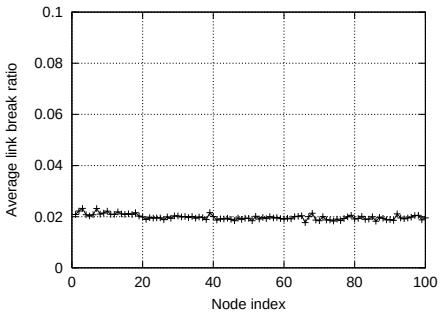
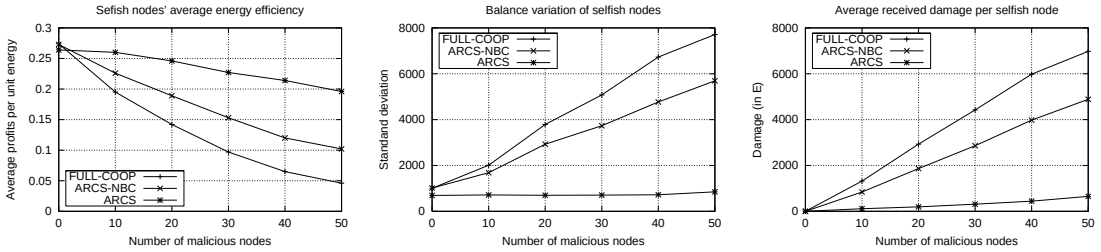


Figure 6.4: Estimated packet dropping ratio



(a) Energy efficiency (b) Balance variation (c) Average damage

Figure 6.5: Performance comparison among the three systems

selfish node S, which is equal to the average pause time. The running time for each round is 5000 seconds. For each selfish node, the link breakage ratio is estimated through its own experience, which is the ratio between the total number of link breakages it has experienced with itself being the transmitter and the total number of transmissions it has tried. Fig. 6.4 shows the estimated values of link breakage ratio by each node, which shows that all nodes have almost the same link breakage ratio (here 2%).

Fig. 6.5 shows the performance comparison among the three systems: ARCS, ARCS-NBC, and FULL-COOP, where in ARCS, $B_{threshold}$ is set to be $60E$, and the value of $LB_{threshold}$ is set according to (6.24) with $LB_{init} = 20E$. The experiments based on other values of $B_{threshold}$ have also been conducted which shows

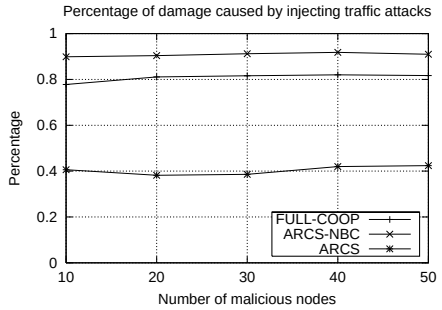


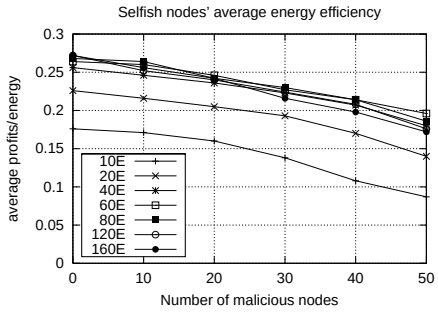
Figure 6.6: Effects of injecting traffic attacks in the three systems

that $60E$ can achieve good tradeoff between performance and possible damage (demonstrated in Fig. 6.7). From the selfish nodes' energy efficiency comparisons (Fig. 6.5(a)) we can see that ARCS has much higher efficiency than ARCS-NBC and FULL-COOP when there exist malicious nodes. When only selfish nodes exist, ARCS-NBC and FULL-COOP have the same efficiency, since they work in the same way, and both have slightly higher efficiency than ARCS with the payment of higher balance variation of selfish nodes, which is shown in Fig. 6.5(b). The balance variation comparison shows that ARCS has much lower balance variation than the other two systems, and almost keeps unchanged with the increase of the number of malicious nodes, while for the other two systems, the balance variation increases linearly and dramatically with the increase of the number of malicious nodes. This comparison also implies the lower unfairness for selfish nodes in the ARCS system. The average damage comparison (Fig. 6.5(c)) shows that in ARCS the damage that can be caused by malicious nodes is much lower than in other two systems, and increases very slowly with the increase of malicious nodes number.

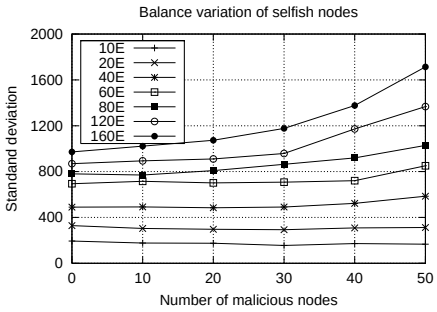
From the results shown in Fig. 6.5(a), Fig. 6.5(b), and Fig. 6.5(c) we can also see that although ARCS-NBC has gained a lot of improvement over FULL-COOP by introducing mechanisms to detect dropping packet and emulating link

breakage attacks, its performance is still much worse than ARCS. The reason is that ARCS-NBC cannot detect and punish those malicious nodes which launch injecting traffic attacks, so a large portion of energy has been wasted to forward packets for those nodes. Fig. 6.6 illustrates different effects of injecting traffic attacks in the three systems, where the vertical axis shows the percentage of damage caused by injecting traffic attacks to the network. From these results we can see that in ARCS, only about 40% percentage of damage is caused by injecting traffic attacks, in FULL-COOP this percentage increases to around 80%, while in ARCS-NBC the percentage increases to more than 90%, although the overall damage caused by all malicious nodes to the selfish nodes in ARCS-NBC is less than that in FULL-COOP. In another words, in Fig. 6.5(c), the gap between the results corresponding to "ARCS" and the results corresponding to "ARCS-NBC" is caused by injecting traffic attacks, while the gap between the results corresponding to "ARCS-NBC" and the results corresponding to "FULL-COOP" is caused by dropping packets/emulating link breakage attacks. These results explain why ARCS-NBC has much worse performance than ARCS, and clearly show that how necessary it is to introduce mechanisms to defend against such injecting traffic attacks.

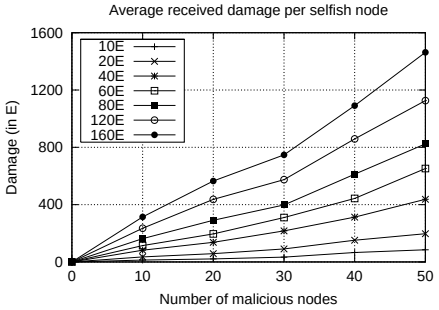
Next we evaluate the ARCS system under different cooperation degree configurations, where all other parameters keep unchanged. Fig. 6.7 shows the performance of the ARCS system by varying cooperation degree from 10E to 160E. From Fig. 6.7(a) we can see that when the cooperation degree is 40E or more, the energy efficiency becomes almost identical. However, Fig. 6.7(b) and Fig. 6.7(c) show that with the increase of cooperation degree, both the balance variation of selfish nodes and the average received damage per selfish node increase. This can be explained



(a) Energy efficiency comparison



(b) Balance variation comparison



(c) Average damage comparison

Figure 6.7: Performance comparison of the ARCS system under different cooperation degrees

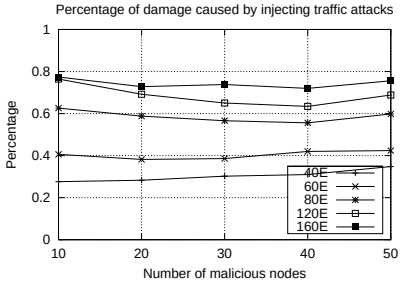


Figure 6.8: Effects of injecting traffic attacks under different cooperation degrees in the ARCS system

using Fig. 6.8, which shows that with the increase of cooperation degree, the percentage of damage that is caused by injecting traffic attacks also increases. That is, the higher the cooperation degree, the more vulnerable to injecting traffic attacks. These results suggest that a relative small cooperation degree (for example 40E) is enough to achieve good performance for selfish nodes, such as high energy efficiency, low unfairness and small damage.

6.5 Summary

In this chapter we have investigated the issues of cooperation stimulation and security in autonomous ad hoc networks, and proposed an Attack-Resistant Cooperation Stimulation (ARCS) system to stimulate cooperation among selfish nodes and defend against various attacks launched by malicious nodes. In the ARCS system, each node can adaptively adjust their own strategies according to the changing environments. The analysis has shown that in the ARCS system, the damage that can be caused by malicious nodes is bounded, and the cooperation among selfish nodes is enforced through introducing a positive cooperation degree. At the same time, the ARCS system maintains good fairness among selfish nodes. The simulation results have also agreed with the analysis. Another key property of the ARCS system is that it is fully distributive, completely self-organizing, and does not require any tamper-proof hardware or central management points.

Chapter 7

Game Theoretic Analysis of Secure Cooperation in Autonomous Ad Hoc Networks

In Chapter 6 we have proposed an attack-resistant resistant cooperation stimulation system for autonomous ad hoc networks. However, the proposed schemes are still heuristics. In this chapter we will formally address the security and cooperation in autonomous ad hoc networks under a game theoretic framework.

Although this work also falls into the category of reputation-based cooperation stimulation for autonomous ad hoc networks in a game theoretic framework, there are several major differences which distinguish our work from the existing work, such as [6, 26, 31, 59, 78, 80]. First, since ad hoc networks are usually deployed in hostile environments, in this work not only selfish behavior, but also malicious behavior has been considered. Malicious behavior has been overlooked in existing work, but can cause severe trouble without necessary countermeasures. Second, in this work the issues of cooperation and security have been studied under more

realistic scenarios, e.g., the communication medium is error-prone. The analysis shows that almost all the existing cooperation schemes will break down under such scenarios. Third, unlike some existing work which assumes that nodes will honestly report their private information (e.g., [78]), in this work the possible cheating behavior has been fully exploited, and cheat-proof strategies have been devised. Fourth, instead of only using Nash equilibrium, in this work other optimality criteria, such as cheat-proofing and fairness, have also been considered.

This chapter is organized as follows. Section 7.1 focuses on a simple yet illuminating two-player packet forwarding game and investigates the optimal cooperation strategies. Section 7.2 identifies the underlying reasons why stimulating cooperation under such scenarios is difficult and describes the secure routing and packet forwarding game. In Section 7.3, a set of reputation-based attack-resistant cooperation stimulation strategies are devised. The analysis of the proposed strategies are provided in Section 7.4 and Section 7.5. Extensive simulations have been conducted to evaluate the effectiveness of the proposed strategies under various scenarios, and the results are demonstrated in Section 7.6. Finally, Section 7.7 summarizes this chapter.

7.1 Two-node Packet Forwarding Game

We first study a simple yet illuminating two-node multi-stage packet forwarding game, which is modeled as follows. There are two players (nodes) in this game, denoted by $N = \{1, 2\}$. Each player needs its opponent to forward a certain number of packets in each stage. To simplify the illustration, we assume that all packets have the same size. For each player i, the cost to forward a packet is c_i, and the gain it can get for any packet that its opponent has forwarded for it is g_i. Here

the cost can be the consumed energy and the gain is usually application-specific. Let B_i be the number of packets that player i will request its opponent to forward at each stage. The values of B_i, c_i, and g_i will be reported by both players to each other, either honestly or dishonestly, before the game is started. It is also reasonable to assume that $g_i \geq c_i$, and there exists a c_{max} with $c_i \leq c_{max}$.

Let $A_i = \{0, 1, \ldots, B_{3-i}\}$ denote the set of actions that player i can take in each stage, where $a_i \in A_i$ denotes that player i will forward a_i packets for its opponent in this stage. We refer to an action profile $a = (a_1, a_2)$ as an *outcome* and denote the set $A_1 \times A_2$ of outcomes by A. Then in each stage players' payoffs are calculated as follows provided the action profile a being taken:

$$u_1(a) = a_2 \times g_1 - a_1 \times c_1, \quad u_2(a) = a_1 \times g_2 - a_2 \times c_2. \tag{7.1}$$

That is, the payoff of a player is the difference between the total gain it obtained with the help of its opponent and the total cost it spent to help its opponent. We refer to $u(a) = (u_1(a), u_2(a))$ as the payoff profile associated with the action profile a. According to the backward induction principle [64], if this game will only be played for fixed finite times, the only Nash equilibrium (NE) is $a^* = (0, 0)$, no matter whether the two players move simultaneously or not. That is, if the game will only be played for one time, no node will help its opponent. The same result also holds for the case when the stage game will be played for only finite times and the game termination time is known by both players.

Next we show that cooperation can still be achieved if the game will be played for infinite times, or for finite times but no player knows the exact game termination time. Let G denote the repeated version of the above one-stage packet forwarding game. Let s_i denote player i's behavior strategy, and let $s = (s_1, s_2)$ denote the

strategy profile. Next we consider the following two utility functions:

$$U_i(s) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} u_i(s), \tag{7.2}$$

$$U_i(s,\delta) = (1-\delta) \sum_{t=0}^{\infty} \delta^t u_i(s) \tag{7.3}$$

Utility function (7.2) can be used when the game will be played for infinite times. The discounted version (7.3) can be used when the game will be played for finite times, but no one knows the exact termination time. Here the discount factor δ (with $0 < \delta < 1$) characterizes each player's expected playing time. Since in general the results obtained based on (7.2) can also be applied to the scenarios when (7.3) is used as long as δ approaches to 1, in this section we will mainly focus on (7.2).

Now we analyze the possible NE for the game G with utility function (7.2). According to the Folk theorem [64], for every feasible and enforceable payoff profile, there exists at least one NE to achieve it, where the set of feasible payoff profiles for the above game is

$$V_0 = \text{ convex hull}\{v \mid \exists a \in A \text{ with } u(a) = v\}. \tag{7.4}$$

and the set of enforceable payoff profiles, denoted by V_1, is

$$V_1 = \{v \mid v \in V_0 \text{ and } \forall i : v_i \geq \underline{v}_i, \text{ where } \underline{v}_i = \min_{a_{-i}\in A_{-i}} \max_{a_i \in A_i} u_i(a_{-i}, a_i)\}. \tag{7.5}$$

Figure 7.1 depicts these sets for the game with $B_1 = 1$ and $B_2 = 2$, where the vertical axis denotes player 1's payoff and the horizontal axis denotes player 2's payoff. The payoff profiles inside the convex hull of $\{(0,0), (g_1, -c_2), (g_1 - 2c_1, 2g_2 - c_2), (-2c_1, 2g_2)\}$ (including the boundaries) are the set of feasible payoff profiles V_0, the set of payoff profiles inside the shading area (including the boundaries) are the set of feasible and enforceable payoff profiles V_1. We can easily check

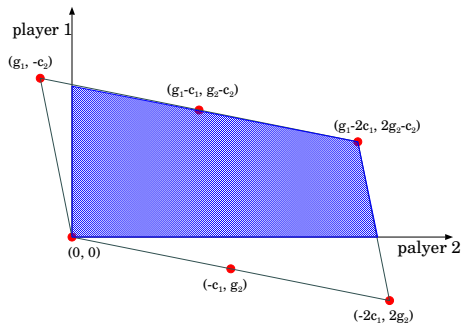


Figure 7.1: Feasible and enforceable payoff profiles

that as long as $g_1 g_2 > c_1 c_2$, there exist an infinite number of NE. To simplify our illustration, in this chapter we will use $x = (x_1, x_2)$ to denote the set of NE strategies corresponding to the enforceable payoff profile $(x_2 g_1 - x_1 c_1, x_1 g_2 - x_2 c_2)$.

7.1.1 Equilibrium Refinement

Based on the above analysis we can see that the infinitely repeated game G may have an infinite number of NE. However, not all the obtained NE payoff profiles are simultaneously acceptable to both players. For example, the payoff profile $(0, 0)$ will not be acceptable from both players' point of view if they are rational. Further, the existence of multiple NE payoff also requires nodes to make an agreement on which NE strategy should be used, which also introduces extra trouble. Next we show how to perform *equilibrium refinement*, that is, how to introduce new optimality criteria to eliminate those NE solutions which are less rational, less robust, or less likely.

When performing equilibrium refinement, the following optimality criteria will be considered: *Pareto optimality*, *subgame perfection*, *proportional fairness*, and *absolute fairness*. In the literature, Pareto optimality and fairness have been used to refine the equilibrium in [78], and subgame perfection has been considered to

remove empty threats in [80].

Subgame Perfection: Our first step towards refining the NE solutions is to rule out those empty threats. This motivates the equilibrium refinement based on more credible punishments known as *subgame perfect equilibrium*, which eliminates those equilibria in which the players' threats are empty. According to the perfect Folk theorem [64], every strictly enforceable payoff profile $v \in V_2$ is a subgame perfect equilibrium payoff profile of the game G, where

$$V_2 = \{v \mid v \in V_0 \text{ and } \forall i : v_i > \underline{v}_i, \text{ where } \underline{v}_i = \min_{a_{-i} \in A_{-i}} \max_{a_i \in A_i} u_i(a_{-i}, a_i)\}. \quad (7.6)$$

That is, after applying the criterion of subgame perfection, only a small set of NE are removed.

Pareto Optimality: Our second step towards refining the set of NE solutions is to apply the criterion of *Pareto optimality*[1]. It is easy to check that only those payoff profiles lying on the boundary of the set V_0 could be Pareto optimal. Let V_3 denote the subset of feasible payoff profiles which are also Pareto optimal. For the case depicted in Figure 7.1, V_3 is the set of payoff profiles which lie on the segment between $(g_1, -c_2)$ and $(g_1 - 2c_1, 2g_2 - c_2)$ and on the segment between $(g_1 - 2c_1, 2g_2 - c_2)$ and $(-2c_1, 2g_2)$. After applying the criterion of Pareto optimality, although a large portion of NE have been removed from the feasible set, there still exist an infinite number of NE. Let $V_4 = V_3 \cap V_2$.

Proportional Fairness: Next we try to further refine the solution set based on the criterion of proportional fairness. Here a payoff profile is proportionally fair if $U_1(s)U_2(s)$ can be maximized, which can be achieved by maximizing $u_1(s)u_2(s)$

[1]Given a payoff profile $v \in V_0$, v is said to be Pareto optimal if there is no $v' \in V_0$ for which $v'_i > v_i$ for all $i \in N$; v is said to be strongly Pareto optimal if there is no $v' \in V_0$ for which $v'_i \geq v_i$ for all $i \in N$ and $v'_i > v_i$ for some $i \in N$ [64].

in each stage. Then we can reduce the solution set to a unique point as follows:

$$x^* = \begin{cases} (\frac{\frac{c_2}{g_2}+\frac{g_1}{c_1}}{2}B_1, B_1) & if \quad \frac{B_1}{B_2} < \frac{2}{\frac{c_2}{g_2}+\frac{g_1}{c_1}} \\ (B_2, B_1) & if \quad \frac{2}{\frac{c_2}{g_2}+\frac{g_1}{c_1}} \leq \frac{B_1}{B_2} \leq \frac{\frac{c_1}{g_1}+\frac{g_2}{c_2}}{2} \\ (B_2, \frac{\frac{c_1}{g_1}+\frac{g_2}{c_2}}{2}B_2) & if \quad \frac{B_1}{B_2} > \frac{\frac{c_1}{g_1}+\frac{g_2}{c_2}}{2} \end{cases} \tag{7.7}$$

Absolute Fairness: In many situations, absolute fairness is also an important criterion. We first consider *absolute fairness in payoff*, which refers to that the payoff of these two players should be equal. By combining the criterion of Pareto optimality, the optimal strategy profile should be

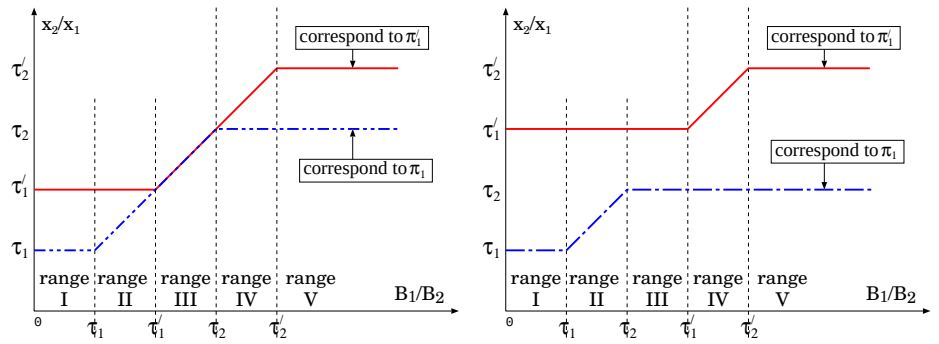
$$x^* = \begin{cases} (\frac{g_1+c_2}{g_2+c_1}B_1, B_1) & if \quad \frac{B_1}{B_2} \leq \frac{g_2+c_1}{g_1+c_2}, \\ (B_2, \frac{g_2+c_1}{g_1+c_2}B_2) & if \quad \frac{B_1}{B_2} \geq \frac{g_2+c_1}{g_1+c_2}. \end{cases} \tag{7.8}$$

Another similar criterion is *absolute fairness in cost*, which refers to that the cost spent by these two players for each other should be equal. By combining the criterion of Pareto optimality, the optimal strategy profile should be

$$x^* = \begin{cases} (\frac{c_2}{c_1}B_1, B_1) & if \quad \frac{B_1}{B_2} \leq \frac{c_1}{c_2}, \\ (B_2, \frac{c_1}{c_2}B_2) & if \quad \frac{B_1}{B_2} \geq \frac{c_1}{c_2}, \end{cases} \tag{7.9}$$

7.1.2 Cheat-proof Nash Equilibrium Strategies

It is worth noting that the above unique solutions (7.7), (7.8) and (7.9) require players to reveal their private information to their opponents. While due to players' selfishness, it's unrealistic to expect them to honestly reveal their private information. Further, to maximize their own payoffs, selfish players may tend to cheat whenever they believe cheating can increase their payoffs. In this chapter, we refer to a NE as *cheat-proof* if no player can further increase its payoff by revealing false private information to its opponents.



(a) $\tau_1' < \tau_2$ (b) $\tau_1' > \tau_2$

Figure 7.2: Player 1 falsely reports the value of π_1

Now we study whether the obtained unique solutions are cheat-proof. We first study the solution (7.7). Let $\pi_i = c_i/g_i$ denote player i's cost-gain (CG) ratio. We first analyze whether player i can increase its payoff by reporting a false CG ratio given that player 2 will honestly report its CG value. That is, we fix the value of π_2, let π_1 be player 1's true value, and let π_1' be the value that player 1 will falsely report with $\pi_1' > \pi_1$. Let $\tau_1 = \frac{2}{\pi_2 + \frac{1}{\pi_1}}$, $\tau_2 = \frac{\pi_1 + \frac{1}{\pi_2}}{2}$, $\tau_1' = \frac{2}{\pi_2 + \frac{1}{\pi_1'}}$, $\tau_2' = \frac{\pi_1' + \frac{1}{\pi_2}}{2}$. It is easy to check that $\tau_1 < \tau_1'$ and $\tau_2 < \tau_2'$. Recall that B_i is the maximum number of packets that player i will request its opponent to forward for it in each stage. Let (x_1, x_2) denote the number of packets in average they will forward for each other in each stage according to the solution (7.7) given that the true values of B_1 and B_2 are known by both players. The relationship between x_2/x_1 and B_1/B_2 under different situations is illustrated in Fig. 7.2(a) and Fig. 7.2(b). In these two figures, the dashed curve corresponds to the relationship between x_2/x_1 and B_1/B_2 given that player 1 honestly reports its CG value, which is π_1; while the solid curve corresponds to the relationship between x_2/x_1 and B_1/B_2 given that player 1 falsely report its CG value, which is π_1'.

From the results illustrated in Fig. 7.2 we can see that by falsely reporting a

higher CG ratio, in most situations player 1 can increase the ratio of x_2/x_1. Next we study the effect of falsely reporting a high CG ratio on player 1's payoff. We first consider the situation that $\tau_1' \leq \tau_2$, which is illustrated in Fig. 7.2(a). In this case the whole feasible space can be partitioned into 5 subareas along the feasible range of B_1/B_2:

- For any value of B_1/B_2 inside range I, the solution corresponding to π_1 is $(\frac{\pi_2+\frac{1}{\pi_1}}{2}B_1, B_1)$, and the solution corresponding to π_1' is $(\frac{\pi_2+\frac{1}{\pi_1'}}{2}B_1, B_1)$. Since $\pi_1' > \pi_1$, by falsely reporting a higher CG ratio player 1 can forward less packets for player 2 than it should, consequently increasing its own payoff.

- For any value of B_1/B_2 inside range II, the solution corresponding to π_1 is (B_2, B_1), and the solution corresponding to π_1' is $(\frac{\pi_2+\frac{1}{\pi_1'}}{2}B_1, B_1)$. Since $B_2 > \frac{\pi_2+\frac{1}{\pi_1'}}{2}B_1$, by falsely reporting a higher CG ratio player 1 can forward less packets for player 2 than it should, consequently increasing its own payoff.

- For any value of B_1/B_2 inside range III, the solution corresponding to π_1 is (B_2, B_1), and the solution corresponding to π_1' is also (B_2, B_1). That is, in this situation by changing the value of π_1 to π_1', player 1's payoff will not change.

- For any value of B_1/B_2 inside range IV, the solution corresponding to π_1 is $(B_2, \frac{\pi_1+\frac{1}{\pi_2}}{2}B_2)$, and the solution corresponding to π_1' is (B_2, B_1). Since $B_1 > \frac{\pi_1+\frac{1}{\pi_2}}{2}B_2$, by falsely reporting a higher CG ratio player 1 can request player 2 to forward more packets for it than player 2 should, consequently increasing its own payoff.

- For any value of B_1/B_2 inside range V, the solution corresponding to π_1 is $(B_2, \frac{\pi_1+\frac{1}{\pi_2}}{2}B_2)$, and the solution corresponding to π_1' is $(B_2, \frac{\pi_1'+\frac{1}{\pi_2}}{2}B_2)$. Since

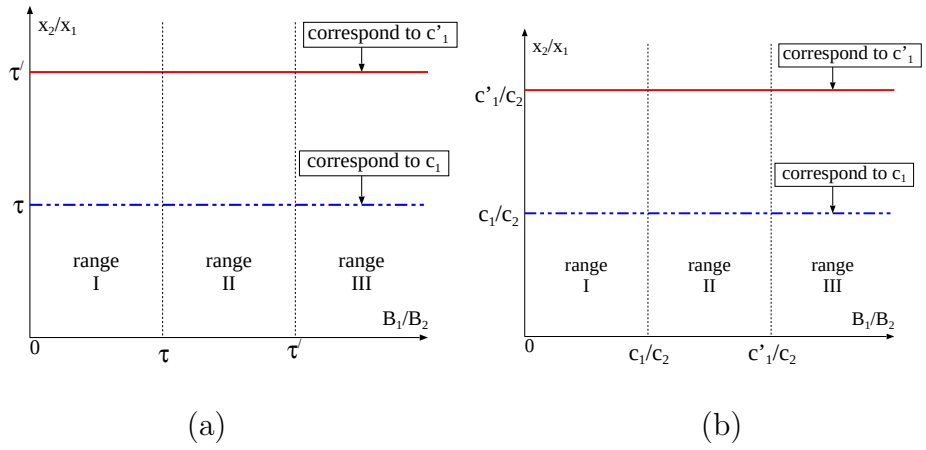


Figure 7.3: Player 1 falsely reports its cost

$\frac{\pi_1'+\frac{1}{\pi_2}}{2}B_2 > \frac{\pi_1+\frac{1}{\pi_2}}{2}B_2$, by falsely reporting a higher CG ratio player 1 can request player 2 to forward more packets for it than player 2 should, and consequently increases its own payoff.

Similar results can also be obtained for the case that $\tau_1' > \tau_2$, where now player 1 can increase its own payoff over all possible values of B_1/B_2 by falsely reporting a higher π_1 value given that π_2 is fixed. In summary, by falsely reporting a higher π_1 value, in most situations player 1 can increase its payoff, and in no situations player 1's payoff will be decreased. Further, the higher player 1 reports the value of CG ratio, the more benefit player 1 can get. Similarly, player 2 can also increase its benefit by falsely reporting a higher CG ratio.

Next we consider the solution (7.8). Now let $\tau = \frac{g_2+c_1}{g_1+c_2}$, and $\tau' = \frac{g_2+c_1'}{g_1+c_2}$, where $c_1 < c_1'$. Fig. 7.3(a) illustrates the relationship between x_2/x_1 and B_1/B_2 for the two different reported cost values c_1 and c_1', where g_1, g_2 and c_2 are fixed. Similar as in Fig. 7.2, the dashed curve corresponds to the case that player 1 reports a true cost value, while the solid curve corresponds to the case that player 1 reports a false cost value. From Fig. 7.3(a) we can see that by falsely reporting a higher

cost value, in all situations player 1 can increase the ratio of x_2/x_1. Next we study the effect of falsely reporting a higher cost on player 1's payoff. As shown in Fig. 7.3(a), the whole space can be partitioned into 3 subareas along the feasible range of B_1/B_2:

- For any value of B_1/B_2 inside range I, the solution corresponding to c_1 is $(B_1/\tau, B_1)$, and the solution corresponding to c_1' is $(B_1/\tau', B_1)$. Since $\tau' > \tau$, by falsely reporting a higher cost player 1 can forward less packets for player 2 than it should, then increase its own payoff.

- For any value of B_1/B_2 inside range II, the solution corresponding to c_1 is $(B_2, \tau B_2)$, and the solution corresponding to c_1' is $(B_1/\tau', B_1)$. Since $B_1/\tau' < B_2$ and $B_1 > \tau B_2$, by falsely reporting a higher cost, player 1 can forward less packets for player 2 than it should and request player 2 to forward more packets for it than player 2 should, then increase its own payoff.

- For any value of B_1/B_2 inside range III, the solution corresponding to c_1 is $(B_2, \tau B_2)$, and the solution corresponding to c_1' is $(B_2, \tau' B_2)$. Since $\tau' > \tau$, by falsely reporting a higher cost, player 1 can always request player 2 to forward more packets for it than player 2 should do, and consequently increase its own payoff.

In summary, by falsely reporting a higher c_1 value, in all situations player 1 can increase its payoff given that c_2 and g_2 are fixed. Further, the higher player 1 reports the value of c_1, the more benefit player 1 can get. Applying similar analysis it is also easy to show that by falsely reporting a lower g_1 value, in all situations player 1 can increase its payoff given that c_2 and g_2 are fixed. Similarly, player 2 can also increase its benefit by falsely reporting a higher c_2 or a lower g_2 given that

g_1 and c_1 are fixed.

Now we consider the solution (7.9). Fig. 7.3(b) illustrates the relationship between x_2/x_1 and B_1/B_2 for the two different reported cost values c_1 and c_1' with $c_1' > c_1$ and c_2 fixed. From Fig. 7.3(b) we can see that by falsely reporting a higher c_1 value, in all situations player 1 can increase the ratio of x_2/x_1. Applying similar analysis as before, we can conclude that given c_2 fixed, by falsely reporting a higher c_1 value, player 1 can always increase its payoff. Further, the higher player 1 reports the value of c_1, the more benefit player 1 can get. Similarly, player 2 can also increase its payoff by falsely reporting a higher c_2 value given c_1 fixed.

Based on the above analysis, it is surprising to see that none of them is cheat-proof. Since all these unique solutions are strongly Pareto optimal, the increase of its opponent's payoff will lead to the decrease of its own payoff. Therefore players have no incentive to honestly report their private information. On the contrary, they will cheat whenever cheating can increase their payoff.

What is the consequence if both players will cheat? Let's first examine the solution (7.7). In this case, based on the above analysis, both players will report a c_i/g_i value as high as possible. Since we have assumed $g_i \geq c_i$ and $c_i \leq c_{max}$, both player will set $g_i = c_i = c_{max}$, and the solution (7.7) will become the following form:

$$x^* = (\min(B_1, B_2), \min(B_1, B_2)). \tag{7.10}$$

After applying similar analysis for the solutions (7.8) and (7.9), it is surprising to see, but easy to understand, that both will also converge to the form (7.10). Accordingly, the corresponding payoff profile is

$$v^* = ((g_1 - c_1)\min\{B_1, B_2\}, (g_2 - c_2)\min\{B_1, B_2\}). \tag{7.11}$$

Besides g_i and c_i, players can also report false B_i information. However, it is

easy to check that a rational player should always report a true B_i value. Next we use player 1 as an example to show this. If $B_1 \leq B_2$, reporting a higher B_1 can only increase the number of packets it should forward for player 2, which introduces no gain to it, while by reporting a lower B_1, although it can decrease the number of packets forwarded for player 2, the number of its own packets forwarded by player 2 will also be decreased and cannot introduce gain to it too due to $g_1 \geq c_1$. Similarly, if $B_1 > B_2$, reporting a higher B_1 will not affect the solution, while reporting a lower B_1 may also reduce the number of packets that player 2 will forward for it, which introduces no gain as long as $g_1 \geq c_1$. Therefore, player 1 should not report a false B_1 value. The same analysis also applies to player 2.

In summary, when cheating behavior is considered, all the above unique solutions converge to the same form as in (7.10) with payoff being (7.11), that is, *in the two-player packet forwarding game, in order to maximize its own payoff and be resistant to possible cheating behavior, a player should not forward more packets than its opponent does for it.* A simple NE strategy to achieve the payoff profile (7.11) is as follows:

Two-node cheat-proof packet forwarding strategy: *For each player $i \in N$, in each stage it should forward* $\min(B_1, B_2)$ *packets for its opponent unless there was a previous stage in which its opponent has forwarded less than* $\min(B_1, B_2)$ *packets for it, in which case it will stop forwarding packets for its opponent forever.*

7.1.3 Remarks

The strategies proposed in [78, 80] may look similar to the one described above. In [78] Srinivasan et al. studied the cooperation in ad hoc networks by focusing on

the energy-efficient aspects of cooperation, where in their solution the nodes are classified into different energy classes and the behavior of each node depends on the energy classes of the participants of each connection. They have demonstrated that if two nodes belong to the same class, they should apply the same packet forwarding ratio. However, they require nodes to *honestly* report their classes, and a node can easily cheat to increase its own performance, such as the approach mentioned in [31] (section VIII). Meanwhile, using normalized throughput alone as the performance metric may not be a good choice in general, as to be explained in the section 7.5.3.

In [80], Urpi et al. claimed that it is not possible to force a node to forward more packets than it sends on average (Lemma 6.2), and then concluded that cooperation can be enforced in a mobile ad hoc network, provided that enough members of the network agree on it, and if no node has to forward more traffic that it generates (Theorem 6.3). However, the above analysis has shown that a strategy profile can still be enforceable even this may require a node to forward more packets than it sends on average, as illustrated in solutions (7.7), (7.8) and (7.9). Second, in mobile ad hoc network, due to the multihop nature, in general the number of packets a node forwards should be much more than the number of packets it generates. Accordingly, their strategy cannot enforce cooperation at all in most scenarios.

One major contribution of our analysis lies in that we have exploited all the possible NE strategies, demonstrated why some strategies are not good, why they cannot be acceptable by the players, and why the solution (7.10) is the only one that should be adopted. In other words, we have provided more insight and physical meaning for the solution (7.10).

The works presented in [7,100] are also related to ours in the sense that cheating behavior has also been considered. They have proposed auction-based schemes to stimulate packet forwarding participation, where by using VCG-based second price auctioning these schemes force selfish nodes to honestly report their true private information (such as cost) to maximize their profit. However, in their schemes, a trusted third-party auctioneer is required per route selection and central banking services are needed to handle billing information, which usually cannot be satisfied in mobile ad hoc network. In our work, we focus on the scenario that neither trusted third-party auctioneer nor central banking service is available.

7.2 Secure Routing and Packet Forwarding Game

Now we investigate how to stimulate cooperation among selfish nodes in autonomous mobile ad hoc networks under realistic scenarios, by also taking into consideration possible malicious behavior. We consider an autonomous mobile ad hoc networks with a finite population of users, denoted by N. We do not assume the availability of any tamper-proof hardware or central banking service, therefore the scheme should be completely reputation-based. We focus on the situation that each user will stay in the network for a relatively long time, such as those students in a campus. But we do not require them to keep connected all time, and we allow users to leave and join the network if necessary. The goal is not to enforce all the users to act in a fully cooperative fashion at all time, which has been shown in [31, 100] to be not achievable in most situations. Instead, the goal is to stimulate cooperation among nodes as much as possible through playing reciprocal altruism, and at the same time take into consideration possible cheating and malicious behavior as well as fairness concern.

We assume that each user has a unique registered and verifiable identity (e.g., a public/private key pair), which is issued by some central authority and will be used to perform necessary access control and authentication. Each node may send information to the others or request information from the others. We focus on the information-push model, where it is the source's duty to guarantee the successful delivery of packets to their destinations. But the obtained results can be easily extended to the information-pull model. We assume that for each user $i \in N$, forwarding a packet will incur cost c_i, and letting a packet be successfully delivered to its destination can bring it gain g_i. Here the cost corresponds to the efforts spent by i, such as energy, and the gain is usually user-specific and/or application-specific.

Before devising cooperation stimulation strategies for autonomous mobile ad hoc networks, we first pose some challenges that we may meet. First, most previous work addresses cooperation enforcement under a repeated game model, such as [6, 31, 78, 80], which assume either random connection or fixed setup. However, the repeated model rarely holds in autonomous mobile ad hoc networks due to the topology change and variable request rate. In such networks, a source may request different nodes to forward packets at differen time and may act as a relay for different sources. Meanwhile, the request rates of each node to other nodes are usually variable, which can be caused either by its inherent variable traffic generation rate, or by mobility. A direct consequence of non-repeated model is that favors cannot be simultaneous granted. In [28], Dawkins demonstrated that reciprocal altruism is beneficial for every ecological system when favors are granted simultaneously. However, when favors cannot be granted simultaneously, altruism may not guarantee satisfactory future payback, especially when the future is un-

predictable. This makes cooperation stimulation in autonomous mobile ad hoc networks an extremely challenging task.

Second, in wireless networks, noise is inevitable and can cause severe trouble. For two-player cheat-proof packet forwarding strategy, if some packets are dropped due to noise, the game will be terminated immediately, and the performance will be degraded dramatically. This will also happen in most existing cooperation enforcement schemes, such as [31,78]. In these schemes, noise can easily lead to the collapse of the whole network, where finally all nodes will act non-cooperatively. Distinguishing the misbehavior caused by noise from those caused by malicious intention is a challenging task.

Third, since nodes can only base on what they have observed to make their decisions, imperfect monitoring can always be taken advantage of by greedy or malicious nodes to increase their performance. For example, when the miss detect ratio is high, a node can always drop other nodes' packets but still claim that it has forwarded. None of the existing approaches are designed with the consideration of noise and imperfect monitoring, which greatly limits their potential applications in realistic scenarios.

Fourth, since autonomous mobile ad hoc networks are usually deployed in adversarial environments, some nodes may even be malicious. If there exist only selfish nodes, stimulating cooperation will be much easier according to the following logic as demonstrated in [31]: misbehavior can result in the decrease of service quality experienced by some other nodes, which may consequently decrease the quality of service provided by them; this quality degradation will then be propagated back to the misbehaving nodes. Therefore selfish nodes have no incentive to intentionally behave maliciously in order to enjoy high quality of service. How-

ever, this is not true when some nodes are malicious. Since the attackers' goal is to decrease the network performance, such quality degradation is exactly what they want to see. This makes cooperation stimulation in hostile environments extremely challenging. Unfortunately, malicious behaviors have been heavily overlooked when designing cooperation stimulation strategies.

In order to formally analyze cooperation and security in such networks, similar as in Chapter 5, we also model the interactions among nodes as **secure routing and packet forwarding game**:

- **Players:** A finite set of network users, denoted by N.
- **Types:** Each player $i \in N$ has a type $\theta_i \in \Theta$ where $\Theta = \{selfish, malicious\}$. Let N_s denote the set of selfish players and $N_m = N - N_s$ the set of attackers. Meanwhile, no play knows the others' types a priori.
- **Strategy space:**
  1. **Route participation stage**: For each player, after receiving a request asking it to be on a certain route, it can either *accept* or *refuse* this request.
  2. **Route selection stage**: For each player who has a packet to send, after discovering a valid route, it can either *use* or *not use* this route to send the packet.
  3. **Packet forwarding stage**: For each relay, once it has received a packet requiring it to forward, its decision can be either *forward* or *drop* this packet.
- **Cost:** For any player i, transmitting a packet, either for itself or for the others, will incur cost c_i.

- **Gain:** For each selfish player i, it can get gain g_i for any successfully delivered packet originating from it.

- **Utility:** For each player i, let $T_i(t)$ denote the number of packets that i needs to send by time t, let $S_i(t)$ denote the number of packets that have successfully reached their destinations by time t with i being the source, let $F_i(j,t)$ denote the number of packets that i has forwarded for j by time t, and let $F_i(t) = \sum_{j \in N} F_i(j,t)$. Let $W_i(j,t)$ denote the total number of useless packet transmissions that i has caused to j by time t due to i dropping those packets transmitted by j. Let t_f be the lifetime of this network. Then we model the players' utility as follows:

  1. For any selfish player i, its objective is to maximize

  $$U_i^s(t_f) = \frac{S_i(t_f)g_i - F_i(t_f)c_i}{T_i(t_f)}. \tag{7.12}$$

  2. For any attacker j, its objective is to maximize

  $$U_j^m(t_f) = \frac{1}{t_f}\sum_{i \in N} \left(W_j(i,t_f) + F_i(j,t_f)\right) c_i - \alpha F_j(t_f)c_j. \tag{7.13}$$

  Here α is introduced to determine the relative importance of attackers' cost comparing to other nodes' cost. That is, it is worth spending cost c to cause damage c' to other nodes only if $\alpha < \frac{c'}{c}$. If the game will be played for an infinite duration, their utilities will become $\lim_{t_f \to \infty} U_i^s(t_f)$ and $\lim_{t_f \to \infty} U_j^m(t_f)$, respectively.

On the right-hand side of (7.12), the numerator denotes the net profit (i.e., total gain minus total cost) that the selfish node i obtained, and the denominator denotes the total number of packets that i needs to send. This utility represents the average net profit that i can obtain per packet. We can see that maximizing

(7.12) is equivalent to maximizing the total number of successfully delivered packets subject to the total cost constraint. If $c_i = 0$, this equals to maximizing the throughput.

The summation in the right-hand side of (7.13) represents the net damage caused to the other nodes by j. Since in general this value may increase monotonically, we normalize it using the network lifetime t_f. Now this utility represents the average net damage that j caused to the other nodes per time unit. From (7.13) we can see that in this game setting the attackers' goal is to waste the other nodes' cost (or energy) as much as possible. Other possible alternatives, such as minimizing the others' payoff, will also be discussed later.

7.3 Attack-Resistant Cooperation Stimulation

Before devising attack-resistent cooperation stimulation strategies, we first study how to handle possible malicious behavior. We focus on two classes of attacks: drop packet attack and inject traffic attack. Next we show how to detect drop packet attack under noise and imperfect monitoring.

Let $R_i(j,t)$ denote the number of packets that j has agreed to forward for i by time t, and let $H_i(j,t)$ denote the times that i has observed j forwarding a packet for it. If j has never intentionally drop i's packets, given p_e, p_f and p_m, for a large $R_i(j,t)$, we should have

$$p_o R_i(j,t) \leq H_i(j,t) \leq (1 - p_e + p_e p_m) R_i(j,t), \tag{7.14}$$

with $p_o = (1-p_e)(1-p_f)$. Then a simple detection rule can be as follows: node i will mark node j as intentionally dropping packets if the following holds

$$H_i(j,t) < R_i(j,t) p_o - \Delta(R_i(j,t), p_e, p_f, p_m), \tag{7.15}$$

where $\Delta(n, p_e, p_f, p_m)$ is a function of p_e, p_f, p_m, and n. In general, there is a tradeoff when selecting $\Delta(n, p_e, p_f, p_m)$. A large $\Delta(n, p_e, p_f, p_m)$ may incur high miss detect ratio, while a small $\Delta(n, p_e, p_f, p_m)$ may result in high false alarm ratio.

Next we show how to choose a good $\Delta(n, p_e, p_f, p_m)$. If the packet dropping due to noise can be modeled as an i.i.d. random process with drop probability p_e, and the observation errors are also i.i.d. random processes, and all are independent of each other, then according to the central limit theorem [49], for any $x \in \mathcal{R}$, we have

$$\lim_{R_i(j,t)\to\infty} Prob\left(\frac{H_i(j,t) - R_i(j,t)p_o}{\sqrt{R_i(j,t)p_o(1-p_o)}} \geq x\right) \geq 1 - \Phi(x), \tag{7.16}$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt. \tag{7.17}$$

Then we can let

$$\Delta(n, p_e, p_f, p_m) = x\sqrt{np_o(1-p_o)}. \tag{7.18}$$

In this case, the false alarm ratio will be no more than $1 - \Phi(x)$ when $R_i(j,t)$ is large. Since in general $\Phi(x)$ can still approach 1 even for a small positive x, $\Delta(n, p_e, p_f, p_m)$ will be a very small value comparing to np_o for a large n. However, in general neither packet dropping nor observation error is i.i.d. Under such circumstances, if the above detection rule is used, the false alarm ratio will usually be larger than $1 - \Phi(x)$. To handle non-i.i.d. scenario, one way is to increase $\Delta(n, p_e, p_f, p_m)$, such as increasing x.

Let β denote i's confidence on its detection decision, which lies in the range of [0,1], with 0 indicating that i has not marked j as malicious and with 1 indicating that i is sure that j is malicious. Then we can have $\beta = \Phi(x)$ for the i.i.d. scenarios and $\beta < \Phi(x)$ for the non-i.i.d. scenarios. In the following of this chapter we will

use $\Delta(n, p_e, p_f, p_m, \beta)$ to denote the detection threshold with detection confidence β.

Once node i has marked node j as intentionally dropping packets, one possible rule is that it should not work with j again. However, such rule has a drawback that if j has been mistakenly marked as malicious, it can never recover, since i will not give it any chance. To overcome this drawback, we modify this decision rule such that j will be given chance to recover, which will be described in the following.

Next we present attack-resistant cooperation stimulation strategies. The strategies for non-malicious players involve decision making in the following three stages: route participation stage, route selection stage, and packet forwarding stage.

Route participation stage We first study what decision a selfish node i should make when it receives a route participation request from node j. First, if i has detected j as malicious with confidence β, with probability $1 - \beta$ it should immediately refuse this request. Second, even if j has not been marked as malicious by i, i should accept this request only if it believes that it can get help from j later. However, whether i can get help from j depends on a lot of uncertain factors, such as i's and j's future requests, the changing network topology, j's strategy, and so on. For example, if i will never need j's help, or if i knows that this is the last packet j will send, i will have no incentive to help j. However, due to the unpredictability of future and favors not being granted simultaneously, stimulating i to act cooperatively is difficult.

In this chapter we focus on the scenario that nodes will stay in the network for a relatively long time. We consider the following strategy: a node may first forward some packets for other nodes without getting instantaneous payback. However,

in order to be robust to possible malicious behavior (e.g., inject traffic attack) or greedy behavior (e.g., request more but return less), a node should not be too generous. Before formalizing the above strategy, we first introduce a simple procedure: let β be i's confidence on whether j is malicious, i then randomly picks a value r between 0 and 1, and will give j another chance if $r < 1 - \beta$. We refer to this procedure as *recovery check* procedure. Let $\widetilde{F}_j(i,t)$ be i's estimate of $F_j(i,t)$. Then the above strategy can be translated as follows: i will accept j's route participation request only if j has passed recovery check and the following holds:

$$F_i(j,t) - \widetilde{F}_j(i,t) < D_i^{max}(j,t) \tag{7.19}$$

We refer to $F_i(j,t) - \widetilde{F}_j(i,t)$ as i's estimated *balance* with j, and refer to $D_i^{max}(j,t)$ as *cooperation level*. Here $D_i^{max}(j,t)$ is a threshold set by i for two purposes: 1) stimulating cooperation between i and j, and 2) limiting the possible damage that j can cause to i. Setting $D_i^{max}(j,t)$ to be ∞ means that i will always help j, setting $D_i^{max}(j,t)$ to be $-\infty$ means that i will never help j. By letting $D_i^{max}(j,t)$ to be positive, i agrees to forward some extra packets for j without getting instant payback. Meanwhile, unlike acting fully cooperatively, the extra number of packets that i will forward for j will be no more than $D_i^{max}(j,t)$, which can limit that possible damage when j plays non-cooperatively. This is analogous to a credit card system where $D_i^{max}(j,t)$ can be regarded as the credit line that i sets for j at time t. Like credit card company adjusts your credit line, $D_i^{max}(j,t)$ can also be adjusted by i over time. In this sense, we can also refer to $D_i^{max}(j,t)$ as the *credit line* that i sets for j.

It is easy to see that an optimal setting of credit lines is crucial to effective cooperation stimulation in noisy and hostile environments. Next we use nodes i

and j as example to illustrate how to set good credit lines. If the request rates between i and j are constant, then setting the credit line to be 1 will be optimal in the sense that no request will be refused when two nodes have the same request rate. However, due to mobility and nodes' inherent variable traffic generation rates, the request rates between i and j are usually not constant. In this case, if the credit lines are set to be too small, some requests will be refused even when the average request rates between them are equal. Let $f_i(j,t)$ denote the number of times that i needs j to help forward packets by time t. If we set the credit lines as follows:

$$D_i^{max}(j) = \max_t\{1, f_i(j,t) - f_j(i,t)\}, \quad D_j^{max}(i) = \max_t\{1, f_j(i,t) - f_i(j,t)\}, \quad (7.20)$$

then except the first several requests, no other requests will be refused when the average request rates between them are equal. If the average request rates between them are not equal, assuming that $\lim_{t\to\infty} \frac{f_i(j,t)}{f_j(i,t)} > 1$, then no matter how large $D_i^{max}(j)$ is, a certain portion of i's requests will have to be refused. This makes sense: to maintain ceratin fairness, j has no incentive to forward more packets for i. This also suggests that arbitrarily increasing credit lines cannot always increase the number of accepted requests. It is worth pointing out that (7.20) requires $f_i(j,t)$ and $f_j(i,t)$ to be known by i and j. However, such prior knowledge is usually not available since each node may not know *a priori* the others' request rates as well as its request rates to the others. Extensive simulations haven been conducted to study the effect of cooperation level, and the results suggest that when all nodes almost have equal request rates, a relatively small cooperation level can work well.

In order for the above strategy to work well, node i needs to have a good estimate of $F_j(i,t)$ for any other node j and needs to select a good cooperation level. We first study how to get a good estimate of $F_j(i,t)$. If i can have accurate

knowledge of monitoring errors experienced by j, denoted by $\widetilde{p}_f$ and $\widetilde{p}_m$, then we should have

$$F_j(i,t)((1-p_e)(1-\widetilde{p}_f)+p_e\widetilde{p}_m) \simeq H_i(j,t). \tag{7.21}$$

Then a good estimate of $F_j(i,t)$ can be

$$\widetilde{F}_j(i,t) = \frac{H_i(j,t)}{(1-p_e)(1-\widetilde{p}_f)+p_e\widetilde{p}_m} \tag{7.22}$$

However, in general i cannot accurately estimate $\widetilde{p}_f$ and $\widetilde{p}_m$. In such scenarios, a more conservative estimate can be

$$\widetilde{F}_j(i,t) = \frac{H_i(j,t)}{(1-p_e)(1-p_f)}. \tag{7.23}$$

Consequently, j can take advantage of such inaccuracy to forward less packets for i, or ask i to forward more packets for it, which will be further investigated in later sections.

It has been shown in [31] that topology will also play a critical role in enforcing cooperation for fixed ad hoc networks, and in most situations cooperation cannot be enforced. For example, a node in a bad location may never be able to get help from other nodes due to that no one will need it to forward packets. In this work we focus on *mobile* ad hoc networks. In such networks, a node in a bad location at a certain time may move to a better location later, or vice versa. This suggests that when a node receives a packet forwarding request from another node, it should not refuse this request *only simply* because the requester cannot help it currently, since the requester may be able to help later. That is, mobility can help alleviate the effect of topology dependence.

Route selection stage: Next we study the strategy in the route selection stage. Once a set of routes have been discovered by node i with all relays on these routes having agreed to forward packets for it, the following strategy will be taken by i:

first, i will not further consider this route if any relay cannot pass recovery check; second, among all those routes with all nodes having passed recovery check, i will pick the one with the minimum number of hops.

Packet forwarding stage: Now we consider the strategy in the packet forwarding stage. For any selfish node, once it has agreed to forward a packet for a certain node, it should not intentionally drop this packet unless the following can hold:

$$(1 - p_e)(1 - \widetilde{p}_f) + p_e\widetilde{p}_m \leq \widetilde{p}_m. \tag{7.24}$$

That is, $\widetilde{p}_f + \widetilde{p}_m \geq 1$, where $\widetilde{p}_f$ and $\widetilde{p}_m$ are the actual false alarm ratio and miss detect ratio experienced by the node. If (7.24) holds, this means that the chance that it will be marked as malicious even after dropping all the packets will still be no more than forwarding all packets due to the high monitoring inaccuracy. However, if (7.24) cannot hold, intentionally dropping packets will not be a good strategy if it still need others' help, since such dropping may cause it to be detected as malicious and consequently cannot get help from other nodes in the future.

Let $\beta(i, j)$ denote i's confidence on whether j is malicious. By combining the attacker detection strategy and the routing and packet forwarding strategies described above, we devise the following attack-resistant cooperation stimulation strategy:

Attack-Resistant Cooperation Stimulation Strategy: *For each single routing and packet forwarding subgame, assuming that P_1 is the initiator who wants to send a packet to P_n at time t, and a route "$P_1 \rightarrow P_2 \rightarrow \cdots \rightarrow P_n$" has been discovered by P_1. After P_1 has sent requests to all the relays on this route asking them to participate, for each non-malicious player on this route the following strategies should be taken:*

1. In the route participation stage: For any relay P_i, it will accept this re-

quest if and only if P_1 can pass recovery check and $F_{P_i}(P_1,t) - \widetilde{F}_{P_1}(P_i,t) < D_{P_i}^{max}(P_1,t)$; otherwise, it should refuse.

2. *In the route selection stage: P_1 will use this route if and only if all relays on this routes have passed recovery check and this route has the minimum number of hops among all those routes with all relays having passed recovery check; otherwise, P_1 should not use this route.*

3. *In the packet forwarding stage: For any relay P_i, it will forward this packet if and only if it has agreed to be on this route and (7.24) does not hold; otherwise, it should drop.*

4. *Attacker detection: Let β be an acceptable false alarm ratio from P_1's point of view. Then it will mark a relay P_j as malicious if (7.15) holds with $i = P_1$, $j = P_i$. Consequently, P_1 updates $\beta(P_1, P_j)$ using β.*

7.4 Strategy Analysis under Noise yet Perfect Monitoring

We first analyze the optimality of the proposed strategies under noise but perfect monitoring. We first consider an infinite lifetime situation with $T_i(t) \to \infty$ as $t \to \infty$. The finite lifetime situation will be discussed later. We assume that credit lines are set in such a way that 1) for any node i,

$$\lim_{t\to\infty} \frac{D_i^{max}(j,t)}{T_i(t)} = 0, \tag{7.25}$$

and 2) for any pair of nodes i and j, when $\lim_{t\to\infty} \frac{f_i(j,t)}{f_j(i,t)} \leq 1$, at most a finite number of i's requests will be refused by j.

Lemma 7.4.1 *For any selfish node $i \in N$ in the secure routing and packet forwarding game with no attackers, once i has received a route participation request from any other node $j \in N$, if the attack-resistant cooperation strategy is used by player j, then accepting the request is always an optimal decision from the point of view of player i.*

Proof From player i's point of view, refusing the request may cause it to lack enough balance to request player j to forward packets for it in the future (i.e., $D_j(i,t) > D_j^{max}(i,t)$), while agreeing to forward the packet will not introduce any performance loss due to the assumption (7.25). Therefore, accepting the request is an optimal decision. ■

Lemma 7.4.2 *In the secure routing and packet forwarding game where some packet forwarding decisions may not be perfectly executed, from the point of view of any player $j \in N$, if the multi-node attack-resistant and cheat-proof cooperation strategy is followed by all the other nodes, in the packet forwarding stage intentionally dropping a packet that it has agreed to forward cannot bring it any gain.*

Proof When a player $j \in N$ intentionally drops a packet that it has agreed to forward for any other player $i \in N$, it cannot get any gain except saving the cost to transmit this packet. However, since player i follows the multi-node attack-resistant and cheat-proof cooperation strategy, that is, it will always try to maintain $\lim_{t\to\infty} \frac{F_i(j,t)}{F_j(i,t)} \geq 1$, by dropping this packet, player j also loses a chance to request player i to forward a packet for it. To get the chance back, player j has to forward another packet for player i. Therefore, intentionally dropping a packet cannot bring any gain to player j. ■

Theorem 7.4.3 *In the secure routing and packet forwarding game with no attack-*

ers, the strategy profile that all players follow the multi-node attack-resistant and cheat-proof cooperation strategy forms a subgame perfect equilibrium, is cheat-proof, and achieves absolute fairness in cost if $c_i = c$ *for all* $i \in N$*. If* $0 < \lim_{t\to\infty} \frac{T_i(t)}{T_j(t)} < \infty$ *for any* $i, j \in N$*, this strategy profile is also strongly Pareto optimal.*

Proof We first prove that this strategy profile forms a subgame perfect equilibrium. Since this multi-player game can be decomposed into many two-player subgames, we only need to consider the two-player subgame played by player i and player j. Suppose that player j does not follow the above strategy, that is, either it will refuse to forward packets for player i when it should, or will intentionally drop packets that it has agreed to forward for player i, or it will forward more packets than it should for player i, or it will use non-minimum cost routes to send packets. First, from Lemma 7.4.1 and Lemma 7.4.2 we know that refusing to forward packets for other players when it should or intentionally dropping packets that it has agreed to forward will not introduce any performance gain. Second, forwarding much more packets (i.e., more than $D_i^{max}(j,t)$) than player j has forwarded for it will not increase its own payoff too according to the assumption of credit line selections. Third, using a non-minimum cost route to send packet will decrease its expected gain. Based on the above analysis we can conclude that the above strategy profile (the multi-player attack-resistant and cheat-proof cooperation strategy) forms a Nash equilibrium. To check that the profile is subgame perfect, note that in every subgame off the equilibrium path the strategies are either to play non-cooperatively forever if player j has dropped a certain number of packets that it has agreed to forward for player i, which is a Nash equilibrium, or still to play the multi-player cheat-proof packet forwarding strategy, which is also a Nash equilibrium.

Since no private information has been involved, based on the analysis presented in section 7.1, we can conclude that the proposed cooperation stimulation strategy is cheat-proof.

Since we have $F_i(j,t) - F_j(i,t) < D_i^{max}(j,t)$ for any player $i, j \in N$ and $\lim_{t\to\infty} \frac{D_i^{max}(j,t)}{T_i(t)} = 0$, and we have assumed that $c_i = c$ for all $i \in N$, it always holds that

$$\lim_{t\to\infty} \frac{\sum_{j\in N, j\neq i} F_i(j,t)}{\sum_{j\in N, j\neq i} F_j(i,t)} = 1. \tag{7.26}$$

That is, this strategy can achieve absolute fairness in cost.

Now we show that the strategy profile is strongly Pareto optimal. From payoff function (7.12) we can see that to increase its own payoff, a player i can either try to increase $S_i(t)$ or decrease $F_i(t)$. However, according to the above strategy, minimum cost routes have been used, therefore $F_i(t)$ cannot be further decreased without affecting the others' payoff. In order to increase its payoff, the only way that player i can do is to increase $\lim_{t\to\infty} \frac{S_i(t)}{T_i(t)}$, which means that some other players will have to forward more packets for player i. Since all $T_i(t)$'s are in the same order, increasing player i's payoff will definitely decrease other players' payoff. Therefore the above strategy profile is strongly Pareto optimal. ■

In the proof of Theorem 7.4.3 we have assumed that 1) $D_i^{max}(j,t)$ is large enough such that forwarding $D_i^{max}(j,t)$ more packets than player j has forwarded for it will not increase its own payoff, and 2) $D_i^{max}(j,t)$ is also small enough such that $\lim_{t\to\infty} \frac{D_i^{max}(j,t)}{T_i(t)} = 0$. If $D_i^{max}(j,t)$ cannot satisfy the above two requirements, the proposed strategy profile is not necessarily a Nash equilibrium. Finding a $D_i^{max}(j,t)$ value to satisfy the first requirement is easy, while to satisfy both requirements may be difficult or may even be impossible when nodes' requests rates and mobility patterns are not known *a priori*, which also explains why stimulating

cooperation in real autonomous mobile ad hoc networks is extremely challenging. However, our simulation results show that in many situations even an non-optimal $D_i^{max}(j,t)$, such as a reasonably large constant, can still effectively stimulate cooperation.

From the above analysis we can see that as long as g_i is larger than certain value, such as $(1-p_e)^{L_{max}} g_i > L_{max} c_i$ where L_{max} is system parameter to indicate the maximum possible number of hops that a route is allowed to have, then varying g_i will not change the strategy design.

Until now we have mainly focused on the situation that the game will be played for an infinite duration. In most situations, a node will only stay in the network for a finite duration. In this case, for each player i, if $D_i^{max}(j)$ is too large, due to its finite staying time, it may have helped its opponents much more than its opponents have helped it, while if $D_i^{max}(j)$ is too small, it may suffer the problems of lacking enough nodes to forward packets for it. How to select a good $D_i^{max}(j)$ remains as a challenge. section 7.6 has studied the tradeoff between the value of $D_i^{max}(j)$ and the performance through simulations, which shows that under given simulation scenarios a relatively small $D_i^{max}(j)$ value will be good enough to achieve near-optimal performance (compared with setting $D_i^{max}(j)$ to be ∞) and good fairness (compared with absolute fairness in cost). Here it is also worth pointing out that the optimality of the proposed strategies cannot be guaranteed in finite duration scenarios.

Now we analyze the multi-node attack-resistant and cheat-proof cooperation strategies in the presence of attackers. The following two widely used attack models are considered: dropping packet attack and injecting traffic attack. To simplify our illustration, we assume that $c_i = c$ and $g_i = g$ for all $i \in N$.

We first study dropping packet attack. By dropping other nodes' packets, attackers can decrease the network throughput and waste other nodes' limited resources, such as energy. Recall that an attacker detection mechanism has been applied in the proposed cooperation strategy, from an attacker's point of view, dropping all the packets passing it may not be a good strategy since this can be easily detected and a detected attacker cannot cause damage any more. Intuitively, in order to maximize the damage, attackers should selectively drop some portion of packets to avoid being detected. According to the multi-node attack-resistant and cheat-proof cooperation strategy, the maximum number of packets that an attacker can drop without being detected is upper-bounded by $np_e + x\sqrt{np_e(1-p_e)}$ where n is the times that it has agreed to forward. That is, it has to forward at least $n(1-p_e) - x\sqrt{np_e(1-p_e)}$ packets. However, among those dropped packets, $n(1-p_e)$ packets are due to noise, which will be there even no attackers are present. Thus, the extra damage is upper-bounded by $x\sqrt{np_e(1-p_e)}c$, while the extra cost is $n(1-p_e)c - x\sqrt{np_e(1-p_e)}c$. Since for any constant value $x \in \mathcal{R}^+$ we have

$$\lim_{n\to\infty} \frac{x\sqrt{np_e(1-p_e)}}{n(1-p_e)} = 0, \tag{7.27}$$

selectively dropping packets can bring no gain to the attackers. In other words, if the game will be played for an infinite duration, dropping packet attack cannot cause damage to selfish nodes.

Now we study injecting traffic attack. By injecting an overwhelming amount of packets to the network, attackers can consume other nodes' resources (e.g., energy) once they help the attackers forward these packets. Since for each selfish node $i \in N_s$, we have $D_i(j,t) \leq D_i^{max}(j,t)$, the maximum number of packets that an attacker j can request i to forward without paying back is upper-bounded by $D_i^{max}(j,t)$. Therefore, the damage that can be caused by injecting traffic attack is

bounded and limited, and will become negligible when $\lim_{t\to\infty} \frac{D_i^{max}(j,t)}{T_i(t)} = 0$ (e.g., $D_i^{max}(j,t)$ is a large constant).

Based on the above analysis we can also see that when the multi-node attack-resistant and cheat-proof strategy is used by all selfish nodes, attackers can only caused limited damage to the network. Further, the relative damage will go to 0 when the game will be played for an infinite duration. Since $R_i(j,t)$ and $F_j(i,t)$ are in the same order, for any constant value $x \in \mathcal{R}^+$, we always have

$$\lim_{R_i(j,t)\to\infty} \frac{x\sqrt{R_i(j,t)p(1-p)}}{F_j(i,t)} = 0. \tag{7.28}$$

Therefore, except some false positive, selfish players' overall payoff will not be affected under attacks. Although false positive may cause a node not to be able to get help from those who have been mistakenly detected as malicious or from those whom it has mistakenly detected as malicious, this will not become a big issue since the false positive probability can be made approach to 0 by using a large constant x without decreasing the overall payoff. From the above analysis we can also see that no matter what objectives the attackers have and what attacking strategy they use, as long as selfish nodes employ the multi-node attack-resistant and cheat-proof cooperation strategy, the selfish nodes' performance can be guaranteed.

Based on the above analysis we can conclude that for the infinite duration case an attacker j's overall payoff is upper-bounded by

$$U_j^m \leq \lim_{t\to\infty} \sum_{i\in N_s} \frac{D_i^{max}(j,t)}{t} c, \tag{7.29}$$

provided that all selfish nodes follow the multi-node attack-resistant and cheat-proof cooperation strategy. This upper-bound can be achieved by the following attacking strategy:

Optimal Attacking Strategy: *In the secure routing and packet forwarding*

game, for any attacker $j \in N_m$, it should always refuse in the route participation stage, should always pick the route including no attackers in the route selection stage, and should not forward packets in the packet forwarding stage.

Following similar arguments as in the proof of Theorem 7.4.3, we can also show that in the infinite duration secure routing and packet forwarding game, the strategy profile where all selfish players follow the multi-node attack-resistant and cheat-proof cooperation strategy and all attackers follow the above optimal attacking strategy forms a subgame perfect equilibrium, is cheat-proof and strongly Pareto optimal, and achieves absolute fairness in cost under some mild conditions.

When the game will only be played for a finite duration, the above attacking strategy is not optimal any more. Now the attackers can try to drop some nodes' packets without being detected, since the statistical dropping packet attacker detection will not be initiated unless having collected enough interactions to avoid high false positive probability. In this case, selfish nodes' performance will be degraded a little bit. However, as long as the game will be played for a reasonably long time, which is the focus of this chapter, the relative damage is still insignificant.

7.5 Strategy Analysis Under Noise and Imperfect Monitoring

This section analyzes the performance of the devised strategies, identifies the conditions under which they can or cannot work well and why, and quantifies the maximum possible damage that the attackers can cause.

7.5.1 Performance Analysis under No Attacks

We first consider the decisions made by the relays in the packet forwarding stage. As long as (7.24) does not hold and the source i can get an accurate estimate of $F_j(i,t)$, from any selfish node's point of view, the only gain after intentionally dropping a packet is saving cost c_j, while the penalty includes the increase of probability being marked as malicious by i and the decrease of the number of packets that i will forward for j in the future. Therefore j has no incentive to intentionally dropping packets in such scenarios.

What is the consequence of inaccurate estimate of $F_j(i,t)$? Let's assume that $\widetilde{p}_f$ and $\widetilde{p}_m$ are the actual false alarm and miss detect ratios experienced by j, and i does not know it. In this case, i may use (7.22) to estimate $F_j(i,t)$, and we have

$$\frac{F_j(i,t)}{\widetilde{F}_j(i,t)} \simeq \frac{(1-p_e)(1-p_f)}{(1-p_e)(1-\widetilde{p}_f)+p_e\widetilde{p}_m}. \tag{7.30}$$

If $\widetilde{p}_f < p_f$, then we have $\widetilde{F}_j(i,t) > F_j(i,t)$, and consequently

$$\lim_{F_i(j,t)\to\infty}\left(\frac{F_j(i,t)}{F_i(j,t)}\right) = \frac{(1-p_e)(1-p_f)}{(1-p_e)(1-\widetilde{p}_f)+p_e\widetilde{p}_m}. \tag{7.31}$$

In other words, node j can take advantage of imperfect monitoring to increase its performance by forwarding less packets for node i. However, if the underlying monitoring mechanism can guarantee p_f and p_m to be small enough, the damage caused to node i will be very limited. Further, if node i also experiences lower false alarm ratio, the damage will be further reduced, since the above analysis is also applicable to i. We can also check that if the false alarm ratio and miss detect ratio experienced by node i and j are the same, then we can still have $\lim_{F_i(j,t)\to\infty}\left(\frac{F_j(i,t)}{F_i(j,t)}\right) = 1$.

Next we consider the source's decision in the route selection stage. If no relays on the selected route have been marked as malicious by the source, it is easy to

see that this is an optimal selection. What is the consequence if some relays have been marked as malicious? First, with very small probability those nodes can pass recovery check, so even they are malicious, the long-term average damage is still negligible. Second, since these nodes may have been mistakenly marked as malicious, such chance can allow them to recover their reputation, and may consequently increase the source's future payoff, since it may have more resources to select and use.

Finally we analyze the relay's decision in the route participation stage. The optimality of the proposed strategy in this stage depends on a lot of uncertain factors, such as the nodes' future request pattern, the changing topology, the nodes' future staying time, the selection of good cooperation level, etc. Since most of these factors cannot be known a priori, the optimality of the proposed strategies cannot be guaranteed. It is usually impossible to find an optimal strategy without being able to accurately predict the future. However, our simulation results (Section 7.6) show that when nodes' request rates do not vary a lot, a relatively small cooperation level can work very well.

If the future is predictable, or at least partially predictable, such as the network will keep alive for a long time, all nodes staying in the network will keep generating and sending packets, and any pair of nodes will meet and request each other's help again and again, then each node can set its cooperation level to be a very large positive constant without affecting its overall performance (any extra constant cost will not affect the overall payoff as long as $\lim_{t\to\infty} T_i(t) = \infty$). Then the proposed strategies can form a Nash equilibrium, and are Pareto optimal, are subgame perfect, and achieve absolute fairness (in cost), provided that each node i can accurately estimate $F_j(i,t)$ for any other node j and $D_i^{max}(j,t)$ is large enough

to accommodate possible variable and bursty requests between them. The proof is easy by following the above analysis, which is not put here due to space limit. Unfortunately, such ideal scenarios may not exist in reality.

7.5.2 Attacking Strategy and Damage Analysis

Thus far we mainly focus on the scenarios that no nodes are malicious. Next we analyze the possible damage that can be caused by the attackers. In an abstract level, to damage the network, one can either drop other nodes' packet, or inject a lot of traffic to consume other nodes' resources. We first consider drop packet attack. According to the devised strategy, for attacker j, to avoid being marked as malicious by node i, the highest packet drop ratio p'_e that it can employ should satisfy the following inequality to avoid being detected:

$$(1-p_e)(1-p_f) \quad \leq \quad (1-p'_e)(1-\widetilde{p}_f) + p'_e\widetilde{p}_m, \tag{7.32}$$

where $\widetilde{p}_f$ and $\widetilde{p}_m$ are the actual false alarm ratio and miss detect ratio experienced by j. The best case from j's point of view is that $\widetilde{p}_f = 0$ and $\widetilde{p}_m = p_m$, then we can have

$$p'_e \leq \frac{p_e + (1-p_e)p_f}{1-\widetilde{p}_m} \tag{7.33}$$

Accordingly, the extra percentage of i's packets that can be dropped by j is upper-bounded by

$$p'_e - p_e \leq \frac{p_e\widetilde{p}_m + (1-p_e)p_f}{1-\widetilde{p}_m} \tag{7.34}$$

From (7.34) we can see that as long as p_e, p_m, and p_f are small, the extra damage will be limited. Meanwhile, in order to continuously drop i's packets, j also needs to forward at least $(1-p'_e)$ percentage of packets for i, which may also incur a lot

of cost to j. In other words, from an attacker's point of view, as long as $p'_e - p_e$ is not large, dropping other nodes' packets may not match its best interest by also taking into consideration of its own cost.

However, if an attacker can successfully exploit the underlying monitoring to avoid being detected, such as experiencing a high $\widetilde{p}_m$, then the extra number of packets it can drop without being detected can increase dramatically. According to (7.34), the extra damage will increase nonlinearly with the increase of $\widetilde{p}_m$. This suggests that it is critical to have a robust monitoring scheme to ensure that the monitoring error will not be too large. Actually, from (7.34) we can also see that even for $\widetilde{p}_m = 0.5$, $p'_e - p_e$ is still upper-bounded by $p_e + 2p_f$, which is still small as long as p_e and p_f are small.

For inject traffic attack, since each selfish node i will maintain $F_j(i,t) \sim F_i(j,t)$, for any node j, the extra number of packets that node j can request node i to forward is always bounded. According to (7.31), the maximum possible ratio between $F_i(j,t)$ and $F_j(i,t)$ is upper-bounded by $\frac{(1-p_e)(1-\widetilde{p}_f)+p_e\widetilde{p}_m}{(1-p_e)(1-p_f)}$ provided $\widetilde{p}_f + \widetilde{p}_m < 1$. Meanwhile, if the underlying monitoring mechanism can ensure that p_m and p_f are small, the ratio will be small. However, if j can successfully manage to let $\widetilde{p}_f + \widetilde{p}_m \geq 1$, such as making the miss detect ratio approach 1, it can always request i to forward packet without returning any favor.

It is worth noting that under the proposed strategies, no matter what goal the attackers may have, the selfish nodes' payoff can always be guaranteed as long as p_e, p_m and p_f are small. Meanwhile, if α (defined in (7.13)) is small enough, from an attacker's point of view, maximizing (7.13) is almost equivalent to minimizing the selfish nodes' payoff. Otherwise, maximizing (7.13) may not cause as much damage as minimizing the selfish nodes' payoff, since in this case the attackers

may not be willing to continuously drop packets without being detected due to the reason that this also requires the attackers to forward a lot of packets for other nodes and may not be their best interest.

7.5.3 Remarks

Compared with existing work, such as [6, 26, 31, 59, 78, 80], we address cooperation stimulation under very realistic scenarios: noisy environment, existence of (insider) attackers, mobile nodes, variable traffic rate, etc. This makes the task extremely challenging, and optimal solutions may not be always available. Meanwhile, our goal is not to enforce all nodes to act fully cooperatively, but to stimulate cooperation among nodes as much as possible.

One major difference between our scheme and the existing reputation-based schemes is that in our scheme pairwise relationship have been maintained by nodes. That is, each selfish node will keep track of the interactions with all other nodes experienced by it. The drawback is that it requires per-node monitoring and results in extra storage complexity. However, the advantage lies in that it can effectively stimulate cooperation in noisy and hostile environments. Meanwhile, for each node i, at any time it only needs to maintain the records $R_i(j)$, $isBad_i(j)$, $F_i(j)$, $F_j(i)$ for any other node j that i has interacted, so the maximum storage complexity is upper-bounded by $4|N|$. As long as $|N|$ is not too large, for most mobile devices, such as notebook and PDA, the storage requirement is insignificant.

In most existing work, such as in [6, 31, 78, 80], however, each node makes its decision based solely on its own experienced quality of service, such as throughput. Although the overhead is much lower than our scheme due to that only end-to-end acknowledge is required and each node only needs to keep its own past state, they

cannot effectively stimulate cooperation at all in noisy and hostile environments. The logic to base only on its own experience quality of service is that they expect no node will behave maliciously since misbehaviors will be propagated back later and the quality of service experienced by the misbehaving nodes will also be decreased. However, such logic cannot hold in noisy and hostile environments. First, attackers will be willing to see such performance degradation, therefore they will try to behave maliciously if possible. Second, even only noise can cause such misbehavior propagation and performance degradation since noise can cause packet dropping. Meanwhile, without per-node monitoring, attackers can always behave maliciously and cause damage to the others without being detected.

In [31, 78], when a node makes its cooperation decision at each step, it only bases on the normalized throughput that it has experienced. If only normalized throughput is used, a greedy user can set a low forwarding ratio, but try to send a lot of packets. Therefore, unless the others also try to send a lot of packets, from the greedy node's point of view, even after a large portion of its packets have been dropped by other nodes, it can still enjoy a high throughput, although the normalized throughput may be low. Meanwhile, as mentioned before, applying the same forwarding ratio to all nodes is not fair to those who have acted cooperatively. To resolve this problem, in our scheme, each node applies different packet forwarding decision for different node based on its past interactions between them.

Besides reputation-based cooperation stimulation schemes, pricing-based schemes have also been proposed in the literature, such as [7, 18, 19, 99, 100]. Comparing to pricing-based schemes, the major drawback of reputation-based schemes is that some nodes may not get enough help to send out all their packets. The most underlying reasons are that favors cannot be returned immediately and future is

not predictable. In other words, when a node is requested by another node to forward packets, since it cannot get compensation immediately and it is not sure whether the requester will return the favor later, it usually has no strong incentive to accept the request. Pricing-based schemes do not suffer such problems since a node can get immediate monetary payback after providing services. However, pricing-based schemes require tamper-proof hardware or central banking service to handle billing information, which is their major drawback. If such requirement can be efficiently satisfied with low overhead, pricing-based schemes can be a better choice than reputation-based schemes. Meanwhile, it is worth pointing out that pricing-based schemes also suffer from noise and possible malicious behavior, and the proposed statistical attacker detection mechanism is also applicable to pricing-based scenarios.

In general, necessary monitoring is needed when stimulating cooperation among nodes. For example, in [58], watchdog is proposed to detect whether some nodes have dropped packets. In this chapter we assume that the underlying monitoring mechanism can provide accurate per-node monitoring. Although this can be a strong assumption in some scenarios, it can greatly simplify our analysis and at the same time provide thoughtful insights. Meanwhile, this can be achieved by some recently proposed monitoring mechanisms, such as those proposed in [91,93]. It is worth mentioning that in some situations perfect monitoring is either not available or too expensive to afford. For example, the one proposed in [91] relies on asymmetric cryptography, which may be too expensive to afford for some resource-stringent mobile devices. The study of imperfect monitoring is beyond the scope of this chapter, but will be investigated in our future work.

In our analysis we have assumed that the packet drop ratio p_e is the same

for all nodes at all time, which may not hold in general. If different nodes may experience different p_e, the nodes experiencing lower p_e may experience high false positive probabilities when performing the proposed attacker detection mechanism. In this case, to decrease false positive probability, nodes need set the threshold to be large enough, that is, using a larger x and p_e in (7.15). Although this may be taken advantage of by the attackers to cause more damage, as long as the gap between the packet dropping ratios experienced by different nodes is not large, which is usually the case, the extra damage is still limited.

It is also worth mentioning that the security of the proposed strategy also relies on the existing secure protocols to achieve secure access control and secure authentication, and to defend those attacks launched during route discovery, such as those in [34–37, 42, 65, 76, 90, 91, 93, 95, 101]. In general, besides dropping packets and injecting traffic, attackers can also have a variety of ways to attack the network, such as jamming, slander, etc. In this chapter our focus is not to address all these attacks, but to provide insight on stimulating cooperation in noisy and hostile environments. To the best of our knowledge, we are the first one to formally address this issue under such realistic scenarios.

7.6 Simulation Studies

In this section we conduct extensive simulations to evaluate the effectiveness of the devised strategy and to identify when and why in some situations these strategies cannot work well.

In our simulations, both static and mobile ad hoc networks have been studied, with mobile ad hoc network being our focus. In these simulations, nodes are randomly deployed inside a rectangular area of $1000m \times 1000m$, and each mobile

node moves according to the *random waypoint* model, with $v_{min} = 10m/s$, $v_{max} = 30m/s$, and the average pause time 100s. The maximum transmission range is 250m.

In these simulations, each node randomly picks another node as the destination to send packets. The total number of selfish nodes is 100. Both p_m and p_f are set to be 5%, and β is set to be 0.1%. Each packet has a delay constraint, which is set to be 10s. If a packet is dropped by some relay, no retransmission will be applied. For each node i, we set $g_i = 1$ and $c_i = 0.1$. The nodes are indexed from 1 to N, where N is the total number of nodes.

To conduct performance evaluation and comparison, the following are measured for each selfish node in the simulations:

- *Normalized throughput*: this is the ratio between the total number of successfully delivered packets and the total number of packets scheduled to be sent.

- *Probability of no route available*: this is the percentage packets dropped due to no valid route is available.

- *Cost per successful packet delivery*: this is the ratio between the total number of forwarded packets (both for itself and for the others) and the total number of successfully delivered packets originating from it.

- *Balance* this is the difference between the total number of packets that this node forwarded for the others and the total number of packets that the others forwarded for it.

According to (7.12) it is easy to see that a selfish node's payoff can be easily calculated based on its normalized throughput and the cost per successful packet

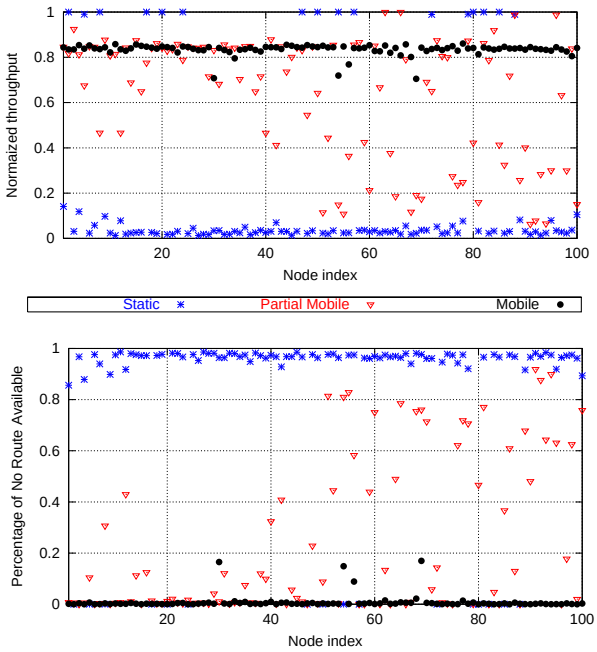


Figure 7.4: Effects of mobility on cooperation stimulation

delivery.

7.6.1 Mobile Ad Hoc Networks vs. Static Ad Hoc Networks

We first study the effect of mobility on cooperation stimulation. In this set of simulations, three types of networks are generated: mobile, partial mobile, and static. In the partial mobile ad hoc network, the nodes with indices ranging from 1 to 50 are mobile, and the other half are static. All nodes employ the same traffic pattern: the packet inter-arrival time follows exponential distribution with mean being 2s. All nodes set their cooperation level to be 60. The simulation results are illustrated in Fig. 7.4.

First, from the throughput comparison we can see that for the static case, except several nodes, the majority of nodes (85%) experience extremely bad through-

put. This is due to the reason that at most times they cannot find a route with all relays willing to help it (shown in the second figure). For those several nodes with high normalized throughput, the reason is that the destinations are in the transmission range of the sources. These results suggest that the devised strategies cannot be used in static ad hoc networks. Actually, in [31, 100] the authors have demonstrated that in networks with fixed topology, cooperation enforcement is impossible to achieve by relying solely on reputation. The most basic reason is that the service that a node can provide is usually not needed by its neighbors, therefore its neighbors have no incentive to help it.

From these results we can also see that when all nodes are mobile, the normalized throughput can be fairly high. For example, except 4 nodes, all the other nodes have normalized throughput being more than 80%. Even for those four nodes, their normalized throughput is still more than 70%. We can also see that for the majority of the nodes (96%), almost none of their packets are dropped due to no available routes, that is, cooperation among nodes has been effectively stimulated. Actually, the positive effect of mobility has also been noticed and studied in many other works. For example, Luo et. al. have studied how mobility can help improving lifetime in wireless networks and help improving the reliability of wireless ad hoc networks [56], and Capkun et. al. have studied how mobility helps security in ad hoc networks [24].

Now we study the partial mobile case. From the throughput comparison we can see that for those mobile nodes, no one has normalized throughput less than 40%, and the majority (33 out of 50) have normalized throughput higher than 80%. However, for those static nodes, the situation is totally reversed: half of them have normalized throughput less than 40%. This suggests that mobility can

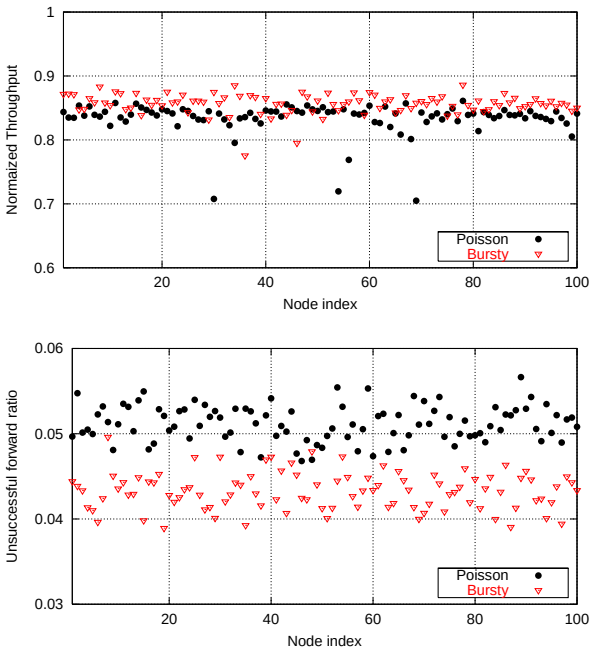


Figure 7.5: Effects of traffic pattern on cooperation stimulation

help stimulating cooperation. The underlying reason is that mobility can make the service exchange more effectively. An analogy to this is the effect of businessman: without them, we can only exchange service locally, the service we can get will be very limited; while with the help of businessman, service can be exchanged globally. From now on, we will mainly focus on mobile ad hoc networks with all nodes being mobile.

7.6.2 Bursty Traffic Pattern vs. Non-bursty Traffic Pattern

Next we investigate the effect of traffic pattern on cooperation stimulation. In these simulations two traffic patterns are considered: bursty and non-bursty. In bursty case, packets are generated in a bursty pattern with average bursty length 10, while in non-bursty pattern the packet arrival follows a Poisson process. In

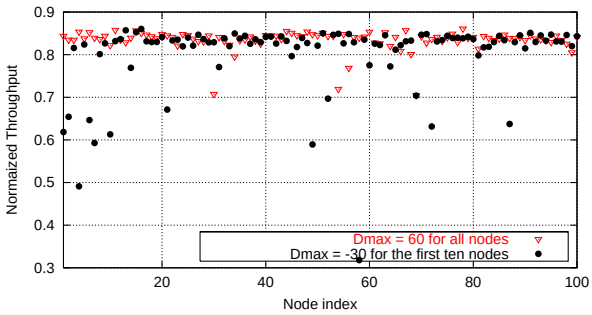


Figure 7.6: Effect of negative cooperation level on cooperation stimulation

both cases the average packet arrival rate is 0.5 packet/s. The simulation results are illustrated in Fig. 7.5.

It is surprising to see that bursty case has slightly better normalized throughput than non-bursty case. This can be explained using the unsuccessful forward ratio experienced by each node (shown in the second figure): in bursty case, the unsuccessful forward ratio experienced by each other is 1% lower than the non-bursty case. This is because in non-bursty case, when a packet needs to be sent, with a high probability the existing route may have broken since this route may be discovered a long time ago, while in bursty case, though link breakages also happen frequently, as long as the current route is good, almost all the packets can be delivered successfully. However, if nodes with bursty-pattern have much higher rates, or the burst length is much longer, the performance of bursty-case may be decreased, as to be shown later.

7.6.3 Effect of Negative Cooperation Level

In this set of simulations some nodes set their cooperation level to be negative. Specifically, the first ten nodes set D^{max} to be -30, and all the others set D^{max} to be 60. The results are illustrated in Fig. 7.6. From these results we can see that

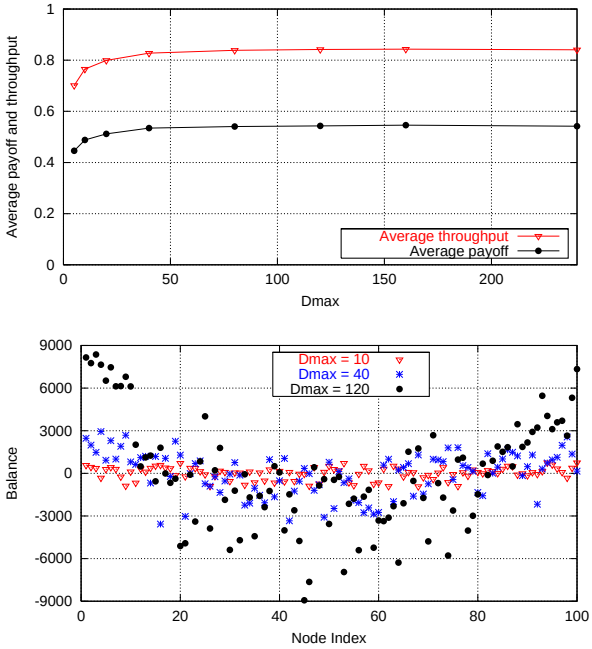


Figure 7.7: Effect of cooperation level on cooperation stimulation

the majority of nodes (6 out of ten) who set D^{max} to be negative have normalized throughput less than 65%. Meanwhile, they also cause some other nodes to experience lower normalized throughput (6 out of 90 have normalized throughput no more than 70%). These results suggest that as long as a node wants to stay in the network for a long time and needs to send packets continuously, they should not set their cooperation level to be negative.

7.6.4 Effect of Cooperation Level on Cooperation Stimulation

In this set of simulations, each node sets its traffic rate to be 0.5 packet/s following Poisson arrival. In each simulation different D^{max} value is used ranging from 10 to 240. The results are illustrated in Fig. 7.7. From the first figure we can see that once $D^{max} \geq 80$, both the average normalized throughput and the average

payoff experienced by selfish nodes do not increase further, which suggests that in this case setting $D^{max} = 80$ can almost approach the optimal solution in term of normalized throughput. However, from the second figure we can see that with the increase of $D^{max} \geq 80$, the balance variation experienced by nodes also increase, which leads to high unfairness. That explains why we have set $D^{max} = 60$ in our simulations: a good tradeoff between payoff and fairness.

7.6.5 Effect of Inhomogeneous Request Rates

In this set of simulations, each node's traffic rate is determined as follows: let i be a node's index ranging from 1 to 100, then its traffic rate will be set as $((i \mod 20) + 1)/2$ packet/s. Based on the configuration of D^{max} and traffic pattern, three cases are studied: in case 1 and 3, for each node its traffic follows Poisson arrival, while in case 2 each node's traffic follows a bursty arrival. Meanwhile, in case 1 and 2, all nodes set D^{max} to be 60, while in case 3, each node with index i set D^{max} to be $60 + (i \mod 2)$. The results are shown in Fig.7.8.

We first study the throughput comparison. From these results we can see that case 3 has the highest normalized throughput while case 2 has the lowest normalized throughput. This suggests that bursty traffic may decrease the performance, while if a node has too much traffic to send, increasing their cooperation level can increase their performance. From these results we can also see that with the increase of traffic rate, the throughput decreases too. Although increasing D^{max} can slightly increase the performance, it cannot completely solve the problem. The reason is that the service provided by those nodes with high traffic rate are not needed by those nodes with lower rates. This can be shown more clearly in the following simulations.

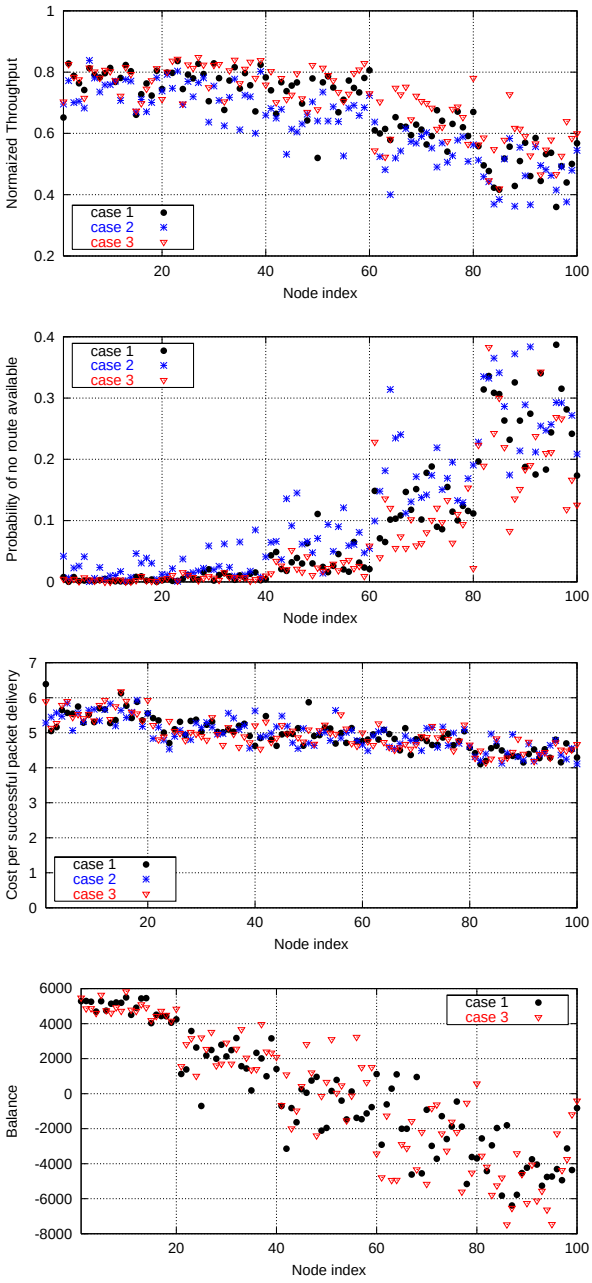


Figure 7.8: Effect of inhomogeneous request rates on cooperation stimulation

By checking the second figure (probability of no route available) in Fig.7.8, we can see that in case 2 (bursty case), a lot of packets will be dropped due to no available routes, especially when the node's traffic rate is high, which explains why they have lowest throughput. From the third figure (cost per successful delivery) in Fig.7.8 we can see that with the increase of traffic rate, the hop number per route may decrease slightly, which is a little bit surprising, but makes sense: when a node with high traffic rate has used up the quota assigned by those nodes with lower rate, they are forced to use short routes such as one-hop route. This is also confirmed by the results in the fourth figure, which indicates that for the first 20 nodes, their overall balance almost reaches to the maximum.

Next we study an extremely asymmetric case, where in this set of simulations, except the first ten nodes which have packet arrival rate 5 packet/s, all the other nodes have packet arrival rate 0.5 packet/s. According to the first ten nodes' D^{max} values, three cases are studied: in case 1 they let $D^{max} = 60$, in case 2 they set $D^{max} = 120$, and in case 3 they set $D^{max} = 180$. For the other nodes in all the three cases, $D^{max} = 60$. The results are illustrated in Fig. 7.9. From these results we can see that by increasing D^{max} from 60 to 120, a lot of gain can be obtained (normalized throughput increases from 8% to 22%), while increasing D^{max} from 120 to 180 introduces almost no gain, and the normalized throughput is still only about 22%. This suggests that although increasing D^{max} can provide some gain, they cannot change the inherent problem.

7.6.6 Effects of Different Drop Packet Attacks

In this set of simulation, we study the effect of different drop packet attacks. Four drop packet attack strategies are studied: not participate in any route discovery,

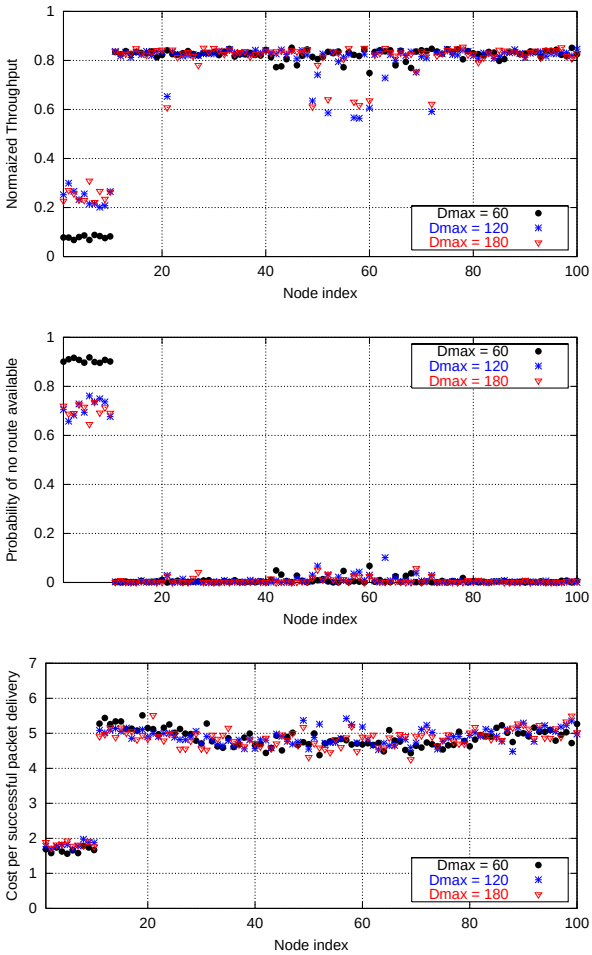


Figure 7.9: Effect of inhomogeneous request rates, an extreme case

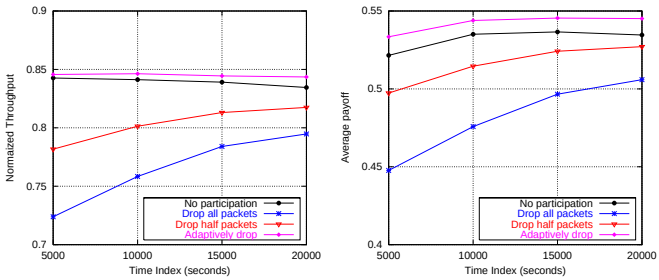


Figure 7.10: Comparison of different drop packet attacks

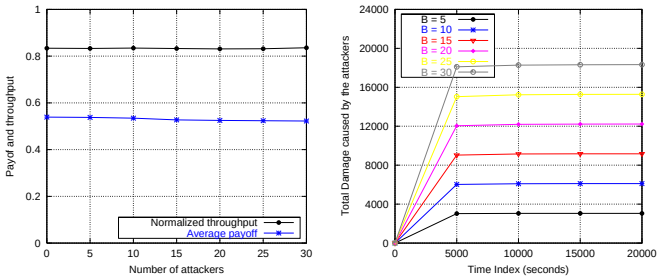


Figure 7.11: Performance comparison under different number of attackers

drop all packets passing through it, drop half of the packets passing through it, and selectively drop packets passing through it and at the same time keep avoiding being detected. Fig. 7.10 illustrates the evolution of the normalized throughput and payoff averaged among all selfish nodes over time. From these results, first we can see that dropping all packet can cause the maximum damage, the reason is the we have set B_{th} to be a large value (200), so each attacker can drop up to 199 of any other node's packets without being marked as malicious. However, we can also see that with time increasing, the selfish nodes' performance will also increase. From these results we can also see that adaptive dropping can even increase the selfish nodes' performance. This is because the damage it can cause is very limited in order to avoid being detected, while keeping forwarding packets for selfish nodes can reduce the selfish nodes' average hop number per selected route. Although intuitively adaptive dropping may cause a lot of damage, in reality this may not be the case.

7.6.7 Effect of Attacker Number

In this set of simulations we study the selfish nodes' average performance in the presence of different number of attackers, with the number of attackers ranging from 5 to 30. All attackers launch inject traffic attack, and will not forward any

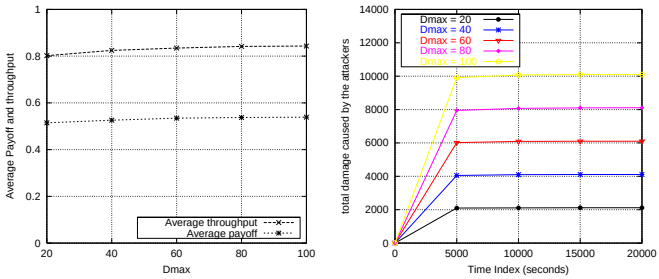


Figure 7.12: Effect of cooperation level on damage

packet for selfish nodes. The results are illustrated in Fig. 7.11. From these results we can see that with the increase of attacker number, the average normalized throughput among all selfish nodes keeps almost unchanged, and the average payoff only decreases very slightly. This can be explained using the second figure, where here total damage is defined as the total number of packets that selfish nodes have forwarded for each attacker. From this figure we can see that after some time, no more damage can be caused to selfish nodes due to the reason that they have used up all the quota assigned to them. This suggests that the proposed strategy is robust to inject traffic attack.

7.6.8 Cooperation Level vs. Damage

In this final set of simulations, the effect of D^{max} on selfish nodes' performance under inject traffic attack is studied, with the selfish nodes' D^{max} varying from 20 to 100. The results are illustrated in Fig. 7.12. From these results we can see that after D^{max} passes 60, the selfish nodes' average performance (normalized throughput and payoff) keep almost unchanged. Similar as the results illustrated in Fig. 7.11, for each given D^{max}, the damage caused by the attackers will not change after some time due to using up all the assigned quota. Meanwhile, the damage will increase linearly with the increase of D^{max}. By also taking into consideration of

fairness issue, these results also suggest $D^{max} = 60$ can be a good choice. However, we need to keep in mind that the selection of D^{max} also depends on the underlying traffic rate. It is easy to understand that with the increase of traffic rate, we should also increase D^{max}, especially when mobility is low and traffic may exhibits strong bursty pattern and/or variable rates.

7.7 Summary

In this chapter we have investigated the issues of cooperation stimulation for self-organized mobile ad hoc networks in a realistic context, where the communication channels are error-prone, the underlying monitoring is imperfect, and the environment is hostile with possible malicious behavior. We have identified the underlying reasons why stimulating cooperation among nodes under scenarios is extremely challenging. Unlike most existing work whose goal is to enforce all nodes to act fully cooperatively, our goal is to stimulate cooperation among selfish nodes as much as possible through reciprocal altruism. We have devised a set of reputation-based attack-resistant cooperation stimulation strategies, which are completely self-organizing and fully distributed, and do not require any tamper-proof hardware or central banking or billing service. Both theoretical analysis and extensive simulation studies have demonstrated that the devised strategies can effectively stimulate cooperation among selfish nodes under various scenarios and meanwhile is robust to attacks. Further, the conditions under which the devised strategies cannot work well have also been studied and we conclude that the most underlying reasons are that favors not being granted simultaneously and future being unpredictable.

Chapter 8

Conclusions and Future Work

In this thesis we have studied how to secure cooperative ad hoc network against insider attacks under noise and imperfect monitoring, and how to design attack-resistant cooperation strategies for autonomous ad hoc networks that can work well in noisy and hostile environments.

First, we have studied how to handle routing disruption attacks in mobile ad hoc networks. Most existing secure ad hoc routing protocols require significant overhead to implement extra security mechanisms, such as secure neighbor discovery and link-level data forwarding monitoring. In this dissertation we present a set of light-weight techniques, referred to as HADOF. We use a novel self-evaluation mechanism that can significantly speed up malicious node detection. With self-evaluation, a malicious node has to either admit dropping packets or provide reports that are most likely conflicting with others. Based on self-evaluation, a distributed cheating record is maintained to track nodes' long-term behavior. In addition, route diversity and adaptive route rediscovery mechanisms are employed to enable quick recovery from route disruption due to malicious attacks, mobility and traffic congestion. HADOF is capable of adaptively adjusting routing strategies

according to network dynamics, and nodes' past record and current performance. It can distinguish routing disruptions caused by nodes' temporary misbehavior and those caused by malicious attacks. It can defeat black hole, gray hole, frame-up, rushing attack, and wormhole attack. More importantly, HADOF introduces little overhead to the existing routing protocols.

Second, we have investigated the possible injecting traffic attacks that can be launched in mobile ad hoc networks, and proposed a set of mechanisms to defend against such attacks. Both query flooding attacks and injecting general data packets attacks have been investigated. Furthermore, for injecting general data packets attacks, the situations that attackers may use some advanced transmission techniques, such as directional antennas or beamforming, to avoid being detected have also been studied. Two set of defense mechanisms have been proposed, one is fully distributed, while the other is centralized with de-centralized implementation. Both theoretical analysis and simulation studies have been conducted, which have confirmed the effectiveness of the proposed defense mechanisms.

Third, we have formally investigated how to secure cooperative ad hoc networks against insider attacks under realistic scenarios, where the environment is noisy and the underlying monitoring is imperfect. We model the dynamic interactions between good nodes and attackers in such networks as securing routing and packet forwarding game. The optimal defense strategies have been devised, which are optimal in the sense that no other strategies can further increase the good nodes' payoff under attacks. The maximum possible damage that can be caused by the attackers have also been analyzed. By focusing on the worst-case scenario from the good nodes' point of view, that is, the good nodes have no prior knowledge of the other nodes' types while the insider attackers can know who are good nodes,

the devised strategies can work well under any scenarios. Extensive simulations have also been conducted to justify the underlying assumptions and to evaluate the proposed strategies.

Fourth, we have investigated the issues of cooperation stimulation and security in autonomous ad hoc networks, and proposed an Attack-Resistant Cooperation Stimulation (ARCS) system to stimulate cooperation among selfish nodes and defend against various attacks launched by malicious nodes. In the ARCS system, each node can adaptively adjust their own strategies according to the changing environments. The analysis has shown that in the ARCS system, the damage that can be caused by malicious nodes is bounded, and the cooperation among selfish nodes is enforced through introducing a positive cooperation degree. At the same time, the ARCS system maintains good fairness among selfish nodes. The simulation results have also agreed with the analysis. Another key property of the ARCS system is that it is fully distributive, completely self-organizing, and does not require any tamper-proof hardware or central management points.

Finally, we have formally investigated secure cooperation stimulation in autonomous mobile ad hoc networks under a game theoretic framework. Besides selfish behavior, possible attacks have also been studied, and attack-resistant cooperation stimulation have been devised which can work well under noisy and hostile environments. First, a simple yet illuminating two-player packet forwarding game is studied. To find good cooperation strategies, equilibrium refinements have been performed on obtained Nash equilibrium solutions under different optimality criteria, including subgame perfection, Pareto optimality, fairness, and cheat-proofing, and a unique Nash equilibrium solution is finally derived, which states that in the two-node packet forwarding game a node should not help its

opponent more than its opponent has helped it. The results are then extended to handle multi-node scenarios in noisy and hostile environments, where the dynamic interactions between nodes are modelled as secure routing and packet forwarding games. By taking into consideration the difference between two-node case and multi-node case, an attack-resistant and cheat-proof cooperation stimulation strategy has been devised for autonomous mobile ad hoc networks. The analysis has demonstrated the effectiveness of the proposed strategy, and shown that it is optimal under certain conditions. The analysis has also shown that the damage that can be caused by attackers is bounded and limited when the proposed strategies are used by selfish nodes. Simulation results have also illustrated that the proposed strategies can effectively stimulate cooperation among selfish nodes in noisy and hostile environments.

Although in this dissertation we have thoroughly addressed several critical issues in securing ad hoc networks, there still exist many issues that need further investigation. In the following of this chapter, we will list some of them that we would like to address in future.

The first topic we would like to address is about trust modeling and reputation propagation in distributed networks. From previous chapters we have learned that both attacker detection and cooperation stimulation involve the evaluation of nodes' trustworthiness. Meanwhile, when evaluating trustworthiness, direct observation alone may not be sufficient; indirect observation should also be considered, which results in reputation propagation. Our focus will be on robust trust modeling and reputation propagation. Besides building theoretic framework for trustworthiness evaluation in distributed networks, we would also like to investigate the possible attacks associated to trust model and reputation propagation, since

like all other schemes associated with security, trust management or reputation systems can also become attackers' targets. As a starting point in our exploration, we have discovered several new attacks on reputation and trust systems, which will lead to more exciting research challenges.

As mentioned before, effective monitoring is one of most crucial component in securing wireless ad hoc and sensor networks, which is an indispensable component in various aspects, such as intrusion detection, cooperation stimulation, and trust evaluation. In the future, we will continually design effective and robust monitoring mechanisms for ad hoc and sensor networks, and formally analyze their performance under different scenarios. We plan to systematically investigate the effect of imperfect monitoring on security and cooperation stimulation. Meanwhile We plan to design low-cost and attack-resistant monitoring mechanisms by also taking into consideration the tradeoff between monitoring accuracy and incurred overhead. We also plan to investigate how to perform monitoring in a collaborative way to simultaneously reduce the overhead and increase the accuracy.

The inherent characteristics and emergent properties of sensor networks make a node's location and time clock important parts of their state. Since sensor networks are usually deployed in hostile environments, security issues associated with localization and time synchronization must also be studied. Recently, some success has been achieved on securing localization and time synchronization services in sensor networks. However, most existing schemes assume ideal scenarios, such as ideal physical layer and perfect observation. Following similar methodologies as used in this dissertational, in the future we plan to investigate how to perform localization and time synchronization in noisy and hostile environments based only on local and imperfect observation.

BIBLIOGRAPHY

[1] Secure Hash Standard. Federal Information Processing Standards Publication 180-1, 1995.

[2] R. Castaneda A. Nasipuri and S. R. Das. Performance of Multipath Routing for On-Demand Protocols in Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications (MONET) Journal*, 6(4):339–349, 2001.

[3] I. Aad, J. P. Hubaux, and E. Knightly. Denial of service resilience in ad hoc networks. In *ACM MobiCom 2004*.

[4] E. Altman and T. Basar. Multi-user rate-based flow control. *IEEE Transactions on Communications*, pages 940–949, 1998.

[5] E. Altman, R. El-Azouzi, and T. Jimenez. Slotted Aloha as a stochastic game with partial information. In *WiOPT'03*.

[6] E. Altman, A. A. Kherani, P. Michiardi, and R. Molva. Non-cooperative Forwarding in Ad-Hoc Networks. Technical report, INRIA, Sophia Antipolis, France, 2004.

[7] L. Anderegg and S. Eidenbenz. Ad Hoc-VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *ACM MobiCom 2003*.

[8] N. Asokan and Philip Ginzboorg. Key agreement in ad hoc networks. *Computer Communications*, 23:1627–1637, 2000.

[9] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilent to byzantine failures. In *ACM Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, September 2002.

[10] R. Axelrod. *The Evolution of Cooperation*. New York: Basic Books, 1984.

[11] R. Axelrod. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University Press, 1997.

[12] L. Barriere, P. Fraigniaud, and L. Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *Workshop on Discrete Algothrithms and Methods for MOBILE Computing and Communications, Proceedings of the 5th international workshop on Discrete algorithms and methods for mobile computing and communications*, Rome, Italy, 2001.

[13] S. Bhargava and D. P. Agrawal. Security enhancements in aodv proocol for wireless ad hoc networks. In *Vehicular Technology Conference*, 2001.

[14] R. B. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh. Bootstrapping security associations for routing in mobile ad-hoc networks. In *IEEE Global Telecommunications Conference (GLOBECOM)*, 2003.

[15] S. Buchegger and J.-Y. Le Boudec. The effect of rumor spreading in reputation systems for mobile ad-hoc networks. In *WiOpt'03*.

[16] S. Buchegger and J.-Y. Le Boudec. Performance Analysis of the CONFIDANT Protocol. In *ACM Mobihoc 2002*.

[17] S. Buchegger and J.-Y. Le Boudec. Cooperative routing in mobile ad-hoc networks: Current efforts against malice and selfishness. In *Lecture Notes on Informatics, Mobile Internet Workshop, Informatik 2002*, Dortmund, Germany, October 2002.

[18] L. Buttyan and J. P. Hubaux. Enforcing Service availability in mobile Ad-hoc Network. In *ACM MobiHOC 2000*.

[19] L. Buttyan and J.-P. Hubaux. Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks. *Mobile Networks and Applications*, 8(5):579 – 592, Oct. 2003.

[20] M. Cagalj. *Thwarting Selfish and Malicious Behavior in Wireless Networks*. PhD thesis, Swiss Federal Institute of Technology - Lausanne, 2006.

[21] M. Cagalj, S. Capkun, and J.-P. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE (Special Issue on Security and Cryptography)*, 94(2):467–478, February 2006.

[22] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On selfish behavior in csma/ca networks. In *Proceedings of IEEE INFOCOM 05*, Miami, Florida, March 2005.

[23] S. Capkun and J.-P. Hubaux. BISS: Building Secure Routing out of an Incomplete Set of Security Associations. In *WiSe*, 2003.

[24] S. Capkun, J.-P. Hubaux, and L. Buttyan. Mobility Helps Security in Ad Hoc Networks. In *ACM MobiHOC 2003*.

[25] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyumn, and L. Viennot. Optimized Link State Routing Protocol. Internet-Draft, draft-ietf-manet-olsr-06.txt, Sep. 2001. Work in progress.

[26] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modelling Incentives for Collaboration in Mobile Ad Hoc Networks. In *WiOPT'03*.

[27] B. Dahill, B. N. Levine, E. Royer, and C. Shields. A secure routing protocol for ad hoc networks. In *Proceedings of the 10 Conference on Network Protocols (ICNP)*, 2002.

[28] R. Dawkins. *The Selfish Gene*. Oxford University Press, 2nd edition, 1990.

[29] J. Douceur. The sybil attack. In *the IPTPS02 Workshop*, 2002.

[30] M. Felegyhazi, L. Buttyan, and J.-P. Hubaux. Equilibrium Analysis of Packet Forwarding Strategies in Wireless Ad Hoc Networks - the Static case. In *Proceedings of Personal Wireless Communications*, 2003.

[31] M. Felegyhazi, J.-P. Hubaux, and L. Buttyan. Nash Equilibria of Packet Forwarding Strategies in Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 5(5):463–476, Sept.-Oct. 2006.

[32] D. Fudenberg and J. Tirole. *Game Theory*. The MIT Press, Cambridge, Massachusetts, 1991.

[33] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, Sep. 2001.

[34] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *ACM MobiCom 2002*.

[35] Y.-C. Hu, A. Perrig, and D. B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. In *IEEE INFOCOM 2003*.

[36] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols. In *WiSe*, 2003.

[37] Y.-C. Hu, A. Perrig, and D. B. Johnson. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. *Ad Hoc Networks Journal*, 1:175–192, 2003.

[38] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(2):370–380, February 2006.

[39] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Efficient security mechanisms for routing protocols. In *Proceedings of the Tenth Annual Network and Distributed System Security Symposium (NDSS 2003)*, San Diego, CA, 2002.

[40] Y. Huang, W. Fan, W. Lee, and P. S. Yu. Cross-feature analysis for detecting ad-hoc routing anomalies. In *the 23rd International Conference on Distributed Computing Systems (ICDCS)*, 2003.

[41] Y. Huang and W. Lee. A cooperative intrusion detection system for ad hoc netwroks. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, 2003.

[42] J. P. Hubaux, L. Buttyan, and S. Capkun. The Quest for Security in Mobile Ad Hoc Networks. In *ACM MobiHOC 2001.*

[43] J.-P. Hubaux, L. Buttyan, and S. Capkun. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, Jan.-March 2003.

[44] IEEE Computer Society LAN MAN Standards Committee. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11-1007. The Institue of Electrical and Electrics Engineers.

[45] J.-P. Hubaux and T. Gross and J.-Y. Le Boudec and M. Vetterli. Toward Self-Organized Mobile Ad Hoc Networks: The Terminodes Project. *IEEE Communications Magazine*, Jan. 2001.

[46] Y. Jin and G. Kesidis. Nash equilibria of a generic networking game with applications to circuit-switched networks. In *IEEE INFOCOM 2003.*

[47] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks, Mobile Computing. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[48] D. B. Johnson, D. A. Maltz, Y. C. Hu, and J. G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). Internet-Draft, draft-ietf-manet-dsr-07.txt, Feb. 2002. Work in progress.

[49] O. Kallenberg. *Foundations of Modern Probability.* New York: Springer-Verlag, 1977.

[50] Y. A. Korilis, A. A. Lazar, and A. Orda. Capacity allocation under noncooperative routing. *IEEE Transactions on Automatic Control*, 42:309–325, march 1997.

[51] Y. A. Korilis and A. Orda. Incentive compatible pricing strategies for QoS routing. In *IEEE INFOCOM 1999*.

[52] J. D. Kraus and R. J. Marhefka. *Antennas: for All Applications*. McGraw-Hill, New York, 3rd edition, 2002.

[53] P. Kyasanur and N. Vaidya. Selfish MAC Layer Misbehavior in Wireless Networks. *IEEE Transactions on Mobile Computing*, April 2004.

[54] R. La and V. Anantharam. Optimal routing control: Repeated game approach. *IEEE Transactions on Automatic Control*, 47(3):437–450, 2002.

[55] Y. W. Law, L. van Hoesel, J. Doumen, P. Hartel, and P. Havinga. Energy efficient link layer jamming attacks against wireless sensor network mac protocols. In *SANS05*, 2005.

[56] J. Luo. *Mobility in wireless networks: friend or foe: network design and control in the age of mobile computing*. PhD thesis, Swiss Federal Institute of Technology - Lausanne, 2006.

[57] J. MacKie-Mason and H. Varian. Pricing congestible network resources. *IEEE Journal on Selected Areas in Communications*, 13(7):1141–1149, 1995.

[58] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *ACM Mobicom 2000*.

[59] P. Michiardi and R. Molva. A Game Theoretical Approach to Evaluate Cooperation Enforcement Mechanisms in Mobile Ad hoc Networks. In *WiOPT'03*.

[60] P. Michiardi and R. Molva. Core: a COllaborative REputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. In *IFIP - Communications and Multimedia Security Conference*, 2002.

[61] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *IPSN04*, Berkeley, California, USA., April 2004.

[62] P. Ning and K. Sun. How to misuse aodv: A case study of insider attacks against mobile ad-hoc routing protocols. In *Proceedings of the 4th Annual IEEE Information Assurance Workshop*, West Point, June 2003.

[63] R. G. Ogier, F. L. Templin, B. Bellur, and M. G. Lewis. Topology Broadcast Based on Reverse-Path Forwarding (TBRPF). Internet-Draft, draft-ietf-manet-tbrpf-05.txt, Mar. 2002. Work in progress.

[64] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, Cambridge, Massachusetts, 1994.

[65] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, Jan 2002.

[66] P. Papadimitratos and Z. J. Haas. Performance evaluation of secure routing for mobile ad hoc networks. In *1st ACM Workshop on Wireless Security (WiSe)*, 2003.

[67] P. Papadimitratos, Z. J. Haas, and E. G. Sirer. Path Set Selection in Mobile Ad Hoc Networks. In *ACM MobiHOC 2002*.

[68] P. Papadimitratos and Z.J. Haas. Secure link state routing for mobile ad hoc networks. In *IEEE Workshop on Security and Assurance in Ad hoc Networks, in conjunction with the 2003 International Symposium on Applications and the Internet*, Orlando, FL, 2003 2003.

[69] C. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2000.

[70] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet-Draft, draft-ietf-manet-olsr-10.txt, Jan. 2002. Work in progress.

[71] R. Poisel. *Modern Communications Jamming Principles and Techniques*. Artech House, 2003.

[72] J. G. Proakis. *Digital Communications*. McGraw-Hill, 4th edition, 2000.

[73] Y. Qiu and P. Marbach. Bandwidth allocation in wireless ad hoc networks: a price-based approach. In *IEEE INFOCOM 2003*.

[74] M. Raya, J. Hubaux, and I. Aad. Domino: a system to detect greedy behavior in ieee 802.11 hotspots. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, 2004.

[75] T. Roughgarden. *Selfish Routing*. PhD thesis, Cornell University, May 2003.

[76] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of the International Conference on Network Protocols (ICNP)*, Nov. 2002.

[77] C. Schleher. *Electronic Warfare in the Information Age*. Artech House, 1999.

[78] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao. Cooperation in Wireless Ad Hoc Networks. In *IEEE INFOCOM 2003*.

[79] C. K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall PTR, 2001.

[80] A. Urpi, M. Bonuccelli, and S. Giordano. Modeling cooperation in mobile ad hoc networks: A formal description of selfishness. In *WiOPT'03*.

[81] A. Wood, J. Stankovic, and S. Son. Jam: A jammed-area mapping service for sensor networks. In *24th IEEE Real-Time Systems Symposium*, 2003.

[82] M. Xiao, N. B. Schroff, and E. K. P. Chong. Utility-based power control in cellular systems. In *IEEE INFOCOM 2001*.

[83] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *ACM Mobihoc*, 2005.

[84] W. Xu, T. Wood, W. Trappe, and Y. Zhang. Channel surfing and spatial retreats: defenses against wireless denial of service. In *Proceedings of the 2004 ACM workshop on Wireless security*, 2004.

[85] H. Yaiche, R. R. Mazumdar, and C. Rosenberg. A game theoretical framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Transactions on Networking*, 18(5), Oct. 2000.

[86] S. Yang and J. S. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *2003 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03)*, Fairfax, VA, USA, October 2003.

[87] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad hoc routing for wireless networks. In *The Second ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'01)*, 2001.

[88] J. Yoon, M. Liu, and B. Noble. Sound Mobility Models. In *ACM MobiCom 2003*.

[89] W. Yu, Z. Ji, and K. J. R. Liu. Securing Cooperative Ad Hoc Networks under Noise and Imperfect Monitoring: Strategies and Game Theoretic Analysis. *submitted to IEEE Transactions on Information Forensics and Security, under review*, 2006.

[90] W. Yu and K. J. R. Liu. Secure Cooperative Mobile Ad Hoc Networks Against Injecting Traffic Attacks. *IEEE SECON 2005*.

[91] W. Yu and K. J. R. Liu. Attack-Resistant Cooperation Stimulation in Autonomous Ad Hoc Networks. *IEEE JSAC special issue on autonomic communication systems*, 23(12):2260–2271, December 2005.

[92] W. Yu and K. J. R. Liu. Game Theoretic Analysis of Cooperation and Security in Autonomous Ad Hoc Networks. *submitted to IEEE Transactions on Mobile Computing, under review*, 2005.

[93] W. Yu, Y. Sun, and K. J. R. Liu. HADOF: Defense against Routing Disruptions in Mobile Ad Hoc Networks. In *IEEE INFOCOM 2005*.

[94] M. G. Zapata. Secure ad hoc on-demand distance vector routing. *ACM Mobile Computing and Communications Review (MC2R)*, 6(3):106–107, July 2002.

[95] M. G. Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *WiSe*, 2002.

[96] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In *ACM MobiCom 2000*.

[97] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *ACM/Kluwer Wireless Networks Journal(ACM WINET)*, 9(5), Sep. 2003.

[98] J. Zhen and S. Srinivas. Preventing replay attacks for secure routing in ad hoc networks. In *Proc. of the Second International Conference on Ad Hoc, Mobile and Wireless Networks*, Montreal, Canada, Oct. 2003.

[99] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. In *IEEE INFOCOM 2003*.

[100] S. Zhong, L. Li, Y. G. Liu, and Y. R. Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks – an integrated approach using game theoretical and cryptographic techniques. In *ACM MobiCom*, pages 117–131, 2005.

[101] L. Zhou and Z. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6):Nov./Dec., 1999.