

Technische Universität Darmstadt

Department of Electrical Engineering and Information Technology
Department of Computer Science (Adjunct Professor)
Multimedia Communications Lab
Prof. Dr.-Ing. Ralf Steinmetz

A Security Framework for Organic Mesh Networks

Diploma Thesis

Submitted by

Kálmán György Graffi

on
12. May 2006

Advisor: Prof. Dr.-Ing. Ralf Steinmetz

Tutor: M.-Tech. Parag Mogre, Dr.-Ing. Matthias Hollick

KOM-D-0249

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 12. Mai 2006 .....

Contents

Ehrenwörtliche Erklärung **iii**

1 Introduction **1**
- 1.1 Motivation . . . 2
- 1.2 Contribution . . . 2
- 1.3 Outline . . . 3

2 The IEEE 802.16 Standard **7**
- 2.1 Specification of the IEEE 802.16 Standard . . . 8
- 2.2 IEEE 802.16 Point-to-MultiPoint Mode . . . 8
- 2.3 IEEE 802.16 Mesh Mode . . . 9
  - 2.3.1 Centralized Scheduling . . . 10
  - 2.3.2 Distributed Scheduling . . . 12

3 Security Terminology and Standard Mechanisms **15**
- 3.1 Protection Goals . . . 16
  - 3.1.1 Data Confidentiality . . . 16
  - 3.1.2 Data Integrity . . . 17
  - 3.1.3 Availability . . . 17
  - 3.1.4 Authenticity . . . 17
  - 3.1.5 Non-Repudiation . . . 18
  - 3.1.6 Privacy . . . 18
- 3.2 Weakness, Vulnerability, Threat, Attack . . . 19
  - 3.2.1 Weakness . . . 19
  - 3.2.2 Vulnerability . . . 19
  - 3.2.3 Threat . . . 20
  - 3.2.4 Attack . . . 20
- 3.3 Adversary Model . . . 20

3.3.1 Defective Nodes 21
3.3.2 Selfish Nodes 21
3.3.3 Malicious Nodes 22
3.4 Mechanisms to Attain the Protection Goals 24
3.4.1 Hashfunctions 24
3.4.2 Symmetric Cryptography 26
3.4.3 Asymmetric Cryptography 27
3.4.4 Additional Mechanisms for Securing Protection Goals 29
3.5 Security in IEEE 802.16 31
3.5.1 Security in IEEE 802.16 Point-to-MultiPoint Mode 31
3.5.2 Security in IEEE 802.16 Mesh Mode 32

4 Routing Protocols for Wireless Mesh Networks 33
4.1 Source Routing - Centralized Deterministic 34
4.1.1 DSR: Dynamic Source Routing 34
4.2 Link State / Distance Vector - Distributed Deterministic 36
4.2.1 AODV: Ad Hoc On-Demand Distance-Vector Routing 37
4.3 Stigmergy based Routing - Distributed Probabilistic 38
4.3.1 Introduction to Stigmergy based Routing Algorithms 39
4.3.2 ABC: Ant Based Control 43
4.3.3 AntNet: Adaptive Agent-based Routing 44
4.3.4 AntNet 1.1 47
4.3.5 ARA: Ant-Colony-Based Routing Algorithm 48
4.3.6 PERA: Probabilistic Emergent Routing Algorithm 50
4.3.7 AntHocNet: Ant-Based Hybrid Routing Algorithm for MANETs 51
4.3.8 Termite 53
4.3.9 Ant-AODV 54
4.4 Comparison and Conclusion 55

5 Security Solutions in Mobile Ad Hoc Networks 57
5.1 Incentive Based Solutions 58
5.1.1 Nuglets 58
5.1.2 Sprite 60
5.2 Overhearing Mechanisms: Watchdog and its Uses 61
5.2.1 Watchdog and Pathrater 61
5.2.2 CONFIDANT: Cooperation of Nodes - Fairness in Dynamic Ad-hoc Networks 65

5.2.3 OCEAN: Observation-based Cooperation Enforcement in Ad hoc Networks . . . 66
5.2.4 CORE: Collaborative Reputation Mechanism . . . 68
5.2.5 Mobile Intrusion Detection System by Zhang/Lee . . . 69
5.2.6 Mobile Intrusion Detection System by Kargl . . . 71
5.2.7 Comparison and Conclusion . . . 74
5.3 Asymmetric Cryptographic Mechanisms . . . 76
5.3.1 ARAN: Authenticated Routing for Ad Hoc Networks . . . 77
5.3.2 SDSR: Secure Dynamic Source Routing . . . 79
5.3.3 SAODV: Secure Ad-Hoc On-Demand Distance Vector Routing . . . 82
5.3.4 Comparison and Conclusion . . . 83
5.4 Symmetric Cryptographic Mechanisms . . . 84
5.4.1 TESLA: Timed Efficient Stream Loss-tolerant Authentication . . . 84
5.4.2 Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks . . . 85
5.4.3 TIK: TESLA with Instant Key Disclosure . . . 86
5.4.4 SRP: Secure Routing Protocol . . . 87
5.4.5 SEAD: Secure Efficient Ad Hoc Distance Vector Routing Protocol 87
5.4.6 Comparison and Conclusion . . . 88

6 A Security Framework for Organic Mesh Networks 89
6.1 Bootstrapping Security in the Network Entry Phase . . . 91
6.1.1 Scanning and Coarse Synchronization to the Network . . . 92
6.1.2 Obtaining Network Parameters . . . 92
6.1.3 Open Sponsor Channel . . . 92
6.1.4 Negotiate Basic Capabilities . . . 93
6.1.5 Perform Network Entry . . . 93
6.2 Neighborhood Establishment . . . 97
6.3 AntSec - Secured AntNet . . . 100
6.3.1 Design Goals and Assumptions . . . 100
6.3.2 AntSec - Routing Messages . . . 102
6.3.3 AntSec - Route Establishment . . . 104
6.3.4 AntSec - Route Maintenance . . . 108
6.3.5 AntSec - Route Error . . . 109
6.3.6 AntSec - Routing of Data Messages . . . 109
6.4 Mechanisms to Detect Misbehavior . . . 110
6.4.1 Watchdog: Overhearing Mechanism for Unencrypted Links . . . 111

6.4.2 WatchAnt: Challenge-Response Mechanism for Encrypted Links 113
6.4.3 Adapting WatchAnt to IEEE 802.16 Mesh Mode 115
6.4.4 The WatchAnt Mechanism 117
6.5 Leak Detection Mechanism 119
6.6 Reputation Management 122
6.6.1 Direct Neighbor: First Hand Reputation 123
6.6.2 Direct Neighbor: Second Hand Reputation 126
6.6.3 Central Reputation Management 128

7 Specific Attack Tree 131
7.1 Analysis of Attacks Committed by Selfish Nodes 132
7.2 Analysis of Attacks Committed by Malicious Nodes 134

8 Simulation Setup and Results 141
8.1 Implementation 142
8.2 Simulation Setup: Scenario A 145
8.3 Simulation Setup: Scenario B 148
8.4 Simulation Setup: Scenario C 150
8.5 Results: Scenario A 152
8.5.1 Throughput 152
8.5.2 Overhead 154
8.5.3 Average Number of Hops for Various Packet Types 157
8.5.4 Detection Quality of AntSec 160
8.5.5 Detailed Analysis of the Throughput of Data Packets over Time . 168
8.5.6 Conclusion of Scenario A 168
8.6 Results: Scenario B 173
8.6.1 Throughput in Scenario B under Varying Drop Rate 173
8.6.2 Detection Quality in Scenario B 174
8.6.3 Reputation Values of the Malicious Node in Scenario B 177
8.6.4 Conclusion of Scenario B 177
8.7 Results: Scenario C 180

9 Conclusions and Future Work 183

List of Figures

2.1 Frame Structure in IEEE 802.16 Mesh Mode 11
2.2 Network Configuration Message (NCFG) 13
2.3 Mesh Distributed Scheduling Message (MSH-DSCH) 14

4.1 Principles of DSR 36
4.2 Principles of AODV 38
4.3 Principles of Ant Routing 41

5.1 Ambiguous Collision Problem 63
5.2 Receiver Collision Problem 64
5.3 Collisions in IEEE 802.16 Mesh Mode due to Optimization 64
5.4 Principles of Iterative Probing 73

6.1 Components of our Security Framework 90
6.2 Observation Radius of a Node when Using WatchAnt 114

8.1 Sample Mesh Network Topology 146
8.2 Topology for Scenario B 148
8.3 Topology for Scenario C 150
8.4 Throughput of Data Packets under Varying Fraction of Malicious Nodes . 153
8.5 Traffic Overhead Measured in Number of Packets 155
8.6 Traffic Overhead Measured in Bytes 156
8.7 Average Number of Hops Taken by Data Packets 157
8.8 Average Number of Hops Taken by Ants 158
8.9 Data Packets Dropped over Time due to Maximum Hop Count Reached . 159
8.10 Characteristic Numbers of Malicious and Benign Nodes in the Network . 160
8.11 Ratio of Malicious Nodes Detected by at least one Neighbor 161
8.12 Ratio of Malicious Nodes Detected by at least one Neighbor over Time (A) 162
8.13 Ratio of Malicious Nodes Detected by at least one Neighbor over Time (B) 162

8.14 Ratio of Nodes Detecting a Malicious Neighboring Node 163
8.15 Ratio of Malicious Nodes Detected by its Neighboring Nodes 164
8.16 Throughput of Data Packets over Time (Part A) 166
8.17 Throughput of Data Packets over Time (Part B) 166
8.18 Fraction of Data Packets Dropped over Time (Part A) 167
8.19 Fraction of Data Packets Dropped over Time (Part B) 167
8.20 Throughput of Data Packets in a Network with 0% Malicious Nodes . . . 170
8.21 Throughput of Data Packets in a Network with 10% Malicious Nodes . . 170
8.22 Throughput of Data Packets in a Network with 20% Malicious Nodes . . 171
8.23 Throughput of Data Packets in a Network with 30% Malicious Nodes . . 171
8.24 Throughput of Data Packets in a Network with 40% Malicious Nodes . . 172
8.25 Throughput of Data Packets in a Network with 50% Malicious Nodes . . 172
8.26 Throughput of Data Packets under Varying Drop Ratio of the Single Malicious Nodes in the Network 173
8.27 Number of Detections in the Network of Scenario B 175
8.28 Ratio of Nodes Detecting a Malicious Neighboring Node in Scenario B . 176
8.29 Ratio of Malicious Nodes Detected by all Neighboring Nodes in Scenario B 176
8.30 Reputation of the Malicious Node in Scenario B (Part A) 178
8.31 Reputation of the Malicious Node in Scenario B (Part B) 179
8.32 Effects of Colluding Malicious Nodes 181

List of Tables

3.1 General Attack Tree Part A: Conserving Resources 22
3.2 General Attack Tree Part B: Delimit the Operability of Nodes or the Network 23
3.3 General Attack Tree Part C: Access to Information in Data Packets 23
3.4 General Attack Tree Part D: Accumulating Information about the Users . 24

4.1 Comparison of the Characteristics of Ant Routing Algorithms 56

7.1 Specific Attack Tree Part A: Selfish Behavior 132
7.2 Specific Attack Tree Part B: Attacks of Malicious Nodes 134

8.1 Parameters Used for Ant and AntSec 144
8.2 Parameters for Scenario A 147
8.3 Parameters for Scenario B 149
8.4 Parameters for Scenario C 151
8.5 Throughput over 1000s in Scenario A 152
8.6 Throughput over last 100s of 1000s in Scenario A 153
8.7 Throughput in Scenario B 174
8.8 Number of Malicious Nodes Detected with Leak Detector in Scenario C . 181

Chapter 1

Introduction

Overview

1.1 Motivation **2**

1.2 Contribution **2**

1.3 Outline **3**

Abstract

In this chapter we introduce the topic of Wireless Mesh Networks. We present our motivation for developing a secure routing algorithm and the contributions to research in this field. Additionally, this chapter gives a brief overview of the organization of the thesis.

1.1 Motivation

In a world where ubiquitous computing is emerging and the anywhere-anytime paradigm gets more and more important, wireless broadband networks seem to be the solution to the continuously increasing demand for a fast and reliable "anywhere" network access. Concurrently, mobile communication providers try to minimize the costs for the maintenance of their wireless networks. Both needs can be fulfilled by *Wireless Mesh Networks* (WMNs), in which parts of the network's infrastructure is maintained by its user-controlled devices.

IEEE 802.16 named "Air Interface for Fixed Broadband Wireless Access Systems" has been standardized in the past few years (2004) to cover the need for broadband wireless network access. It provides a Mesh Mode, in which any participating node in the network must act as a router and forward messages of other nodes in the network. This new concept in the organization of network infrastructure provides some benefits, but also challenging issues which have to be solved. Especially security in routing needs further research.

Routing is essential for the usability of the network and the financial success of its provider, so having subscribers being responsible for this critical task requires thorough investigations of security. Participating nodes have to be authenticated, they shall be enforced to participate in routing and attacks have to be detected. It is important to introduce the necessary security features in this early state of development to avoid costly security patches or even a lack of security.

In this challenging scenario we investigate security mechanisms for routing algorithms to counteract attacks in Wireless Mesh Networks, especially in IEEE 802.16.

1.2 Contribution

Our work is to be seen in the context of deploying Wireless Mesh Networks as anytime-anywhere available infrastructure. We investigate in detail the infrastructure provided by IEEE 802.16 Mesh Mode and review current routing algorithms and security mechanisms for Mobile Ad Hoc Networks (MANETs). MANETs show common characteristics of Mesh Networks and depend on the participation of the nodes in routing, too.

Our main focus in research is on planning and implementing a secure routing algorithm for IEEE 802.16, as routing is crucial for the functioning of the whole network infrastructure. Our contribution to research in this demanding topic can be categorized into the following areas:

- First we investigate the IEEE 802.16 standard [ICS04] and research the requirements and limitations for routing algorithms using the Mesh Mode of IEEE 802.16.
- Second we identify the motivation of attackers in a WMN and develop an attack tree for each type of attacker. By this we get a set of actions that have to be prevented by the applied routing algorithm.
- Third we research the adaptability of both insecure and secure routing algorithms for MANETs to WMNs.
- Fourth we develop a security framework for Wireless Mesh Networks (WMNs), which is optimized for IEEE 802.16 operated in Mesh Mode, but can be easily adapted to other WMN specifications. This framework is our main contribution for securing routing in WMNs, as it provides four components covering all security needs on routing layer.

  Our Framework provides a security bootstrapping feature for new nodes, in which authenticity is deployed. Further we introduce AntSec, a secure routing algorithm based on AntNet [DCD97] and a watchdog mechanism inspired by the work of Marti et al. [MGLB00] to detect misbehavior of neighboring nodes in WMNs with unencrypted links. For networks with encrypted links we present WatchAnt which fulfills the same task as the aforementioned watchdog mechanism. Additionally we develop an extension to AntSec which enables AntSec to detect malicious nodes which are dropping packets and are hiding their misbehavior by colluding.
- Fifth we analyze AntSec, extended by WatchAnt and the leak detection mechanism in an attack tree, considering the most common and feasible attacks done by selfish or malicious nodes.
- Sixth we have implemented our security framework in JiST/SWANS [BHvR05b] and simulated various scenarios defining hostile environments. We evaluate the efficiency of AntSec with and without having the security features enabled in comparison to the Dynamic Source Routing Algorithm (DSR) [JM96] as reference.

1.3 Outline

In this diploma thesis we focus on the development and evaluation of a secure routing algorithm for Wireless Mesh Networks. Therefore we first investigate the mechanisms of the underlying MAC layer, which is in our case IEEE 802.16 operated in Mesh Mode. With this knowledge and a short overview of the terms used in security we investigate

and evaluate various routing algorithms and security mechanisms for MANETs in terms of applicability in IEEE 802.16 Mesh Mode and security considerations. After this we introduce our security framework for routing in IEEE 802.16. We analyze our solution in terms of security using a detailed attack tree. Analysis is completed with simulations done in JiST/SWANS. The thesis ends summarizing the results and giving an outlook on future work.

The organization of the thesis is described in the following:

- In the first chapter we give an introduction to the topic of the research field and show our motivation to develop a secure routing algorithm for Wireless Mesh Networks. Additionally we give an outline of the thesis.

- In the second chapter we present the characteristics of IEEE 802.16 and especially its Mesh Mode. Investigating the limitations and mechanisms provided by the MAC layer is crucial for the design of a routing algorithm which shall operate on the MAC layer of IEEE 802.16.

- In the third chapter we give an overview of the terms used in security context. We introduce common security goals and the terminology used to describe inadequacies of a system's security. Knowing the terminology we look at the motivation of attackers, derive the three types of attackers and identify their attack trees.

- As Wireless Mesh Networks are a young area of research, especially routing algorithms are still rare. Therefore we evaluate routing algorithms for Mobile Ad Hoc Networks in terms of adaptability to IEEE 802.16 Mesh Mode and security in Chapter five. We focus on stigmergy based probabilistic routing algorithms, as the paradigm of distributed probabilistic routing seems most promising for our requirements.

- In Chapter five we present various methods to secure routing and detect misbehavior in wireless MANETs. MANETs and WMNs have several common characteristics. Three main topics are evaluated: incentive-based mechanisms to stimulate the motivation to participate in routing, overhearing mechanisms to detect misbehavior in messages forwarding and routing protocols extended with cryptographic mechanisms to provide different levels of security.

- In Chapter six we introduce our framework for secure routing consisting of four components. First we suggest a solution to bootstrap security, i.e. creating a basis to verify authenticity of nodes in the network. We use AntSec as routing algorithm providing common routing functionality and guaranteeing authenticity and integrity

of the routing messages. To control the forwarding behavior of neighboring nodes we introduce two mechanisms performing the same task; we need two solutions to meet the needs of networks with unencrypted and with encrypted links. To complete our security framework we present a leak detection mechanism. The Leak Detector can detect colluding malicious nodes, which drop data packets, on a route.

- In Chapter seven we present a sophisticated attack tree regarding the security issues for routing in Wireless Mesh Networks to evaluate the quality of our security framework.

- We have implemented and simulated our security framework in JiST/SWANS [BHvR05b]. In Chapter eight we outline the simulation setup and optimization details that had been applied to simulate the behavior of AntSec and DSR [JM96] as reference under varying hostile network environments in detail. Chapter eight also summarizes the results of the simulations, showing that AntSec and WatchAnt effectively counteract the effects of malicious nodes in Wireless Mesh Networks. It is also shown that our leak detection mechanism detects colluding malicious nodes on the route very precisely.

- Finally we summarize in Chapter nine the contributions of our work and completed the thesis by an outlook on areas for future work.

Chapter 2

The IEEE 802.16 Standard

Overview

2.1 Specification of the IEEE 802.16 Standard 8

2.2 IEEE 802.16 Point-to-MultiPoint Mode 8

2.3 IEEE 802.16 Mesh Mode 9

Abstract

In this chapter we look at the IEEE 802.16 standard and especially focus on the Mesh Mode. After giving a brief overview of the functionality of IEEE 802.16 we examine the constraints which result from its design. With this overview we learn the scenario in which IEEE 802.16 Mesh Mode is applied and that secure routing mechanisms are crucial. In this thesis we focus on the Mesh Mode of IEEE 802.16.

2.1 Specification of the IEEE 802.16 Standard

The IEEE 802.16 Standard [ICS04] (IEEE Std 802.16-2004) has been standardized in 2004. Number 802 is the IEEE Standard for Local and Metropolitan Area Networks. Part 16 of this standard framework is the WirelessMAN Standard for Wireless Metropolitan Area Networks. Several amendments have been made to extend the capabilities of the standard. For example, [ICS05b] updates and expands IEEE Std 802.16-2004 to support mobility for the network participants. Whereas [ICS05a] "defines the Wireless MAN Interface Management Information Base (WMANIFMIB) to support the management of Medium Access Control (MAC) layer and Physical layer features as defined in IEEE 802.16-2004".

In the following we want to refer to IEEE Std 802.16-2004 as *IEEE 802.16*. IEEE 802.16 is a standard for Wireless MANs. Wireless networks base on communication over a shared medium. As collisions might occur when two nodes are transmitting at the same time, access to the wireless medium has to be scheduled.

In this chapter we give an overview of IEEE 802.16 and discuss its two modes. IEEE 802.16 can be operated in two different modes, in Point-to-MultiPoint Mode (PMP) and in Mesh Mode (MSH). Both modes require the presence of a Base Station (BS), but in Mesh Mode its usage can be limited to authorization of new nodes in the network. IEEE 802.16 enables a subscription-based network in which every participating node has to be accepted from a central authorization node. Additionally some other restrictions count for the participating devices: they have to be capable to perform asymmetric cryptography and need to have a X.509 certificate issued by its manufacturer (see Section 7.2.1 in [ICS04]). This certificate is used to authenticate the node to the authorization server. In the next sections we introduce the two modes of IEEE 802.16.

2.2 IEEE 802.16 Point-to-MultiPoint Mode

The main area of application of IEEE 802.16 is the PMP Mode. In this mode two kinds of nodes exist: The Base Station (BS) and Subscriber Stations (SS). As in GSM or similar communication networks, new nodes (i.e. SS) can only communicate with the BS directly. The BS acts as a central station, which is used as relay node both for communication to distant and neighboring nodes.

As the BS has a central role, medium access is separated in IEEE 802.16 into downlink (from the BS to the SS) and uplink (from the SS to the BS). This can be realized in IEEE 802.16 both with Time Division Duplex (TDD) and Frequency Division Duplex (FDD). In this either the time or the frequencies are slotted. Time is slotted to provide

for the downlink and uplink disjunctive sending time. Using FDD frequencies are divided into two disjunctive intervals so that uplink and downlink communication can take place simultaneously.

With TDD uplink and downlink are separated by having sending intervals. The uplink channel itself is managed using TDMA and DAMA and so divided into a number of time slots. These timeslots are assigned for various uses such as registration, contention or user traffic). It is important to notice that the BS defines the usage of these time slots. For the downlink channel TDM is used. In this the information for all SS is multiplexed and sent as a single stream to all SS in the transmission range.

PMP is a centralized communication system in which all communication is relayed by the BS, even if the communication partner is in the direct neighborhood of the communication node. As we show in 3.5.1 the BS is providing all relevant security information, too. In PMP routing is done by sending all data to the BS. And the BS as trusted node forwards the data to the BS in which's sector the destined node is. Due to the similarity of the PMP Mode to GSM or 802.11 in infrastructure mode, security solutions might be adaptable to IEEE 802.16 PMP Mode.

2.3 IEEE 802.16 Mesh Mode

A mesh is a network topology in which the participating devices are connected using numerous redundant interconnections. This network topology provides several benefits in comparison to base station / subscriber station models. In this chapter we present these benefits and discuss how the Mesh Mode of IEEE 802.16 is defined.

Mesh networks provide a greater redundancy than BS / SS model networks. Having several additional paths enables smart traffic balancing mechanisms. In cases of heavy traffic load, alternative paths can be used. Due to the increased interconnections the locality of information is more important. It is more efficient to use information, which has been retrieved locally, at the point of its retrieval.

In wireless networks line of sight (LOS) is often required to establish a long-range connection. Mesh networks cope with this requirement by providing the possibility to forward messages on multi-hop paths circumventing obstacles. Subscriber stations having no LOS to the base station can connect using multi-hop paths. This connectivity challenge is solved easily by mesh networks. As each device in the mesh network acts as a router and forwards messages, routing is crucial for the functionality of the network.

In this section we discuss the characteristics and specifications of the IEEE 802.16 Mesh Mode. The Mesh Mode of IEEE 802.16 is introduced as "Optional Mesh topology support" (see Section 6.3.6.6 in [ICS04]). Through the specifications of the standard the

Mesh Mode is only seen as addition to the PMP Mode. As a result several mechanisms are only adapted from PMP mode and were not redesigned.

The Mesh Mode does not provide clearly separated downlink and uplink channels / subframes in contrast to the PMP Mode. The separation cannot be done as every node in the mesh network acts as transmitting and relaying station. To establish various interconnections to neighboring nodes each node is able to create direct communication links to other subscriber stations in the network. In PMP Mode communication links could only be established between SS and BS, not between two SS nodes. To control access to the wireless medium scheduling is needed.

There are two scheduling mechanisms specified in the standard: Centralized Scheduling (see Section 6.3.6.6.2 in [ICS04]) and Distributed Scheduling (see Section 6.3.6.6.1 in [ICS04]). Both scheduling mechanisms require time division duplex (TDD). This means that IEEE 802.16 Mesh Mode can only be operated using TDD. In TDD time is broken down into frames. Figure 2.1 presents the frame structure and the different control messages that can be sent in the control subframe.

A Mesh frame consists of a control and a data subframe. Access to both subframes has to be scheduled among the neighboring nodes. Transmissions in the data subframe are directed, this means that a minislot in the data subframe can only be reserved for a link between two nodes. Broadcasting is not possible. Scheduling of the access to the data subframe is performed in the control subframe. The access to the control subframe is scheduled collision-free using an election algorithm, which determines in which frame the current node is allowed to transmit. The scheduling mechanism ensures collision freedom, i.e. no other node in the two-hop-neighborhood interferes. The election algorithm is described in Section 6.3.7.5.5.6 in [ICS04].

2.3.1 Centralized Scheduling

Using the Centralized Scheduling Mechanism is characterized by central scheduling decisions of the Mesh Base Station: The transmissions between the SS nodes, as well as the distribution of the Centralized Scheduling information, are determined by the BS. Subscriber Stations can request additional transmission slots in the data subframe by sending resource requests during the control portion of the control subframe, which was assigned by the BS to them.

A SS can only request a directed link to a single neighboring node. So broadcasting data messages using Centralized Scheduling is not supported in the specification of IEEE 802.16. Virtually a tree is build with the Mesh BS as root and each node in this tree has to accumulate the resource requests of its subordinated neighbors when sending a resource

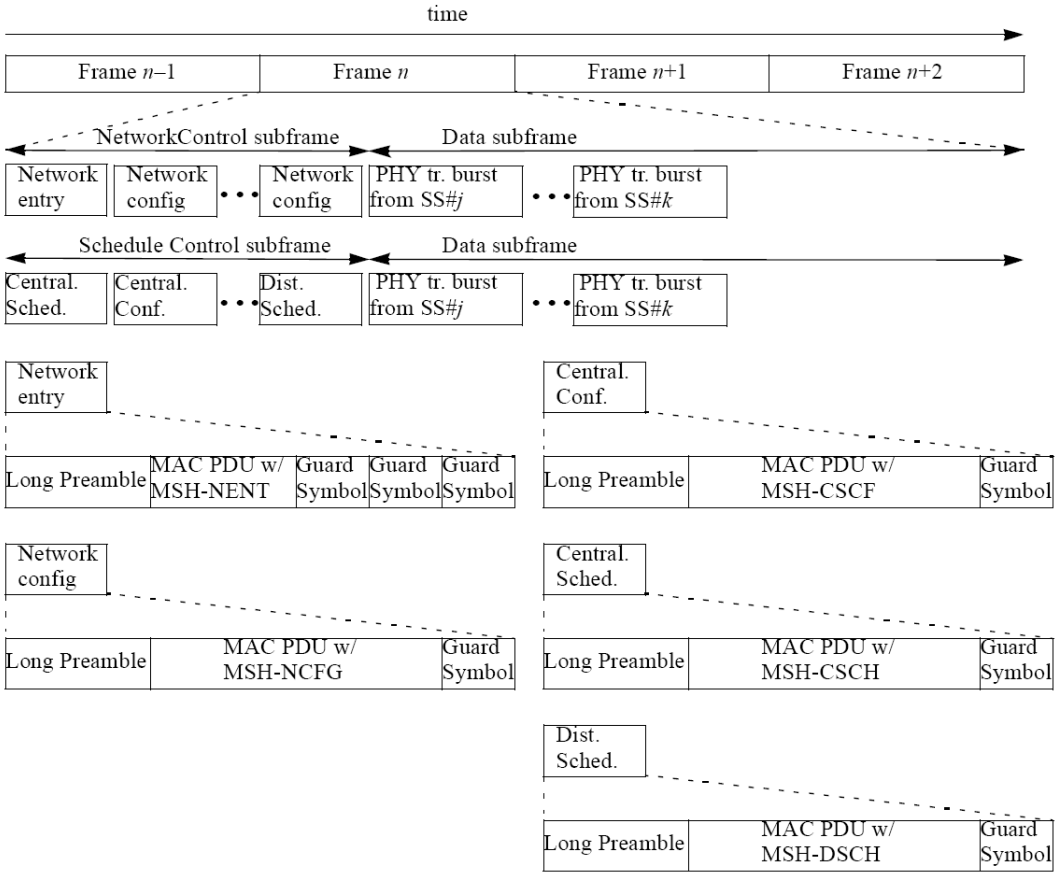


Figure 2.1: Frame Structure in IEEE 802.16 Mesh Mode. Time is slotted in frames. Each frame contains a control subframe and a data subframe. The figure shows the various messages that can be sent in the control subframe. Scheduling of the minislots in the data subframe is done using MSH-DSCH or MSH-CSCH messages. Source: 8.3.5.3 in [ICS04]

request to its neighbor which is its father in the virtual tree.

In contrast to the data subframe the control subframe is only utilized by one node in a two-hop-neighborhood at a time. This guarantees collision-free communication. This scheduling algorithm show similar characteristics like the communication principles in PMP, only the communication of the SS to BS is routed across the mesh network. This centralized principle results in disadvantages for the maintenance a mesh network. First of all the BS needs to be contacted periodically by a huge number of nodes in the network. The BS is both bottleneck for the performance of the network as well as single point of failure.

2.3.2 Distributed Scheduling

In the Distributed Scheduling mechanism the access to the control subframe is collision-free. The control subframe is used to negotiate connections on the data link layer to neighboring nodes within the two-hop-neighborhood. As the access to the control subframe has to be collision-free: no node in the two-hop-neighborhood of the transmitting node is allowed to transmit as well.

Mesh Network Config (MSH-NCFG) messages are sent periodically in the control subframe. Figure 2.2 shows the structure of the MSH-NCFG. MSH-NCFG messages are used to communicate information on current neighboring nodes and information relevant for the physical layer to maintain the functionality of the network.

Additional to NCFG messages each node broadcasts Mesh Distributed Scheduler messages (MSH-DSCH) (see Figure 2.3) in the control subframe. MSH-DSCH messages are used to establish a connection using a 3-way-handshake. The process includes a request for a connection offering a set of data minislots which are available for reception or transmission. The destined neighbor cross-checks these availabilities with its own availabilities and replies with a grant for matching time slots. This grant is returned by the previously requesting node as grant confirm. Every node receiving a grant message or a grant confirm message marks the corresponding minislots as not available anymore.

For the following description see Figure 2.3. The MSH-DSCH message contains first flags signaling whether a request or grant is transmitted in the current MSH-DSCH message. Then the number of availability, request and grant information entries is presented. After this the corresponding information entries follow.

The Distributed Scheduler does not provide any broadcasting capability, except of using the control subframe. Additionally, we have to note that a single minislot (transmission slot in the data subframe) can be in 4 states: completely available, available for receiving, available for transmission and unavailable. By this classification the MAC layer

Syntax	Size	Notes
MSH-NCFG_Message_Format() {		
Management Message Type = 39	8 bits	
NumNbrEntries	5 bits	
NumBSEntries	2 bits	
Embedded Packet Flag	1 bit	0 = Not present, 1= present
Xmt Power	4 bits	
Xmt Antenna	3 bits	
NetEntry MAC Address Flag	1 bit	0= Not present, 1= present
Network base channel	4 bits	
reserved	4 bits	Shall be set to zero
NetConfig Count	4 bits	
Timestamp Frame Number Network Control Slot Number in frame Synchronization Hop Count	 12 bits 4 bits 8 bits	 See 8.2.3.2
NetConfig schedule info Next Xmt Mx Xmt Holdoff Exponent	 3 bits 5 bits	
if (**NetEntry MAC Address Flag**) **NetEntry MAC Address**	 48 bits	
for (i=0; i< NumBSEntries; ++i) {		
BS Node ID	16 bits	
Number of hops	3 bits	
Xmt energy/bit	5 bits	
}		
for (i=0; i< NumNbrEntries; ++i) {		
Nbr Node ID	16 bits	
MSH-Nbr_Physical_IE()	16 bits	See Table 66
if (**Logical Link Info Present Flag**) **MSH-Nbr_Logical_IE()**	16 bits	See Table 66 See Table 67
}		
if (**Embedded Packet Flag**) **MSH-NCFG_embedded_data()**	*variable*	 See Table 68
}		

Figure 2.2: Network Configuration Message in IEEE 802.16, Source: 6.3.2.3.35 in [ICS04]

Syntax	Size	Notes
MSH-DSCH_Message_Format() {		
Management Message Type =41	8 bits	
Coordination Flag	1 bit	
Grant/Request Flag	1 bit	
Sequence counter	6 bits	
No. Requests	4 bits	
No. Availabilities	4 bits	
No. Grants	6 bits	
reserved	2 bits	Shall be set to zero.
if (Coordination Flag == 0)		
MSH-DSCH_Scheduling_IE()	*variable*	
for (i=0; i< No_Requests; ++i)		
MSH-DSCH_Request_IE()	16 bits	
for (i=0; i< No_Availabilities; ++i)		
MSH-DSCH_Availability_IE()	32 bits	
for (i=0; i< No_Grants; ++i)		
MSH-DSCH_Grant_IE()	40 bits	
}		

Figure 2.3: Mesh Distributed Scheduler Message in IEEE 802.16, source: 6.3.2.3.37 in [ICS04]

can be utilized more efficiently as collisions in the data subframe may occur, only nodes for which the information is relevant must not be affected. An investigation of the performance characteristics of the Distributed Scheduling algorithm is given in [CMZ+05].

Chapter 3

Security Terminology and Standard Mechanisms

Overview

3.1 Protection Goals 16
3.2 Weakness, Vulnerability, Threat, Attack 19
3.3 Adversary Model 20
3.4 Mechanisms to Attain the Protection Goals 24
3.5 Security in IEEE 802.16 31

Abstract

In this chapter we give an introduction to the terms we use. First we look at the common protection goals denoted in security and the terms used to describe security leaks. After this we research the motivation of misbehaving nodes in Wireless Mesh Networks and break down into detail the attacks that result from these motivations. After this investigation of attacks we introduce security mechanisms that can apply to prevent misbehaving nodes from threatening the protection goals. At the end of the chapter we look at the security mechanisms provided by IEEE 802.16.

In our scenario we assume that the mesh network consists of many nodes, nodes having various interests. Communicating in this network with other nodes, which are not in our direct neighborhood, bears danger because it is necessary to expose the transmitted data to intermediate nodes on the route. The interests of these intermediate nodes are unknown and may be contrary to the interests of the source and the destination node.

Security is an essential component in a routing algorithm. A secure protocol shall fulfill a specific task, even when some participants in the network follow other interests. Whereas the term *safety* means the proper functionality of an algorithm or protocol under normal terms [Eck04], *security* means the proper functionality of the algorithm under attack or non-compliant or even malicious behavior of the participants. It is essential to implement security mechanisms in early stages, in order that the operation mode of the algorithm is predictable under any circumstances.

In this chapter we give a brief introduction to the terms used in the security context. It is important to know what security is, so we look at the common protection goals in security. We follow the terminology introduced in RFC 2828 [IETF00]. We also give a look at the terminology of weaknesses and attacks, so that we can determine an attacker classification for our scenario afterwards. This chapter ends with the presentation of common mechanisms that are used to guarantee protection goals. Knowing the tools we have at hand, we better understand the principles of the security suites designed to secure routing operability against various attacks.

3.1 Protection Goals

In the following section we give a short overview of the protection goals we want our protocol to guarantee. Protection goals are essential properties of a system, which have to be safeguarded under any circumstances. Especially effects of mischievous behavior have to be avoided. In 3.4 we survey mechanisms known to attain these protection goals.

3.1.1 Data Confidentiality

A system provides *Data Confidentiality* if it is not possible for an unauthorized subject to retrieve any information from the system, which is not allowed to be retrieved by this subject. The access to the disclosed data or information is only permitted to authorized subjects and no one else subject is able to get access to it. In [IETF00] *Data Confidentiality* is defined as follows:

> *"The property that information is not made available or disclosed to unauthorized individuals, entities, or processes [i.e., to any unauthorized system*

entity]."

As an example for *Data Confidentiality* we denote that in some financial scenario, critical data has to be sent to the communication partner over a multi-hop route. Only the destination of the message containing the secret data is permitted to disclose the critical data, all intermediate nodes on the route must not retrieve the secret data. In this case the *Data Confidentiality* of critical data has to be guaranteed.

3.1.2 Data Integrity

In [IETF00] *Data Integrity* is defined as

> *"The property that information has not been modified or destroyed in an unauthorized manner."*

So a system provides *Data Integrity* (shorter: integrity) if an unauthorized manipulation of the data or information is either impossible or detectable. *Integrity* is essential in order to fulfill other protection goals in a communication process, because it is necessary to be certain, that the messages have not been altered on the route.

To pick up the example from above, the critical data may be a bank order. The content of the bank order, e.g. the amount of money to transfer, must not be altered on its way to the communication partner. *Integrity* of the message means that the message containing the bank order is not modified invalidly during transmission.

3.1.3 Availability

The term *Availability* describes the attribute of a system, that this system provides a specific service under any circumstances. The service it has to provide cannot be disabled by an unauthorized subject. A typical example is a scenario with a central server being responsible for distributing critical data to a group of clients. If this central server is attacked by sending it more packets than it can process, congestion is the result and the server cannot communicate. The service of distributing the critical data is no longer provided by the central server, *Availability* has been successfully attacked.

3.1.4 Authenticity

The *Authenticity* of a subject is ascertained if the identity of the subject is proved by checking its characteristic, unique properties. Or as [IETF00] defines:

> *"The property of being genuine and able to be verified and be trusted."*

The term *Authentication* is used to describe the verification of the identity of a subject on the basis of specific properties. *Authenticity* is essential for differentiating the behavior for various subjects. When specific rights are granted to a subject corresponding to its role and function, this procedure is called *Authorization*.

In our scenario it is essential to know who the communication partner of a node really is to avoid sending messages to an unknown node. Furthermore it is essential to cooperate only with nodes which are authorized to participate in the network. With this mechanism malicious and banned nodes are kept out of the network and it is avoided that these nodes use service without paying for it.

3.1.5 Non-Repudiation

Non-Repudiation means that no subject is able to deny an action it has done. Each subject has to admit its responsibility for deeds it has done. [IETF00] defines *Repudiation* as

> *"Denial by a system entity that was involved in an association (especially an association that transfers information) of having participated in the relationship."*

In our scenario we have to determine which node was responsible for an attack, therefore mechanisms have to be used to identify the initiator of the attack or malfunction. *Non-Repudiation* requires *Authenticity* to be ensured.

3.1.6 Privacy

The identity of devices is often linked to the identity of users. Any trace a node disseminates reveals information about its user. Often it is not wanted by the user to allow others to journalize its behavior, its movements and interests. Privacy means to provide only minimal information about the user so that no other institution is able to log the behavior patterns of other participants in the network.

Therefore for the design of security mechanisms the need-to-know principle has to be used, which enables to fulfill the task of a mechanism but also avoids giving too much detail about its user. *Privacy* is defined in [IETF00] as:

> *"The right of an entity (normally a person), acting in its own behalf, to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share information about itself with others."*

In our scenario, especially for routing, *Privacy* is prioritized lower than the other protection goals, because it is most important to guarantee the availability of the network. Authenticity and non-repudiation is needed to be able to exclude malicious nodes. Whereas integrity is essential for a routing protocol, confidentiality and privacy are less important for the operability of the network.

3.2 Weakness, Vulnerability, Threat, Attack

As we have seen a system or protocol can fulfill several protection goals. To describe the state of systems not fulfilling these protection goals or offering only partial implementation of security, we introduce the terminology to describe flaws and inadequacies in providing security. We give an overview of the terms weakness, vulnerability, threat and attack.

3.2.1 Weakness

A *weakness* describes a flaw of the system which can be used to attack the system or to effect a behavior which is not conform to the design of the system. The fact that any node in the mesh network must act as a router is a *weakness*, because this state enables malicious nodes to have effect on routing and therefore on the performance of the network.

3.2.2 Vulnerability

A *vulnerability* is a weakness which circumvents the security mechanisms of the system. An exploit of this weakness is not noticed or blocked by the system's security mechanisms. [IETF00] defines *vulnerability* as follows:

> *"A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy."*

So the difference between a weakness and a *vulnerability* is that the exploit of the *vulnerability* could violate the systems security policy, whereas the exploit of a weakness is not relevant for the functionality of the system.

A *vulnerability* would be to accept any message and process it without checking whether this packet has been sent legitimately. Processing of a single packet is not endangering, but processing of a great amount of illegitimate packets can only be realized as an attack when the node considers the history of packets received, but this often is not the case, so the mischievous stressing of the computational power of the node is not detected.

3.2.3 Threat

A *threat* is the combination of one or more exploits of vulnerabilities or weaknesses to attack one of the protection goals, namely confidentiality, integrity, authentication or availability. The combination of these flaws results in a potential to breach protection goals as defined in [IETF00]:

> *"A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm."*

An attacker follows in this case a specific intention, which he wants to accomplish by exploiting one or more weaknesses. Undetected modification of routing packets would be a *threat* to the functionality of the network. By these invalid modifications the routes may be altered and the availability of the network might be endangered.

3.2.4 Attack

An *attack* describes the unauthorized access to a system in order to exploit a threat, i.e. to tap the potential of a threat and cause harm. In [IETF00] an *attack* is described as

> *"An assault on system security that derives from an intelligent threat, i.e., an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system."*

We differentiate between active and passive *attacks*. While a *passive attack* is aimed to breach confidentiality by unauthorized information retrieval, *active attack* means that the attacker has to modify data, knowing that his actions are contrary to the specification of the system, to achieve his goal. By active attacks the integrity or availability of the system is endangered.

3.3 Adversary Model

In order to design a secure routing protocol we need to know the intentions of misbehaving nodes. Knowing the intentions we can develop an attack strategy adapted to the properties of the identified adversary models. An adversary is defined in [IETF00]: *"An entity that attacks, or is a threat to, a system"*. In this section we have a look at the categories of misbehaving nodes in the network. Considering their intentions we give an abstract view at the threats on routing in Wireless Mesh Networks. A specific and detailed attack tree

has to be adapted to the routing algorithm which is used, so that the specific flaws are considered. This is done in Chapter 7.

Before we give look at the adversary models in detail, we describe how the attack trees have to be interpreted. We divided the set of attacks into sections, so that we can assign to each type of attacker a single attack tree based on his motivation. The labels of these sections are entitled with big roman letters and a description of the misbehavior. Each of these sections contains a list of actions that can be taken to implement the specific misbehavior. Any misbehaving effect can be produced by more than one action. To describe which alternatives exist and which subordinated tasks must be taken to fulfill such an action we introduce the Boolean operators OR and AND.

An action, which requires subordinated tasks to be fulfilled, in order to have effect is tagged with AND. Let X.Y be the number of the superior action marked with AND, then any action identifier beginning with X.Y. is a sub-task for X.Y. All subordinated actions X.Y.i must be fulfilled for X.Y to be achieved. Tasks which can be accomplished using various methods are tagged with OR. If any of these various methods come true then the superior task is achieved.

3.3.1 Defective Nodes

The trivial case for unintended misbehavior is covered by this category. The nodes in this category behave compliant to their protocol instructions; they act in good faith, but their implementation of the protocol is not compliant to the protocol used in the network. This may have various reasons, for example outdated firmware, bugs in the code, errors caused by physical impact on the device, by electrical shocks or contact to water. Or other reasons which may lead to differences in the protocols followed by the defective node and the other nodes in the network.

Due to this category enfolding all none intentionally misbehaving nodes, we do not analyze the threats caused by this group in detail. Any action not taken intentionally by a defective node can also be taken by a selfish or malicious node intentionally. The effects of defective nodes are covered by both categories following and need no further examination.

3.3.2 Selfish Nodes

In this category we subsume all those nodes which are not willing to contribute any resources to the network, for which they do not receive an advantage in return. In our scenario some devices are mobile; they only have a limited contingent of energy. Energy is a precious resource, since transmission of data and computational calculations decrease

the energy level in a device. For a selfish node it is worthwhile to refuse to provide service for other nodes in order to conserve energy. This service for other nodes includes the forwarding of packets, participating on their routes and performing calculations, from which only other nodes take advantage.

In Table 3.1 we give an general attack tree, which describes the actions that can be taken by a selfish node to conserve resources.

We identified energy as the main resource. Other resources like computation power or memory capacity can be attributed to the consumption of energy. Transmissions serving other nodes, for example to forwarding their messages, require spending energy for the transmission itself and to postpone the transmission of own packets. This is not in the selfish node's interest, so dropping of packets, which have to be forwarded, is a feasible way for the selfish node to conserve energy. (A.1.1, A.2.1)
A better way to conserve energy is to avoid being part of the route between two other nodes, therefore the selfish node has to take appropriate action, so that any route, which contains the selfish node, breaks or will not be created (A.1). To give more details on the actions that can be taken, we have to look at the functionality of the routing protocol. In Chapter 7 we give a detailed analysis of the possible strategies of the selfish node and show, that they do not harm the system when using AntSec and the other components of our framework.

Tree A: Conserving resources
OR 1. No participation in routing and data forwarding
1.1 Dropping of routing messages
1.2 Modification of routing messages
1.3 Generating false routing messages
OR 2. No forwarding of data packets
2.1 Dropping of data packets

Table 3.1: General Attack Tree Part A: Conserving Resources

3.3.3 Malicious Nodes

A malicious node aims to cause damage to other nodes or the network itself. There are various methods to harm other nodes, like degrading the quality and performance of the network participation of the specific other node by destroying packets from or to the other node (B.1). Or the malicious node could exhaust the resources of a victim node (B.2). Not only other nodes can be attacked, even the performance of the whole network can be effected by exploiting vulnerabilities in the routing protocol (B.3). Another target for

Tree B: Delimit the operability of nodes or the network
OR 1. Destroy some/all data packets
OR 1.1 Attack on the physical layer
1.1.1 Send permanently junk messages (air jamming)
OR 1.1.2 Send only when other nodes send
1.1.2.1 Listen and send on signal
1.1.2.2 Follow the schedule of the timeslots to effect collision
OR 1.2 Attack on the MAC layer
1.2.1 Reserve all available bandwidth permanently
1.2.2 Send data to destroy the control messages
OR 2. Overloading of a component (bandwidth, CPU, energy)
2.1 Overload direct neighbors
2.2 Overload some distant node
2.3 Overload the network
OR 3. Sabotage the routing operability
3.1 Forge routing messages
3.2 Drop valid routing messages
3.3 Modify valid routing messages

Table 3.2: General Attack Tree Part B: Delimit the Operability of Nodes or the Network

Tree C: Access to information in data packets
OR 1. Log and analyze data packets that pass
OR 2. Effect to participate in many routes
2.1 Choose a central physical location
2.2 Sabotage routing to appear as favorable next hop
AND 2.3 Wormhole attack
2.3.1 Place a cooperative node near the destination node
2.3.2 Build up tunnel to cooperative node
2.3.3 Use this tunnel for routing / forwarding to destination node

Table 3.3: General Attack Tree Part C: Access to Information in Data Packets

malicious nodes are the users of the nodes. The retrieval of significant information about the user (D) or the content of his communication (C) may be in the interest of the malicious node. We see that in opposite to the selfish node, there are cases where the malicious node intentionally wants to break the possibility of communication in the network (A), whereas the selfish node prefers to keep the performance in the network on a high level to receive better service.

Tree D: Accumulating information about the users in the network
AND 1. Log the movements behavior of other nodes
OR 1.1 Track the position of the node
1.1.1 Use information about the radio signal
1.1.2 Use routing information
1.2 Determine the corresponding user to the Node-ID
OR 2. Determine the identity of the users
2.1 Analyze the passing data packets
2.2 Retrieve the information during net entry of the node
3. Log the patterns of utilization of other nodes

Table 3.4: General Attack Tree Part D: Accumulating Information about the Users

3.4 Mechanisms to Attain the Protection Goals

In this section we introduce standard mechanisms that are used to achieve the protection goals. We also give an overview of the efficiency and the prerequisites of the introduced mechanism and evaluate their advantages and disadvantages.

3.4.1 Hashfunctions

With the term *hash function* we denote one-way functions. For *hash functions* counts that to any element in the co-domain the corresponding element in the domain of the function is nearly impossible to determine. The size of the function value is fixed (for example 20 bits), whereas that size of the input value is arbitrary. A survey of cryptographic hash functions is given in [BSNP95].

The mathematical definition of hash functions is:

- Let $B = \{0, 1\}$, $n \in \mathbb{N}$ fixed and $b \in B^*$
- hash: $B^* \longmapsto B^n$
- One way function: Let X = hash(b), then for given X one cannot calculate c so that X = hash(c) with feasible effort.

Note that any finite set can be mapped to B^*. We give an example: Let f be a *hash function* mapping from the co-domain X into the domain f(X). Let x be an element in X and y = f(x) the function value of x. Then the property of f being a *hash function* would mean, that by knowing y it is nearly impossible to determine x. But by knowing x, we can easily determine y. It is nearly but not completely impossible to determine x from f(x) because the possibility exists to calculate the function values for each element in the

domain and check whether the corresponding function value is y or not. By choosing a sufficiently large domain for the hash function, we can say that it is not feasible to put this calculation into execution. We give a short impression of the application area of hash functions:

3.4.1.1 Map large amounts of data to short representation

The size of the hash value is fixed, the input value may be of any finite size. This means that any input is mapped to a (non-unique) representation with a fixed size. As every input value results in exactly one function value and as no (other) input value can be found with the same function value, we say that the function value can be used represent the input value.

In order to identify some data it is sufficient to identify its hash value. This mechanism is often used in cryptography as it is easier to handle shorter fixed size data than large data of arbitrary size.

3.4.1.2 Only the generator of a hash value knows the corresponding input value

Since it is not feasible to calculate the preimage of a hash value, only the generator of the hash value is capable of knowing the corresponding input value. This can be used to authenticate once the creator of the hash value y by disclosing the argument value x whose corresponding hash value is y.

3.4.1.3 Hashchains can be used to guarantee authenticity more than once

A *hashchain* is the composition chain of the hash function applied to an input value [IETF00]. This means that the hash function is applied to the function value of the last input, so the hash function is applied to the input value x times (for example 365): $hash^{365}(x)$. By publicizing the last function value ($hash^{365}(x)$) the originator of the *hashchain* can prove his identity 365 times.

For example the issuer of a certificate wants to keep control of the certificate and therefore he implements a revocation system based on *hashchains*. The mechanism requires that the validity of the certificate has to be reaffirmed every day of the year. Therefore the issuer publishes the certificate which contains $h^{365}(x)$ once, stores x securely and confidential and publicizes on each day i the current verificator $h_{365-i} = h^{365-i}(x)$. Any subject can check whether the value h_{365-i} for day i has been released and, if it was released, whether it is valid, i.e. $h^i(h_{365-i}) = h^{i+365-i}(x) = h^{365}(x)$. The *hashchain* is used to prove that the originator of h_{365-i} is the originator of $h_{365-i+1}$. Only the orig-

inator of h_{365} has to be authenticated using a different authentication mechanism (for example a digital signature, see Subsection 3.4.3).

3.4.2 Symmetric Cryptography

Symmetric cryptography is historically the oldest known mechanism of cryptography [Sim00], it is commonly known as secret key cryptography. Security is based on the nondisclosure of a common secret key. Every participating entity (in most cases only two) knowing this secret key is able to encrypt, decrypt and verify messages encrypted or signed with this key. The essential characteristic of symmetric cryptography is the usage of the same secret key for both operations: for encryption and decryption.

The problem which is caused using symmetric cryptography is known as the *key-exchange problem*. We look at a scenario with n entities, which want to communicate with each other using secret key cryptography. Then $\frac{n \cdot (n-1)}{2}$ keys a necessary and this amount of keys has to be exchanged.

Symmetric cryptography is not well suited for large scale scenarios in which every entity exchanges a secret key with every other entity. But the advantage of secret key cryptography is its efficiency. The operations provided need far less computational power than operations performing asymmetric cryptography (see Subsection 3.4.3).

3.4.2.1 Encryption and Decryption

Encryption using secret key cryptography is done by deriving an encryption key from the secret key and using this encryption key on the plain text that is to be encrypted. Various symmetric encryption algorithms can be found in literature [Eck04]. Common encryption algorithms are Data Encryption Standard (DES) [Ame81] and Advanced Encryption Standard (AES) [DR02].

For the decryption of cipher text a decryption key has to be derived from the secret key which was used for encryption. Confidentiality of encrypted messages is based on the confidentiality of the secret key.

3.4.2.2 Keyed-Hash Message Authentication Code

Keyed-Hash Message Authentication Code (HMAC) [BCK96] [IETF97] is a mechanism based on keyed hashing [BL73] to authenticate messages. We describe its functionality in the following: Consider a scenario of two nodes communicating in an insecure network. They want to guarantee integrity of the messages sent to each other and have therefore previously exchanged a secret key. The two nodes can calculate HMACs for the mes-

sages, which can only be verified having the secret key, which was used for its generation. Similar to the generation of the MAC (Message Authentication Code) a hash of the message is calculated. But in HMAC a secret key is involved in the generation of the hash. For further details how this is done see [BCK96].

The mathematical definition is:

- Let B = $B = \{0, 1\}$, $n \in \mathbb{N}$, $m \in \mathbb{N}$ fixed, $b \in B^*$ and $key \in B^m$ a secret key
- HMAC : $(B^*, B^m) \longmapsto B^n$
- One way function: Let X = hash(b,key), then for given X and b one cannot calculate $key2$ so that X = hash(b,$key2$) with feasible effort.

Thus the HMAC contains information derived from a secret key. Only an entity possessing this secret key can calculate the same HMAC for the same message. Using another key for the HMAC generation of the same message results in different HMAC. If an attacker modifies the message or appends an HMAC of his own, the receiver of the message can detect the illegal HMAC. The HMAC for a received message is generated by the receiver using the secret key, which the receiver shares with the sender of the message. If the HMAC generated by the receiver differs from the HMAC generated by the sender, the message is corrupt. For a successful attack on the integrity of the message the attacker has to know the secret key, which the source and the destination entities are using. HMACs are efficient to generate and can only be verified by entities possessing the corresponding secret key.

3.4.3 Asymmetric Cryptography

The concept of asymmetric cryptography was first introduced by Whitfield Diffie and Martin Hellman [DH76] and Ralph Merkle [Mer78]. In contrast to symmetric cryptography, asymmetric cryptography uses two different keys per participating entity. Each entity has a Private Key and a Public Key, the latter is public and can be shared. Separating the keys needed for encryption and decryption and also for signing and verifying leads to the advantage of solving the key exchange problem.

The security of asymmetric cryptography is based on the difficulty of mathematical problems in number theory (factorization of integers, discrete logarithms). No efficient algorithm for the solving of these problems exists until now, so they are supposed to be hard to solve. That means that finding the correct result demands excessive computational power. Calculations to solve this problems without any hint (given by the Private Key) would last more than several years. But no proof exist that these problems are really hard

to solve. In [Sho99] it is shown that quantum computers can solve any common mathematical problem (large number factorization, discrete logarithm) used for asymmetric cryptography in tolerable time.

One of the most common asymmetric cryptography standard is RSA [RSA77] named after its inventors Ronald L. Rivest, Adi Shamir and Leonard Adleman. Its security is based on the problem of large number factorization. Finding an algorithm to solve the problem of integer factorization makes RSA insecure. Finding an algorithm for breaking the security of RSA, providing functionality for deriving the RSA Private Key from the corresponding Public Key, would not automatically result in an efficient algorithm for integer factorization.

The Rabin cryptography system [Rab79] is similar to RSA, but it is mathematically proved to be as hard to solve as the integer factorization problem. ElGamal introduced an asymmetric cryptosystem [ElG85] based on the difficulty to determine the discrete logarithm of elements in finite-dimensional fields.

In asymmetric cryptosystems each entity has a public and a Private Key. The Public Key can be derived from the Private Key but not vice versa. The solution itself or a "shortcut" to the mathematical problem is represented in the Private Key. Using the Public Key for encryption generates a mathematical problem which can only be solved practically with the knowledge of the Private Key.

Signing a message means to encrypt the hash of the message using the Private Key, generating a so-called signature of the message. Decrypting this signature with the Public Key of the signer results in the original hash of the signed message. By comparing the hash resulting from decrypting the signature to the hash of the message produced directly shows whether the signature is valid for this message or not. Each entity can verify the authenticity of the message this way.

Every entity is allowed to encrypt messages for other entities and to verify their signatures. For these operations the Public Key is used. In contrast to encryption and signature verification, only the corresponding entity is allowed to decrypt messages encrypted with its Public Key. And only the owner of the key pair is allowed to generate signatures verifying the authenticity of the signed messages. These restrictions are guaranteed due to the inability to derive the Private Key from the Public Key, but still it is necessary to keep the Private Keys undisclosed.

So using asymmetric cryptography enables each entity to encrypt messages for other entities and to verify signatures of other entities. The Private Key has to be kept secure to avoid attacks on confidentiality and integrity. The corresponding Public Keys have to be made public and accessible. Various efforts of realizing such a Public Key infrastructure (PKI) exist in literature. Further details on PKIs can be found in [Eck04].

3.4.4 Additional Mechanisms for Securing Protection Goals

In this section we give a short overview of additional mechanisms providing functionality for securing protection goals.

3.4.4.1 Timestamp and Sequence Numbers

Timestamps or sequence numbers are numbers which can be added to messages to keep track of the current state in transmission, i.e. to provide a tool to differentiate between similar messages.Appending a timestamp or a periodically increasing sequence number to a message, whose integrity is guaranteed, results in an effective mechanism against replay attacks. Each message received has to be checked whether no message has been received with the same sequence number or timestamp before.

Additional mechanism has to be implemented to guarantee the integrity of the message containing the timestamp, so that an attacker cannot modify the timestamp and keep the valid content of the message.

A typical attack scenario is sending bank order to the bank, in order to submit a certain amount of money to another bank customer. If the other customer overhears this order, he could replay this valid order several times. The bank receiving the orders is not capable of detecting the attack, because the signature and form of the orders may be valid. Appending a timestamp or sequence number to the order counteracts this attack, duplicate orders will be detected.

3.4.4.2 Number Used Once (Nonce)

Nonces are used to add randomness to security mechanisms and to avoid the recognition of encrypted identical messages. Symmetric and asymmetric cryptosystems have the characteristic that plain texts always produce the same cipher texts if encrypted. Attackers can use this to analyze encrypted traffic and to set up effective attacks. To avoid each plain text generating the same cipher text any time it is encrypted, a number used once is appended to the plain text, resulting in differing cipher texts at several encryptions [NS87].

Numbers used once are used in challenge-response protocols to verify the authenticity of other entities. In challenge-response protocols a Nonce is sent to an entity whose authenticity has to be checked, i.e. the sending entity wants to check whether the checked entity possesses the secret or Private Key it pretends to possess. Therefore an encrypted Nonce is sent to the entity to be checked. This challenge is responded with the original Nonce proving the possession of the decryption key, which is the private or secret key.

3.4.4.3 Diffie-Hellman Algorithm

Diffie and Hellman introduced an algorithm for secure key exchange [DH76] in the middle of the seventies. With this algorithm two entities can exchange a secret key without previous authentication or prerequisites. The algorithm is similar to asymmetric cryptography requiring a Private Key and a Public Key derived from it. Let A and B be two entities willing to negotiate a secret K using an insecure communication medium. The algorithm consists of the following steps:

- Let p be a prime integer and g an element of the finite-dimensional field $\mathbb{Z}/p\mathbb{Z}$ so that $ord(g) = p - 1$, i.e. $span(g) = \mathbb{Z}/p\mathbb{Z}$
- A chooses as secret $a \in 0, 1, ..., p - 2$ and calculates $K_a = g^a \ mod \ p$
- B chooses as secret $b \in 0, 1, ..., p - 2$ and calculates $K_b = g^b \ mod \ p$
- A sends K_a to B, B sends K_a to A. Eavesdropping may reveal this information to an attacker.
- A calculates $K = (K_b)^a \ mod \ p = (g^b)^a \ mod \ p = g^{b \cdot a} \ mod \ p$
- B calculates $K = (K_a)^b \ mod \ p = (g^a)^b \ mod \ p = g^{a \cdot b} \ mod \ p = g^{b \cdot a} \ mod \ p$

A and B have exchanged information derived from their secrets a and b. Combining the received part of the key with the retained secret produces the common secret key K. An attacker receiving K_a and K_b cannot combine its information $g^a \ mod \ p$ and $g^b \ mod \ p$ to the secret key $K = g^{a \cdot b}$. To break the security of the Diffie-Hellman algorithm an attacker has to derive from $g^x \ mod \ p$ the logarithm of g^x to the base g. This problem is known as the *Discrete Logarithm Problem* (DLP), no efficient solution is known for it, so the algorithm is assumed to be secure (at least until an efficient solution to the Discrete Logarithm Problem is found).

The protocol of Diffie and Hellman is a very effective method for two entities to negotiate a shared secret key. It provides perfect forward secrecy (PFS), which means that even if an attacker analyzes all messages exchanged between two other entities and additionally breaks all keys previously used by them, the attacker would still be incapable to learn the shared secret key. The common secret key $K = g^{a \cdot b} \ mod \ p$ can only be calculated by knowing the public g and p and the secret a and b. But a and b can be destroyed by A and B as they both calculate K, so no clue remains to re-calculate the secret key K.

3.5 Security in IEEE 802.16

It is essential for the design of a secure routing algorithm for IEEE 802.16 Mesh Mode, to look at the security architecture specified in the standard of IEEE 802.16 [ICS04]. For a better understanding, we first look at the security mechanisms provided in Point-to-MultiPoint Mode, after this we look at the Mesh Mode.

3.5.1 Security in IEEE 802.16 Point-to-MultiPoint Mode

The security sublayer of IEEE 802.16 PMP mode has two main motivations:

- Protecting the network against unauthorized access.
- Encrypting communication between the Base Station and the Subscriber Stations.

The first design goal is implemented by a key management protocol (PKM - privacy key management). With PKM each SS maintains a set of keys which it shares with the BS. During network entry the BS and the new SS use public cryptography to exchange a shared secret between each other, named Authorization Key (AK). The AK is used to derive HMAC keys to authenticate messages transmitted between the BS and the new SS. Additionally a Key Encryption Key (KEK) is derived from the AK, which is used to encrypt communication in which other keys are exchanged. Hereby we introduce the Traffic Encryption Key (TEK), which is renewed periodically between the BS and each SS. The TEK is used to encrypt casual communication between the SS and BS on the MAC layer. The BS maintains a list of Security Associations (SA), which contain the security information which is important for a connection. A SA contains the participants in the security relation, the security system used and additional information that are relevant to maintain a secure connection. The BS maintains the list of available SAs, SS nodes can request to participate in a SA.

Concluding security in IEEE 802.16 PMP is limited to authorize the participating node in the network and to encrypt the link between the BS and each of the SS. The SS have almost no control about decisions related to security. All Security Associations are provided by the BS. The BS is assumed to be trusted. Routing is very easy: SS forward all messages to the BS, there are no attacks on routing, which cannot be detected by the BS. In PMP security is only needed against eavesdropping and to provide authenticity.
For more details about security in IEEE 802.16 PMP Mode see the work of Johnston and Walker [JW04].

3.5.2 Security in IEEE 802.16 Mesh Mode

The IEEE 802.16 Mesh Mode has been designed as an extension to the previously existing Point-to-MultiPoint mode. This development can be found in the security mechanisms specified for the Mesh Mode. Most of the mechanisms for PMP were taken also for Mesh Mode.

Security Associations are still used. Keying material belonging to a SA can be requested by each node. Requests are forwarded by neighboring nodes to the BS, which is responsible for the maintenance of the SAs. The corresponding reply is forwarded by the SSs in the network to the requesting node. Nevertheless, the BS decides on the keys used in the SAs. It is inefficient to use this concept to deploy system wide keys for secure end-to-end communication between SSs. The security messages which are available in IEEE 802.16 Mesh Mode are presented in Subsection 6.3.2.3.9 in [ICS04].

Besides the Security Associations, which are used to grant rights to specific services in the network, each node maintains for each neighboring node a separate TEK state machine. These keys are used to encrypt communication between the two involved nodes. Key refreshing is performed periodically with each neighboring node using a request-reply mechanism. Nevertheless, only the MAC-PDUs are encrypted and authenticated. And as the security sublayer is placed below the MAC sublayer, any link is encrypted.

From a routing layer's point of view, only hop-to-hop communication is encrypted. Messages transmitted from one node in the network to another one are in plaintext in every node on the route. Encryption is only active during the transmission on air. Forging, modification and dropping of routing and data packets are not detected by the security mechanisms provided by the standard. Further security mechanisms are needed, providing end-to-end security and dependable routing. For more details about security on the MAC layer of IEEE 802.16 Mesh Mode, see [Sar06].

Chapter 4

Routing Protocols for Wireless Mesh Networks

Overview

4.1 Source Routing - Centralized Deterministic 34

4.2 Link State / Distance Vector - Distributed Deterministic 36

4.3 Stigmergy based Routing - Distributed Probabilistic 38

4.4 Comparison and Conclusion 55

Abstract

Various routing concepts are discussed in this chapter. Starting with centralized deterministic source routing and distributed deterministic routing algorithms we show that they are inapplicable for our needs. We present distributed probabilistic routing algorithms which are designed taking advantage of routing paradigms that were observed in insect colonies.

In this chapter we look at various routing protocols for Wireless Mesh Networks, of which most of them were designed for Mobile Ad Hoc Networks. An overview of routing algorithms for MANETs is given in [Aka02]. Single-channel routing protocols can be distinguished on uniform and non-uniform protocols. Non-uniform routing protocols allow using of different routing strategies, depending on the role a node has in a network. On the other side uniform routing protocols force any participating node to follow the same routing strategy. We focus on the latter of these two categories, because each node in mesh networks acts uniformly as a routing peer, even the mesh base station.

First we analyze source routing in general and discuss some common protocols afterwards. When using source routing protocols the route is mainly determined by the source, any information about the quality of the route has to be transmitted to the source. This generates an significant transmission overhead and limits the opportunities of dynamically adapting the chosen route to the network topology. The key property of this category is, that the source decides, which route the data packet has to take.

The next category is titled "distributed deterministic", in this each node on the path decides deterministically which neighbor of the node is chosen for the next hop for the data packet. This algorithms generate less traffic overhead, due to the fact that the source has not to be informed about the current state of the route. Decisions are made locally, based on local observations and measurements. Following this rule, the planning of the route is shared among the nodes in the network, but only one fixed route is used for the traffic between two nodes in the network.

At the end of this chapter we look at distributed probabilistic routing protocols, which provide the flexibility of local routing decisions as well as the distribution of the traffic load due to probabilistic elements while choosing the next hop for a data packet.

4.1 Source Routing - Centralized Deterministic

In this section we investigate source routing protocols. These protocols are characterized by the property that the source decides centralized and deterministically which route the data packets have to follow.

4.1.1 DSR: Dynamic Source Routing

Dynamic Source Routing [JM96] is a common routing protocol for multi-hop ad hoc wireless networks. Routing functionality is composed of two autonomous tasks: route discovery and route maintenance.

During the route discovery phase which is done periodically the source node floods

the network with a route request message containing the address of the destination and a list of visited nodes. Upon receiving such a request each node checks whether it is already in this list. If is not in the list, the node appends its own network address to the list of visited nodes and forwards the request to those of its neighbors, that are not visited yet. Loops can be detected by keeping a list of forwarded route requests. If a route requests is received a second time, it is dropped.

The destination node receiving the route request converts it into a route reply and sends the message back along the visited nodes list to the source node. Any node on the route may keep a copy of such a route reply in its cache and reply to route requests, if the destination is already enlisted in its cache. The intermediate nodes forward the route reply according to the visited nodes list contained in the reply.

The source node learns various routes to the destination when it receives route replies. After choosing one of these routes, the source node adds the route list in the source-routing header of each data packet it sends to the destination node. All intermediate nodes are supposed to forward the packet to the next node listed in the nodes history of that packet. Figure 4.1 shows the principles of route discovery in DSR.

If an intermediate node detects a broken link so that the data packets cannot be forwarded it sends a route error packet over the reverse route to the source of the traffic stream, signaling that the current route cannot be used anymore. Any node on the way of this route error packet invalidates the entry for this route in their caches. This process is called route maintenance. The source node uses an alternative route to the destination when receiving a route error message. Or the source node may start a new route discovery. Many optimizations exist for DSR like Package Salvaging, caching negative information or Automatic Route Shortening.

The advantage of this routing algorithm is that the source has total control over choosing the route. This makes it is easier to secure the few centralized parts of the routing protocol. By sing a centralized routing protocol it is necessary to provide information about the route to the source. So because these information are collected distributed it requires significant transmission overhead to deliver them to the source and provide authenticity information.

The authenticity of the nodes on the route is an essential component to prevent attacks. Additionally route discoveries are costly and attacks on route replies and route error messages are feasible. Because of this an attacker can harm the routing functionality in the network with minimal effort. To thwart these attacks each participating node has to be authenticated and all relevant information has to be transmitted to the source node. This can be done (as seen in 5.3) but the centralized design does not qualify for an efficient usage of bandwidth and computational resources.

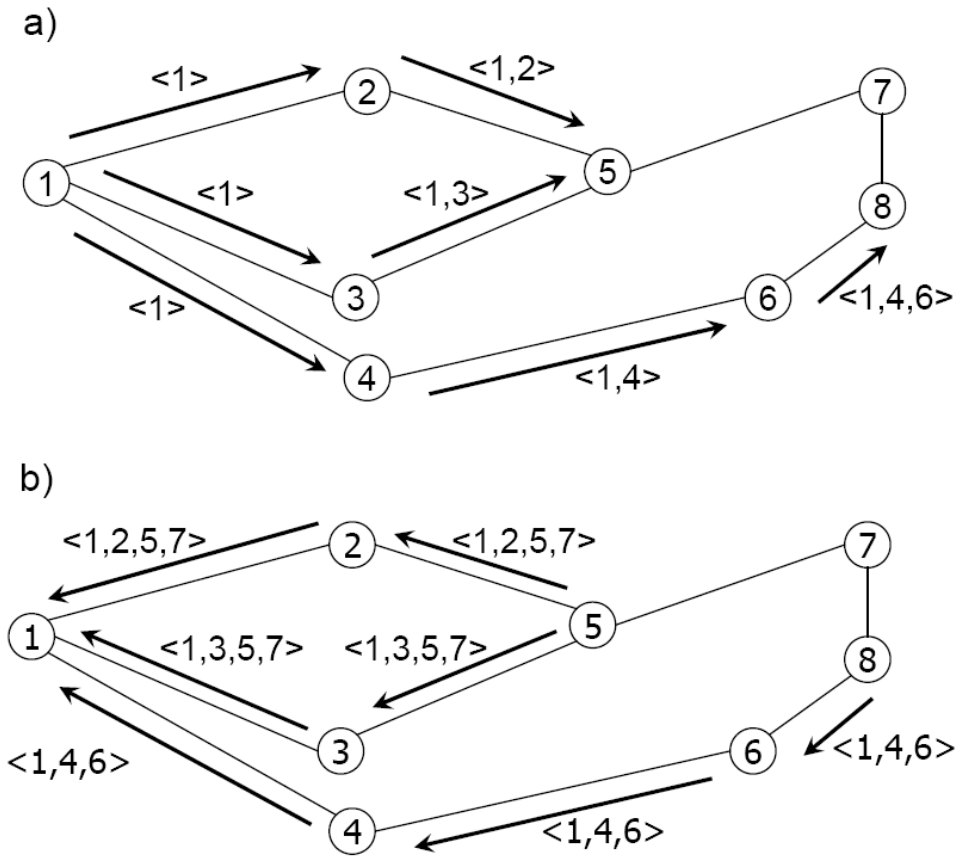


Figure 4.1: Principles of the Dynamic Source Routing Protocol: a) A route request, originated by node 1, is broadcasted through the network until the destination node 8 is reached or an intermediate node (5) has already a route to the destination node in its cache. The route request packets contain the list of visited nodes. b) The route reply containing the path to the destination node 8 is unicasted back to the route requesting node 1 using the path the route request messages traversed. Source: [Kar03]

4.2 Link State / Distance Vector - Distributed Deterministic

In this section we discuss routing protocols which are based on distributing the decision about which path is to be used. Each intermediate node chooses the next hop for the packet based on its local sight of the network. The source node is not involved in this process. This strategy leaves the control and supervision of finding the correct route to the intermediate nodes. It decreases the traffic overhead by distributing the decisions and

the necessary information.

4.2.1 AODV: Ad Hoc On-Demand Distance-Vector Routing

Ad Hoc On-Demand Distance-Vector Routing (AODV) [Per97] [IETF03] is a reactive routing protocol for ad hoc mobile networks. Traffic is only generated when a source requests a route to a destination node. Therefore the source node broadcasts a route request (RREQ) containing the IP addresses of the source and the destination, a sequence number and a hop count. The sequence number identifies the routing message, so that loops can be avoided by dropping duplicates.

Upon receiving a RREQ from the source node each intermediate node stores the hop count to the source node, for the case that the previous hop of the RREQ is chosen as next hop to the source node. It checks whether a RREQ with this sequence number has already been observed, In this case the RREQ is dropped. Furthermore it checks if there has been a route reply RREP message for this sequence number or higher, so that the RREP can be sent directly to the source node. If both cases are inapplicable, the RREQ is broadcasted again, so it is spread all over the network leaving a trail to the source node.

When the destination node receives a RREQ sent to it, it converts the RREQ into a route reply (RREP) (keeping the sequence number) and sends it back using the same links as the RREQ. Though having left a trail to the source node, the RREP is unicasted back to the source node, leaving during this process a trail to the destination node and the communication between the source node to the destination node can start. The "trail" is characterized by the IP address of the specific node and the hop count to it. If the node receives a RREQ or RREP with a lower hop count, the entry is updated. Figure 4.2 shows the principles of route discovery in AODV.

Each entry has a timeout and is deleted after a while when it is not used, so the route is maintained as long as the route is active. On the occurrence of a route error, i.e. when a link fails or is not available anymore, the previous node in the route sends a route error (RERR) message to the source and leaves it up to the source to start a new route discovery.

The advantage of this routing protocol is that it generates traffic only when a route is needed. The decision of choosing a specific route is distributed over all participating nodes in the route. Each node only needs to know to which neighbor the data packets shall be forwarded to. Considering security requirements this means that only the direct neighborhood of a node is relevant and costly messages containing state information are not needed. But still constraints of deterministic routing behavior limit its usage. As only one path is used it is attractive for malicious nodes to be on essential routing paths. Whereas selfish nodes have to avoid being on routing paths of others.

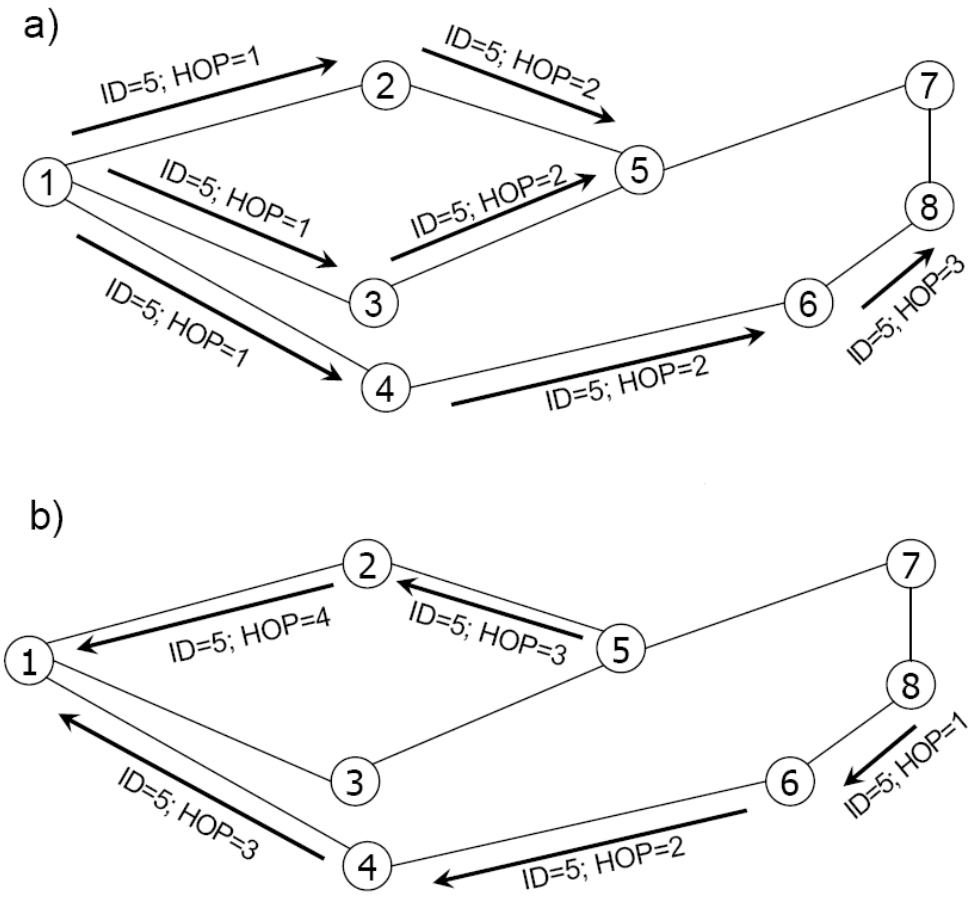


Figure 4.2: Principles of the Ad Hoc On-Demand Distance-Vector Routing protocol: a) A route request, originated by node 1, is broadcasted through the network until the destination node 8 is reached or an intermediate node (5) has already a route to the destination node in its cache. The route request packets contain the number of hops taken and a route request message identifier. b) The route reply is unicasted back to the route requesting node 1 using the path the route request messages traversed. Source: [Kar03]

4.3 Stigmergy based Routing - Distributed Probabilistic

We have seen that centralized routing algorithms produce significant overhead to keep the routing tables of the deciding nodes up to date and to communicate every routing decision on the path in each packet. This leads us to investigate distributed routing algorithms. Distributed deterministic routing algorithms are characterized by routing decisions in each node. Each network node on the route determines the next hop of the packet by itself. Due

to the deterministic aspect, only one path is used for a route although various may exist. On top of that single point of failures are considered harmful in a security context.

In this chapter we talk about distributed probabilistic routing algorithms which are characterized by local decisions on a per hop basis where each neighbor of a node has a certain probability to be chosen as the next hop for a packet. First we give an introduction to the characteristics of distributed probabilistic routing algorithms in comparison to deterministic routing algorithms.

Routing in a network is an optimization problem of finding a minimal path between two nodes in the network, minimal corresponding to certain attributes like delay, energy consumption or error rate. Centralized routing algorithms solve the problem by gathering all relevant information in one node (usually the source node) and making decisions in only this node. Using this principle very good solutions are possible, but the traffic overhead generated and the bottleneck at the central node is worth considering. Distributed routing algorithms are more suitable for wireless networks, as decisions are made locally and changes of the topology can easier be taken into account.

Still, the deterministic character of routing algorithms (like AODV [IETF03]) bears disadvantages for routing in wireless networks, since only one path is used per route. The effects of deterministic routing are besides the uneven traffic load on diverse nodes, the attractiveness for attackers to be on routes and the aversion of selfish node to participate on routes. As there is only one active path between a source node and a destination node, an attacker's goal would be to be on this single path, to be in control of all passing packets. A selfish node tries to avoid being on this single path.

Another essential drawback of deterministic routing algorithms is, that no "history" is maintained. As deterministic decisions are made only the current state of the network is relevant. Therefore each routing message is important, as it can change routing behavior dramatically.

4.3.1 Introduction to Stigmergy based Routing Algorithms

Another routing paradigm is found by observing ant colonies in nature. Ants as individuals are behaviorally very unsophisticated insects, but acting as a colony they can perform a remarkable variety of complicated tasks. In [HOW94] several observations of the complex behavior of ants in a colony are described. This complex behavior emerges from the interactions between many individual ants and their environment.

Indirect communication between ants using the environment is called *Stigmergy* and has been observed by Grassé [Gra59] as typical for social insects. But ants do not follow a central plan in their actions. They produce specific actions in reaction to stimuli from

the local environment. The clue to complex behavior is that the actions of a single ant change the local environment and by this the actions of other ants observing the local environment are changed.

We focus on routing strategies of ants to find routes (for example from their nest to food sources). It is observed that swarm intelligence (i.e. the behavior as a colony) leads to (near) optimal routing solutions (for more details see [SBHR96]).

Routing strategies of ants work the following way:

- As ants do not have knowledge about the complete topology. They can only use local information for routing decisions.
- Traversing the world / network they excrete pheromones, through which succeeding ants are attracted.
- If several branches are available the probability to select a specific branch is higher if there is already a pheromone trail. The higher the pheromone concentration the higher the probability to select this branch.
- With time the pheromone concentration decreases, this is called vaporization.
- Ants that find their destination, for example a food source, pick up some food and return to the nest following the pheromone trail.

By this simple algorithm optimal or near optimal routing solutions can be achieved. As ants on shorter paths arrive faster back at the nest, there is a higher probability that succeeding ants traverse the same path than paths on which other ants are still on the way. Shorter paths are utilized more often and attract by this more ants to use these shorter paths. We have seen that ants decide probabilistically in each step which branch to choose. As the decision is made on local observations and ants have no global knowledge about the topology, this routing strategy is to be called distributed. Figure 4.3 shows the principle of Ant routing.

It is important to denote that the principles of stigmergy can be adapted to computational (especially optimization) problems. Therefore autonomous agents with small individual capabilities are modeled after biological ants. Using autonomous agents (sometimes called Ants) to solve the problem requires communication methods between the agents (stigmergy), so that good solutions can be adopted and used as a basis for improvements. Two types of information are relevant for an autonomous agent to build a solution for the optimization problem:

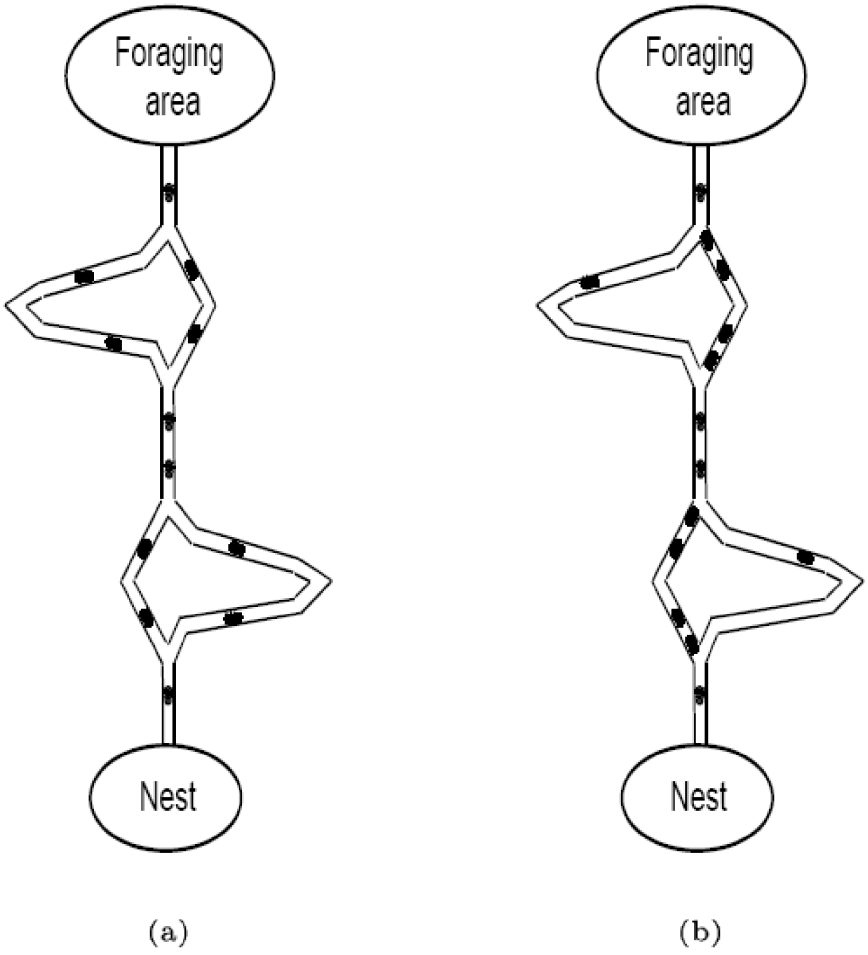


Figure 4.3: Principles of Ant Routing: a) Ants walk between a source point (nest) and an end point (foraging area) back and forth. As there are several paths to take, the ants can decide which way to take. They do not know a priori which of these paths is better or shorter. The probability for taking a specific branch is equal for all branches. b) Shorter paths have the characteristic that more ants pass in the same time interval. By this the pheromone concentration on shorter path increases faster than on longer paths. This results in a higher probability that the Ants use these shorter branches. Source: [DDCG98]

- Metrics to estimate the quality of its current solution
- Information about good strategies in the past

Using the information about the quality of previous strategies and the quality of the current solution the autonomous agents build a solution and modify the information about past

strategies stored in each node.

Four main components can be identified when describing stigmergy based routing algorithms in general.

1. Ants are periodically sent from each node to a virtual food source (destination node). One ant has only little effect on routing, several ants are needed to produce good routing solutions.

2. Ants are following a path deviated by previous solutions and the quality described by current metrics.

3. During traversing the network they modify the environment (i.e. routing tables in the nodes) to attract more nodes to use good solutions. Usually this is done on the way back from the food source to the nest.

4. Vaporization of previous information over time.

Before we give an introduction to stigmergy based routing protocols, we itemize the benefits of stigmergy based routing algorithms in comparison to deterministic routing principles:

- No broadcasting properties are needed, as ants cannot split in nature to traverse several branches.

- One ant has only little effect on routing, several ants are needed to produce good routing solutions. For security aspects this means that ants do not have to be secured completely, as several attacks are needed to have an effect on routing. With each new attack the probability of being detected raises, when intrusion detection systems are applied.

- Multipath routing leads to a better utilization of the nodes in the network and a fairer share of traffic and computational load.

- Stigmergy based routing algorithms can easily adapt to dynamic topologies without additional mechanisms. Since every branch receives a probability for being used, a link loss is automatically taken into account. The probability for the lost link is set to 0 and the probabilities of the other links are increased according to their current values.

- Due to the local characteristic of stigmergy based routing algorithms no routing tables or other information blocks have to be transmitted to other nodes. Data packets do not have to carry large routing headers denoting the path to use.

- Various additional information can be added to modify the routing probabilities. For example reputation values of neighboring nodes or additional information from the MAC layer describing link properties can have effect on routing too. In terms of cross layer routing stigmergy based protocols provide a large extendability potential. There are several routing stigmergy based algorithm taking the energy consumption into account to result in energy efficient routing [JK05].

For further details on benefits of stigmergy based routing protocols, see [GS03], [AGK+01] and [SBHR96]. Now we introduce various implementations of stigmergy based routing algorithms.

4.3.2 ABC: Ant Based Control

Ant Based Control [SBHR96] is a routing protocol which is adapting the principles of stigmergy to telecommunication networks. The algorithm is designed for load balancing in telephone networks. Calls in telephone networks are routed over different intermediate switches or nodes, depending on the routing algorithms; local congestion on a route is possible, even if other routes provide sufficient capacity. Congestion leads to unsuccessful connection attempts and so to loss of money for the network provider. A main goal of the telephony network providers is to minimize the number of lost connections due to local congestion.

Each node in the network has a pheromone table for each destination node containing probabilities of forwarding a packet over each of the neighboring nodes. Every node periodically sends an ant to a randomly chosen destination node. Ants are routed to the destination node using the neighbor node with the highest probability listed in the pheromone table as the next hop.

On the trail to the destination node, the ant modifies the pheromone tables of the intermediate nodes. The ant strengthens the pheromone value of the link to the previous hop for being used to get to the source node. So each ant leaves a trail leading to the source node on its way to the destination node. The magnitude of the modification of the pheromone value decreases with the age (hop count) of the ant to generate shorter paths. Ants are additionally delayed at congested nodes to penalize routes over congested nodes. The effect of the delay of the ant on the pheromone values can be used as a parameter to generate either more short paths or a better balanced network.

ABC is optimized for static networks in which congestion may appear but the set of participating nodes is fixed (or at least stable). Basically ABC is similar to AODV [IETF03], in both protocols routing messages are sent out to leave a trail pointing back to the source node. Since ABC uses multipath routing load balancing is automatically

applied. ABC shows highly promising results [BDT99] [SBHR96]. For Wireless Mesh Networks ABC is applicable with slight modifications. Ants have to be authenticated and replay attacks need to be prevented.

4.3.3 AntNet: Adaptive Agent-based Routing

The first routing algorithm for packet-switched communication networks based on stigmergy was developed by Gianni Di Caro and Marco Dorigo, they introduced AntNet [DCD97]. It is based on the behavior of real ants laying and following a pheromone trail and the related framework of ant colony optimization (ACO) [DD99].

AntNet uses two kinds of autonomous agents: forward and backward ants. Both types of ants have an AntID which uniquely identifies the ant. Both ants have the same structure, but differ in the fields which are allowed to be modified. Both contain a list of visited nodes. The FANT builds this list during its trip; the Backward Ant (BANT) uses this list to traverse back through the network.

The algorithm can be described as follows:

1. Each node regularly sends Forward Ants (FANT) over data packet priority queues to randomly chosen destination nodes.

2. Each intermediate node receiving a FANT first checks whether this FANT has been observed before by comparing the FANT identification information (ID) to a list of recently observed FANTs. Having observed this FANT before means that it has entered a cycle and it needs to be returned to the previous hop. In the case that the FANT has not been observed before, its AntID and time from the source to this current node is stored at the current node for further recognition of cycles; additionally the FANT is modified. The identifier of the current node and the time elapsed since generation at the source node is stored as a tuple on the stack of visited nodes of the FANT.

   For choosing the next hop of the FANT neighbors of the current node are prioritized which have not yet been chosen to forward FANTs to the node listed as destination node in the current FANT. If all neighbors have already been chosen previously to forward a FANT to the given destination node, the next hop is determined according to the routing table entries for this destination node, which contains probabilities for each neighbor to be chosen as the next hop. These probabilities are built based on the quality of previous routes. The ants are communicating with each other through these routing table entries.

3. The FANT keeps in its visited nodes list the memory of the path used (node IDs of the intermediate nodes) and the traffic conditions (time elapsed from the generation of the ant until reaching the visited nodes).

4. If the destination node is reached then the FANT is converted into a BANT, keeping the AntID and the list of visited nodes. The BANT is returned to traverse the path back to the source node using higher priority queues the used for normal data traffic.

5. Each intermediate node receiving a BANT updates the routing table entries for all nodes on the path from the destination node to the current node If S is the source node, the current node is K_i, the destination node is D and the route of the FANT was $S \rightarrow K_1 \rightarrow K_2 \rightarrow \ldots \rightarrow K_{i-1} \rightarrow K_i \rightarrow K_{i+1} \rightarrow \ldots \rightarrow D$, then the routing entries for all nodes between K_i and D are reinforced to use the link between K_i and K_{i+1}, i.e. K_{i+1} as next hop. The magnitude of the reinforcement is proportional to the goodness of the route to the specific intermediate node between the current node and the destination node. The goodness is determined by comparing the quality of the current route with the quality measurements of the best route observed until the current time. To emphasize good results a squash function is used which polarizes the results, so that the function mapping the goodness of a route to the amount of reinforcement is not linearly but exponentially growing.

   Additionally each node maintains a traffic model observing the traffic passing the node. This information is used to determine which nodes shall be selected more often as destination for fresh FANTs originated at this node. The entries in this traffic model are updated too.

6. When the BANT reaches the source node it effects the routing table and traffic model of the source node in the same way it effected the intermediate nodes. After the modification the BANT is deleted.

7. AntNet does not take into account any kind of vaporization to simulate natural pheromone fading. AntNet is designed for scenarios with low alteration in the topology.

Di Caro and Dorigo analyzed AntNet in simulations by comparing it to OSPF, SPF, BF, Q-R and PQ-Routing. Simulations show that AntNet is dominating in throughput and average packet delay when the network has high load. Simulating a network with low load leads to similar results for all routing algorithms.

Additionally to the first proposal of AntNet [DCD97], Di Caro and Dorigo suggested improvements [DCD98a] [DCD98b] [DDCG98] and changes to routing table and traffic

model updates. This second version of AntNet is sometimes referred to as AntNet 2.0 in literature [BS01].

In the simulation each routing protocol had a prephase for route establishment lasting 500 seconds. AntNet does not adapt to topology changes like common routing protocols for Mobile Ad Hoc Networks like DSR or AODV, because it has no concept of "route error" messages. The entries in the routing table for broken links are fading due to the lack of missing BANTs. New nodes in the network are only recognized, when FANTs pass them on their way to other destination nodes or when another node tries to build up a connection to this node without knowledge of its presence.

Another characteristic of AntNet is its relinquishment of broadcasting routing messages. The advantage of this attribute is that the Mesh Mode of IEEE 802.16 does not provide sufficient capabilities to broadcast message. Because of this AntNet is adaptable to the Mesh Mode of IEEE 802.16 with fewer modifications than routing protocols, which demand broadcast capabilities. The disadvantage is that using unicast transmission only takes a big amount of time to build up a route from a node to a new node that only recently entered the network. Routing protocols based on broadcasts only need to send of one request (or at least O(1) when using expanding ring search) to build up a route, if the destination node is in the maximum hop radius of the requesting node. AntNet does not give any guarantees to find a node in the network during route establishment. AntNet works similar to a depth-first search. Each FANT follows one random path to find the requested node and disregards the paths which have previously been used. So in the worst case any path leading to the requested node is left out and is only taken if all paths leading away from the requested node have been tried before.

Therefore AntNet does not guarantee quick results when a node requests a path to another node, to which no route is known for the requesting node. Looking at packet sizes of ants we see that routing messages in AntNet are smaller than the routing messages in SPF, OSPF and BF.

Considering the security issues, we see that the missing integrity of BANTs open a vulnerability to modify routes in the network. Malicious nodes can attract routes to pass them while dropping the data packets.

On the other hand we see that selfish node not processing any messages for other nodes, or nodes dropping any data, are circumvented after a short time because the routes circumventing these nodes provide a better quality. In contrast to deterministic routing protocols nodes are not bound to use the same routes all the time. When more than one route exist, traffic is split and both routes are used according to the quality. Passive attackers eavesdropping on a route between two nodes do not receive the whole traffic, if other routes exist.

Route errors do not lead to extraordinary actions unless there is no alternative link for the failed link, so an attacker cannot provoke costly action by feigning link loss. Additionally AntNet is not so susceptible for denial of service attacks like routing protocols using broadcasting for route requests, in which a single route request origination leads to operating expenses for all nodes in a given hop radius. In AntNet only a fixed number of nodes (maximum hop count) are effected by a route request (FANT).

4.3.4 AntNet 1.1

AntNet is one of the first connection-less network routing algorithms based on stigmergy (see [DCD98a]) and it is still an active research topic. Analyzing AntNet 1.0 [DCD97] and AntNet 2.0 [DCD98b] Baran suggests the following improvements [BS01]. Based on these improvements he proposes his version of AntNet, called AntNet 1.1 [Bar01].

- Intelligent initialization of routing tables:

  AntNet 1.0 and AntNet 2.0 do not specify how to initialize the routing tables, so uniform distribution is assumed. Baran suggests assigning a higher probability to neighboring nodes initially which are also destination nodes so that these neighbors are chosen more likely as next hop when packets are addressed to them.

- Intelligent updating of routing tables after network resource failures:

  The original AntNet does not cope with link errors. Broken links are used less over time by strengthening of the alternative links, but they are still used. Baran suggests to set the probability of using a broken link to 0. If a broken link becomes active again, then the previous probability multiplied with a constant shall be restored. The constant is called coefficient of memory. Using this memory effect leads to a faster recovery time after a link or node failure.

- Introduction of a noise factor and limitation of probability values:

  AntNet 1.0 does not have any limitation of the probability values, so after a while the best route tends to dominate the other routes. The probability of the best link converges to 1 and freezes in a range near 1. The probabilities for the other links converge to 0. Baran introduces a noise factor which is a probability for using alternative routes. Choosing the next hop for a packet according to the routing tables is only done with a probability equal to this noise factor. With a probability of (1 - noise factor) the node chooses the next hop assuming uniform distribution. This concept is introduced to enable nodes to find new routes which may provide better quality than the old routes. [SBHR96]

Additionally a limitation of the probabilities is introduced, which is $1 - (number\ of\ neighbors\ of\ the\ current\ node * 0.05)$.

- Dual method for selection of the next hop:

  AntNet1.1 introduces two strategies to choose the next hop for a packet. AntNet1.0 determines the next hop for a packet by choosing a neighbor randomly according to the probabilities of the routing tables, [SBHR96] introduced a strategy called hierarchical method to chose always the neighbor with the highest probability. AntNet1.1 uses both strategies with a probability of 0.5 for each option.

- Control of the number of ants in the network:

  To avoid congestion due to an uncontrolled number of ants in the network, AntNet1.1 limits the number of nodes in the network to four times the number of nodes. How this shall be controlled or maintained by a single node is not mentioned.

- Self-destruction of a Backward Ant:

  If a BANT cannot return to the source node following the route listed in the visited nodes list because of link errors, then the BANT shall be deleted. The visited nodes list is outdated and no longer valid.

Simulation results have shown that AntNet 1.1 outperforms AntNet 1.0 in all the experiments done. The dominant contributions to this result were the intelligent initialization and the dual method for selection of the next hop.

The advantages and disadvantages of AntNet1.1 are similar to AntNet1.0 compared to other routing algorithms. But due to the better simulation results AntNet1.1 shall be preferred.

4.3.5 ARA: Ant-Colony-Based Routing Algorithm

ARA [GS03] is a reactive routing algorithm for mobile, multi-hop ad-hoc networks. Its main design goal is to reduce the overhead for routing. For route establishment between a source node and a destination node, the source node broadcasts a Forward Ant (FANT) destined to the destination node. Each intermediate node receiving a FANT updates its routing table entry for the source node and stores the identifier of the FANTs. Depending on the number of hops the FANT needed to reach the intermediate node the pheromone value is modified. Fewer hops result in a higher pheromone bonus than many hops.

When the FANT is received by the destination node, it launches a Backward Ant (BANT) destined to the source node. The only difference in the processing of FANTs and

BANTs is that in reaction to the receiving of a BANT by the node it was destined to, no other ant is generated. After the route establishment phase a track has been established between the two nodes.

For route maintenance regular data packets are used. Each time a packet sent from source node S to destination node D is forwarded from node N to node M, node N increases the pheromone value of the link to M in the routing table corresponding to D. At the same time M increases the pheromone value of the link to N in the routing table corresponding to S. The link is strengthened in both directions at its usage. Pheromone vaporization is modeled by periodic decrease of the probabilities. Every t seconds the current probability for each entry is reduced by a factor q.

To detect route failures the authors assume an acknowledgment mechanism on the MAC layer. Missing acknowledgments are interpreted as an effect of the link failure. The probabilities of using this link to forward packets are set to 0 and alternative links are used. If no alternative link exists, the node passes the packet back to the previous node, so that the forwarding node can use an alternative link. Using this backtracking scheme the packet ends at the source node if no node was able to use an alternate link.

In ARA FANTs and BANTs are small in size, only containing an identifier. This way nodes can determine if they have processed recently received ants before or not. Using data packets for route maintenance reduces routing overhead as well. The authors show in simulations that ARA creates significantly less routing overhead than DSR [JM96], AODV [Per97] and DSDV [PB94] for all mobility scenarios considered in the simulations. The performance of the routing protocols is measured with the delivery rate, i.e. the ratio of successfully delivered packets. ARA's results are similar to the results of DSR, only these two routing protocols succeeded in delivering more than 95% of the packets in the simulation.

ARA is a routing algorithm with a very low overhead during route maintenance, although it is still adapting to the actual network topology. The overhead in the route discovery phase is similar to DSR, the route failure handling phase needs far less overhead, because a new route discovery is only done sporadically, when no other route exists.

ARA is a simple implementation of ant routing, using only one type of ant, containing only its identifier. All information relevant to build up routes is extracted from the network layer header. Route establishment is done using broadcasting, route maintenance is based on traffic analysis.

As ARA is based on broadcasting, it has to be modified to be suitable for IEEE 802.16 Mesh Mode. So due to this prerequisites and the lack of differentiation while processing the ants, which contain almost no information, ARA requires many changes and modifications to make it secure. Nevertheless ARA is well adequate for learning the principle

of ant routing algorithms.

4.3.6 PERA: Probabilistic Emergent Routing Algorithm

PERA [BM03] [H. 02] is a proactive routing algorithm designed for mobile ad-hoc networks with broadcasting capabilities. During initialization neighbor lists are build by broadcasting "Hello"messages. After having built a neighbor list the routing table is initialized, for each entry (destination node, next hop, probability) and each destination node uniform distribution is set. After this initialization phase the algorithm is designed as follows:

- Each node periodically sends Forward Ants (FANTs) to randomly chosen destination nodes. A FANT contains the network layer address of the source node and the destination node. Additionally it contains the current hop count and the list of visited nodes.
- FANTs are broadcasted through the network using higher prioritized queues, so that the information about the network is disseminated faster
- Each intermediate node appends its IP address and the current time. If it detects a loop in the list of visited nodes the FANT is destroyed.
- When the destination node receives a FANT it converts it into a Backward Ant (BANT) and sends it back using unicast communication.
- The BANT traverses the network using the list of visited nodes as its path back to the source node. High priority queues are used.
- Each intermediate node receiving a BANT modifies its routing tables and traffic statistics table. The quantity of reinforcement is declared to be depending on the number of hops and the delay, but it is left unspecified.
- data packets are routed probabilistically, using the probabilities in the routing tables.

The authors compare PERA with AODV [Per97] and observe lower end-to-end delay using PERA than AODV, but the throughput is lower too. PERA does not introduce new concepts in ant routing. The mixture of broadcasting of FANTs and unicasting BANTs is similar to DSR [JM96]. In conclusion we can say that PERA is a mixture of known concepts, lacking the maturity of algorithms like AntNet.

4.3.7 AntHocNet: Ant-Based Hybrid Routing Algorithm for MANETs

AntHocNet [DDG04] is a hybrid routing algorithm combining reactive route setup with proactive route probing and exploration. This hybrid approach is taken "to combine the best both worlds".

AntHocNet is similar to AntNet [DCD97] and uses the same structure for the ants and the routing tables. AntHocNet is defined by following steps:

4.3.7.1 Route Establishment

- Reactive Forward Ants (RANTs) are launched on demand from a source node to a destination node to build up a connection.
- RANTs are either broadcasted or unicasted, depending on the routing tables of the current intermediate node. If an entry for the destination node exists, then the RANT is unicasted, otherwise it is broadcasted. This is done to fasten the route establishment phase by approximating the destination node with each hop. Duplicate RANTs (same source-destination node combination, same RANT identifier) are only forwarded when they provide a better path than the previously observed.
- Each intermediate node on the route appends its IP address and the time when the RANT was received as a new entry to the list of visited nodes.
- The destination node converts the RANT on its arrival into an Backward Ant by copying the memory of the RANT into the new BANT and then deleting the RANT. Afterwards the BANT is unicasted back, destined to the source node of the RANT.
- Each intermediate node receiving the BANT build an entry in its routing table using the destination node of the RANT as destination and the node forwarding the BANT to the current intermediate node as next hop for this connection. If the entry already exists, then the entry for reaching the destination node (of the RANT) using the previous hop of the BANT is reinforced. In order to determining the new probability for using the previous hop of the BANT as next hop on a path to reach the destination node following is done: the old entry in the routing table is weighted with 0.7 and the new reinforcement value is weighted with 0.3.

4.3.7.2 Data Routing

- Data packets are routed probabilistically according to the square of the probabilities in the routing tables. The effect of squaring the probabilities is that better paths are

used more frequently.

- The usage of this probabilistic routing strategy results in better load balancing in the network

4.3.7.3 Proactive Route Maintenance

- Determined by the data sending rate from the source node to the destination node, the source node sends out Proactive Forward Ants (PANTs) destined to the destination node. For every n^{th} data packet a PANT is sent.
- PANTs are routed according to the probabilities (not squared) in the routing table entries using normal priority queues. PANTs measure the path quality on their way and reinforce the routing table entries (pheromone values) on their way from the source node to the destination node.
- Each PANT has a small probability of being broadcasted instead of being unicasted. Each receiver of the broadcasted PANT tries to forward it to the destination node. If the current neighbor node does not have any entry for the destination node in its routing table, then the PANT is broadcasted again. If a node receives a PANT broadcasted twice the receiving node is only allowed to unicast it, when its routing table contains an entry for the destination node. If no entry exists, the PANT is deleted; it must not be forwarded three times in a row.
- Since the probability of broadcasting is small and the broadcast hop limit is small too, this exploration process concentrates around the current path, trying to improve it.
- To support the detection of better variations of the current path a neighborhood discovery concept is added: Frequently each node broadcasts hello messages, which are not broadcasted again.
- When the destination node receives the PANT it converts it into a BANT (see [DCD97]) and unicasts it back.
- Each intermediate node updates the pheromone values along the path from the source to the destination of the BANT.

4.3.7.4 Link Failure

- Frequent broadcasts of "Hello" messages by each node let other nodes determine the active nodes (and new nodes) in their neighborhood.

- If a link failure occurs (observed by missing "Hallo" messages) other links are used (if existent).
- If no other link is leading to the destination node a route-repair-ant is launched, which is similar to Reactive Ant but is broadcasted twice at most.
- If the route-repair-ant finds a way to the destination node, a Backward ant is generated which should reach the node, which has launched the route-repair-ant, after a while. In this case the route is repaired.
- If after launching the route-repair-ant the Backward Ant does not arrives in a given timeout period, the current node assumes that the route cannot be repaired (from its point of view) and the node removes the destination node from its routing table.
- After having deleted the link from its routing table, the intermediate node informs its neighbors about its current state, i.e. the loss of connection to the specified destination node. This notification message contains information about the quality of the current best link to the destination node, so that the neighboring nodes can update their routing tables.

Simulations show that AntHocNet outperforms AODV observing the delivery ratio and average delay for a packet to reach the destination. AntHocNet is designed for mobile ad-hoc networks with rapid changes in the topology and the capability of broadcasting.

4.3.8 Termite

Termite [RW03] is a more recent approach which uses the principles of stigmergy to solve routing problems in mobile ad-hoc networks. Termite combines benefits from ABC, AntNet and ARA to cope with traffic congestion, relative node mobility, link quality and variability. Based on the four principles of self organization identified by the authors, they have designed the Termite algorithm. These four principles are positive feedback, negative feedback, randomness and multiple interactions, which when combined lead to stigmergy [BDT99]. The Termite algorithm consist of following steps:

There are five types of packets used by Termite:

- Data packets are routed through the network using pheromone tables. If a node does not have any entry for the destination of a received node, it issues a route request. If no route reply arrives in a given timeout, the packet is dropped.
- Route request packets (RREQ) are issued to find a route to an unknown destination. Therefore they walk randomly until they have found a node listing the destination

node in its routing table. If the RREQ cannot be forwarded anymore it is dropped. Upon discovery of a node which has pheromones stored pointing to the destination node, the RREQ is deleted and a route reply is sent to the source node of the RREQ.

- Route reply packets (RREP) are sent to signalize that a pheromone trail leading to the requested node has been found. On its way from the intermediate node between the requesting node and the requested node, the RREP leaves a pheromone trail leading to the requested node. By doing this a complete trail is generated.

- "Hello" packets are sent regularly for finding neighbors and for adding them to the pheromone table.

- Seed packets spread the information of the existence of its source node. They walk the network randomly and leave pheromones for their source node.

Route maintaining is done with regular data packets. Each data packet leaves pheromones for its source node while being routed to the destination node, using the pheromone based routing tables in each node. Pheromone concentration increases linearly with each packet and decreases exponentially by time. Data packets are never routed to the node they were received from. After reaching a dead end the packet is dropped.

Through this simple mechanism Termite provides a full routing algorithm, designed for low complexity and routing overhead. Routes adapt as the network changes without the need for special routing messages. Termite provides best-effort service and multipath routing. Some packets may get lost and are dropped due to the probability of being routed into dead ends or even away from the destination node. To improve the quality of routing, nodes in promiscuous mode can observe routing information from the traffic in their neighborhood. By this they learn more about the network and discover new nodes.

A limitation of Termite is the fact that it does not provide any route discovery mechanism which guarantees successful route establishment. The concept of route discovery in Termite is efficient for repairing an existing routes or for finding routes to a nearby nodes. But building a route to a node which is far away using random walk requires several attempts in average to succeed. Termite does not require broadcast capabilities so it is suitable for IEEE 802.16 Mesh Mode without drastic modifications.

4.3.9 Ant-AODV

Tham et al. introduced in [TMS02] a mobile agents based routing protocol for mobile ad-hoc networks. It is declared as a hybrid routing protocol combining AODV [Per97] with ant-based routing algorithms. The ants described by the authors differ from the ants in the

previous section. Ants here are independent from their source nodes. They walk through the network carrying a list of the 15 last nodes traversed. Using this information each node updates its routing table. For route discovery a node can wait for a passing ant having the destination node in the visited nodes list. The requesting node can broadcast an on-demand route discovery packet to find a route leading to the destination node. The route request packet is re-broadcasted by each intermediate node. If it reaches the destination node, it is converted into a route reply message and unicasted over the shortest path back to the requesting node using source routing. Periodic "Hello" messages are broadcasted to maintain a neighbor table. Route failures are reported using Route Error messages, which are sent to the source node thereby informing the intermediate node about the link failure.

The concept of independent mobile agents providing route information to the nodes depends on the benevolence of the participating nodes. From a node's point of view it is complex to supervise the correct proceeding of the routing algorithm. For example it is difficult to determine the total number of ants circulating in the network locally. The demand of broadcasting capabilities for on-demand route discovery and its insufficient mechanism for proactive route discovery require essential modifications to adapt Ant-AODV for IEEE 802.16 Mesh Mode.

4.4 Comparison and Conclusion

Reviewing stigmergy based routing algorithms we can identify the advantages of probabilistic routing algorithms based on local decisions. In Table 4.1 we present our conclusion. In contrast to centralized routing algorithms route errors can be bypassed through the usage of alternative routes, if no alternatives exist, by locally launching route discovery mechanisms. Probabilistic routing adapts the routes to changes in network topology and fastens the integration of new nodes. ABC, AntNet and Termite do not demand broadcasting capabilities, so they can be used without modifications in IEEE 802.16 Mesh Mode. By looking at the overview of the current ant-based routing protocols we see which concepts can be combined to improve the quality of ant-based routing mechanism.

For route discovery probabilistic random walk or broadcasting can be used. Spreading additional advertisement messages increases the probability of being reached by a random walking route request ant destined to the advertising node. Having established a route to a destination node, route maintenance can be done by sending routing messages or by sending regular data messages. Special routing messages can thereby explore new and better paths (see AntHocNet in Subsection 4.3.7) and are better suited for security mechanisms. Generally these route maintaining ants are sent periodically to all known destination nodes. In contrast to this time based approach, some of the ant-based routing

algorithms introduced in the previous chapter stick closer to the approach of the biological ants. Each ant lays a pheromone trace, so some ant-based routing algorithms (see Termite in Subsection 4.3.8) increase the pheromone concentration on the links based on the amount of data flow passing it. For security reasons it is more complex to counteract the misuse of all data packets than the misuse of special routing messages.

Stigmergy based routing algorithms are in many terms applicable for Wireless Mesh Networks (WMNs). Most of them do not relay on broadcasting (which is rare for wireless networks), they provide good results and several attacks have to be done in combination to have significant impact on routing, as one ant has only small effect on routing. But many attacks are easier to detect than only a few, when using intrusion detection systems.

AntNet is the most mature candidate among the routing algorithms presented, for AntNet even security issues have been explored [ZE02]. We conclude that among the stigmergy based routing protocols (which itself have several advantages in comparison to deterministic routing protocols): AntNet is most promising to fulfill our needs in terms of a secure routing protocol for WMNs.

	ABC	AntNet1.x	ARA	PERA	AntHocNet	Termite	Ant-AODV
Requires broadcasting	no	no	yes	yes	yes	no	yes
FANTs modify routing prob.	yes	no	yes	no	no	no	yes
BANTs modify routing prob.	-	yes	yes	yes	yes	yes	-
Data packets modify routing prob.	no	no	yes	no	no	yes	no
Suitable for IEEE 802.16	-	++	O	-	O	+	–

Table 4.1: Comparison of the Characteristics of Ant Routing Algorithms

Chapter 5

Security Solutions in Mobile Ad Hoc Networks

Overview

5.1 Incentive Based Solutions 58
5.2 Overhearing Mechanisms: Watchdog and its Uses 61
5.3 Asymmetric Cryptographic Mechanisms 76
5.4 Symmetric Cryptographic Mechanisms 84

Abstract

After having investigated common security terms and mechanisms and diverse routing algorithms, in this chapter we present security solutions for routing in MANETs discussed in literature. Although our scenario is slightly different, we can still learn from the solutions presented in this related topic. We first give an introduction to incentive based solutions where nodes in the network are awarded for contributing resources. After this we look at various watchdog mechanisms, where each node is given a tool for monitoring (mis)behavior of neighboring nodes. Since these mechanisms are not sufficient to guarantee all protection goals, we look at both asymmetric and symmetric mechanisms to protect routing messages.

First we look at security mechanisms for Mobile Ad Hoc Networks (MANETs). MANETs and mesh networks are similar in concept; both are built of nodes not knowing each other. In both networks each node has to forward packets to extend the range of connectivity. The concepts for motivating nodes to participate in routing discussed for MANETs show us the main principles which can be used to achieve this goal. The IEEE 802.16 Mesh Mode scenario gives some limitations to the adaptability of these principles. Therefore we analyze the limitations of IEEE 802.16 Mesh Mode in detail and observe that the security mechanisms provided by the IEEE 802.16 standard are not adequate.

In this chapter about security solutions for routing in MANETs we first look at incentive based solutions, where participating nodes are being motivated to behave compliant to the routing algorithm. Since incentive based solutions do not enforce cooperation nor do they reveal misbehavior, the next sections discuss current trends for solving these problems.

For the detection of misbehavior in routing activities in MANETs several overhearing mechanisms spying on the behavior of neighboring nodes has been introduced. Based on these observations the behavior towards the observed node can be adapted to become less vulnerable to attacks. After discussing overhearing mechanism we talk about cryptographic mechanisms enforcing protection goals in security.

5.1 Incentive Based Solutions

The main problem of routing in a mesh network or a MANET is that every node in the network has to participate in routing to keep the total throughput (ration of packets received to originated) of the network up. In [Buc04] Sonja Buchegger shows that in a MANET with 50 nodes throughput decreases to 35% when 20% of the nodes are not participating in DSR. With the following two concepts participation in routing is awarded so that the nodes in the network are attracted to participate in routing.

5.1.1 Nuglets

Nuglets [BH00] is an incentive based security mechanism adaptable to any routing protocol. The main idea of this concept is that nodes forwarding packets originated by and destined to other nodes earn a small amount of a virtual currency called nuglets. Nuglets earned by forwarding are needed to either send or to receive packets. If a node stops forwarding data packets for other nodes the node will be unable to send or receive data packets itself.

The authors introduce two models which define who pays for a sent packet: the Packet Purse Model and the Packet Trade Model.

In the Packet Purse Model the originator of a packet appends a virtual purse containing n nuglets to the packet. The number n defines the maximum hop count of the packet. Each intermediate node on the route of the packet to the destination picks a nuglet out of the purse and forwards the packet to the next node. Therefore each intermediate node is awarded for forwarding the packet with one nuglet. In this model the originator of the packet has to pay with nuglets for routing its packet to the destination node.

In the Packet Trade Model the destination node is paying for the packet. A node originating a packet sells this packet to the next hop in the route for 1 nuglet. This first hop sells the packet for 2 nuglets, earning 1 nuglet by this forwarding process. The n^{th} intermediate node sells the packet for $n + 1$ nuglets to the $(n + 1)^{th}$ intermediate node on the route. The destination node has to pay $m + 1$ nuglets, where m is the number of intermediate nodes on the route between the source and the destination node.

In [BH02] the authors introduce an improvement to their original Nuglets concept. The new version only requires a tamper proof hardware module for core functions (modifications on the nuglets counter). Additionally, packets have cryptographically protected security headers.

The concept of incentive based forwarding can be used to enforce participation in routing. Malicious nodes forwarding fewer packets for other nodes will run out of nuglets and cannot send or receive data packets by themselves. But this concept has some limitations which lowers its applicability.

The clearance of nuglets is required to be done in real-time; any delay during this operation will delay the delivery of the packet. If not enough nuglets are circulating the transfer rate decreases and performance gets worse.

Another problem occurs at nodes residing at the outer regions of the MANET: they will get fewer opportunities to forward a packet for other nodes than nodes in regions with dense population. Therefore they will run out of nuglets while being conform to the routing protocol. This occurs especially when using the Packet Purse Model. A mechanism is proposed to refill nuglets at a central authority, but this requires additional overhead and the delay leads to bad performance.

The Packet Trade Model is vulnerable to network overload, since the sender does not have to pay. This concept can also be used to make other nodes in the network loose great amounts of nuglets using distributed denial of service attacks. The attacker can attract other nodes to send packets to the victim node, generating a great amount of packets, for which the victim has to pay.

The most important attribute of the nuglets concept is that it needs tamper-proof hard-

ware to avoid manipulation of the amount of nuglets stored at the own node. With the power to change the amount of nuglets at the own node, without having to participate in routing (i.e. by direct hardware manipulation) the concept of virtual currency would be pointless. No node would have to forward packets of other nodes to earn nuglets, which can then be used to send or receive packets. With the ability of changing the amount of nuglets without interaction with other nodes, the necessity of having to behave compliant would not exist. Tamper-proof hardware is hard or even impossible to realize [AK96] [AK97], so the concept of using nuglets as incentive for routing is limited in its applicability.

5.1.2 Sprite

Zhong et al. introduced in 2002 a simple, cheat-proof, credit-based system for Mobile Ad Hoc Networks [ZYC02], called Sprite. The main idea in Sprite is similar to the main idea of the nuglets system. Nodes can earn virtual money by forwarding packets for other nodes. This money can then be used to utilize the network for own needs (sending or receiving packets).

In contrast to the nuglets system clearance is not done in real-time, but in times, when a broadband connection is available. Sprite requires a trusted Credit Clearance Service (CCS) node for handling the transaction. The calculations required for clearance are done the following way:

- Each node keeps a receipt of all the packets it has forwarded.
- Upon having a broadband connection to the CCS node, the receipts are transmitted.
- For each receipt which is also confirmed by the next hop of the corresponding packet a certain amount of virtual currency is paid.
- For the creation of data packets the originating node has to pay a small amount of this virtual currency.

The concept can be used to enforce forwarding of packets in the network. As the next hop of the packet has to acknowledge its receipt too, any dropping of packets will be detected by the CCS node. The concept can be implemented without the requirement of a tamper-proof module, the solution is purely software based. The CCS node defines the limitation of the concept. First of all the CCS node has to be trusted by all nodes. Whereas this requirement can be fulfilled in our IEEE 802.16 scenario where we have a mesh base station, in MANETs it is not easy to find a per se trusted node.

The biggest problem of this concept is the great amount of data generated by the receipts which each node has to store. Having a high speed wireless network with several megabits per second huge amounts of traffic are generated at each node. The authors speak of 100 to 200 Bytes per packet which is stored as receipt. The combination of this size with the amount of traffic passing each node in a high speed wireless network results in challenging requirements on the memory size of each node.

Even if the problem of the memory required for all receipts could be solved, another problem would still lead to bad performance: The transmission of the receipts generates significant traffic overhead. And as no formal specification of the type of incentive provided by such a system is given, we cannot predict whether the efforts invested result in a sufficient benefit.

In conclusion we can say that Sprite provides mechanisms to supervise the forwarding behavior of each participating node. This can be used to detect misbehaving nodes. However due to the additional overhead generated by the storage and the transmission of the receipts Sprite is not well suited for the IEEE 802.16 Mesh Mode scenario.

5.2 Overhearing Mechanisms: Watchdog and its Uses

As shown incentive based principles either require tamper-proof hardware to prevent misuse of the clearance, if performed in real-time. Or, if clearance is performed after the forwarding procedure, it is hard to efficiently give proof of the forwarding work which had been done. The proof of the work done requires either a lot of memory at the forwarding nodes or significant network usage to report the current status frequently.

In this section we discuss the concept of overhearing the activities of direct neighbors. By spying on neighboring nodes, each node can check whether its neighbors behave compliant to the routing algorithm. Since we have a wireless network each node is capable of receiving messages from its neighbors even if the packets were not directed to it. Because of this feature of wireless networks a node can check whether the packet sent to a certain neighbor is forwarded by this neighbor, too. Using these observations the reputation of neighboring nodes can be defined. Based on this reputation of the neighbors each node can determine its own behavior to improve the performance of the network by avoiding malicious nodes.

5.2.1 Watchdog and Pathrater

Marti et al. introduced in 2000 the idea of Watchdog and Pathrater [MGLB00], which is the first mentioning of this principle in literature. The system the authors propose consists

of two components. The mechanism to observe the behavior of neighboring nodes, which is called Watchdog, and the mechanism for considering these observations in the routing procedure, which is called Pathrater.

The authors assume bidirectional links, i.e. each neighbor that can receive messages from the current node can be overheard too. Further they assume the possibility to operate a node in promiscuous mode, so that the node can overhear the messages its neighbors are sending, even if they are not directed to this node. The authors assume that the forwarding node knows where the packet has to be in 2 hops, i.e. if a packet is sent from node A to node B and the next hop is C, it is assumed that A knows that the next hop after B shall be C. This assumption suits well to centralized deterministic routing algorithms, like DSR (see Section 4.1.1). The authors implement their system on top of DSR but denote that the concept can be generalized to any kind of source routing algorithm. They showed in simulations that the Watchdog and Pathrater mechanisms increase throughput by 16% at 40% of all nodes misbehaving with only 6% additional network overhead.

The Watchdog and Pathrater mechanisms work as follows:

- Each packet which is forwarded by node A to node B is stored locally either completely or only those parts are stored, which must not be altered during routing.
- When the node to which the packet has been sent (B) forwards the packet to C the transmission is overheard by A. (There exists some exceptions when the overhearing fails).
- A checks whether B has forwarded the packet correctly, if yes B's reputation in A increases. If B did not forward the packet correctly, B's reputation decreases in A.
- If B does not forward the packet in a given timeout window, B's reputation is decreased in A. To sum it up: If B forwards the packet, then B's reputation raises, if B does not forward the packet or if the packet is modified incompliant by B then B's reputation decreases.
- The second component proposed is a mechanism called Pathrater, which decides whether a path can be kept up or has to be reported back to the source as not usable due to malicious nodes on it. Since a routing protocol is assumed in which each node knows where the forwarded packet has to be in 2 hops, only source routing algorithms fulfill this requirement. Each node forwarding a packet can check whether the next hop's reputation exceeds a certain reputation threshold signaling constant misbehavior. If the next hop is identified as a misbehaving node, the current node can send a message to the source node notifying it of the malicious behavior of the next hop.

- Each node maintains a list of all nodes known, combined with a reputation for each of these nodes. The reputation of nodes in use is periodically increased. Link failures and timeouts lead to decreases in the reputation. If a message arrives declaring that a node on the route has been misbehaving, the reputation decreases by a great amount. The Pathrater component takes part in the decision process after receiving route replies in DSR (see Section 4.1.1). As DSR is a source routing protocol, route requests are broadcasted and any successful route is answered with a route reply. The routes listed in the route reply are evaluated using the Pathrater component: routes with trustworthy nodes are rated higher than routes with failures or link losses even if these routes are shorter. Routes containing malicious nodes are excluded and not used.

The authors discuss several scenarios in which the watchdog mechanism is incapable of overhearing the correct forwarding of the packet by the next hop. For the following discussion we want to look at the scenario consisting of five nodes A, B C, D and E.

The *ambiguous collision problem* defines a scenario where a node C observes a collision, while listening to its neighbor D whether it forwards a certain packet. Figure 5.1 shows the ambiguous collision problem. C does not know whether D had sent the packet during the collision to the next node (E) or not. Due to this uncertainty C should not accuse D for misbehaving directly but keep on observing D for a while before sending an accusation to the source node of the route.

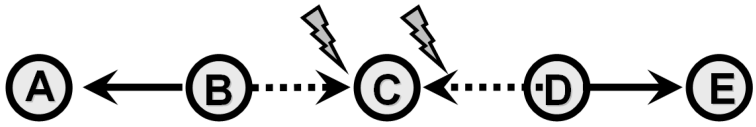


Figure 5.1: Ambiguous Collision Problem: Node C listens to the transmissions of D to find out, whether it forwards packets correctly. A collision occurs at node C. Node C does not known whether D has forwarded data in during the collision or not. In this case D has forwarded the packet to E.

Another problem is the so called *receiver collision problem*, where B sends the packet to C in a moment, in which C is already receiving or sending a message. This causes a collision at the receiver C but is not detected by the observing node A. The receiver collision problem is demonstrated in Figure 5.2.

A problem that occurs often in IEEE 802.16 results from optimized usage of the wireless medium: neighboring nodes may send at the same time. Important hereby is that the receiving node do not have collision at their side. This problem is shown in Figure 5.3.

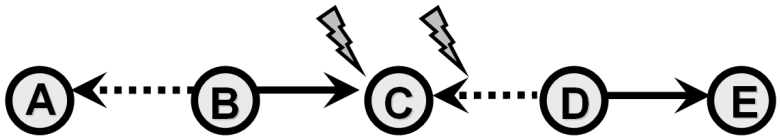


Figure 5.2: Receiver Collision Problem: Node A listens to the transmissions of B to find out, whether it forwards packets correctly. Node B forwards the packet to the next node C but a collision occurs at C. B detects the collision but does not retransmit the packet. For A it seems like B is behaving benign, as B did forward properly.

Transmitting false accusations to the source node can also be a problem. In this case A could send a message to the source of the route that B is misbehaving and shall be excluded from subsequent routes. This attack can be circumvented by comparing the accusation from other routes containing B. If B is not reported to be misbehaving on the other routes not containing A, then A seems to be lying.

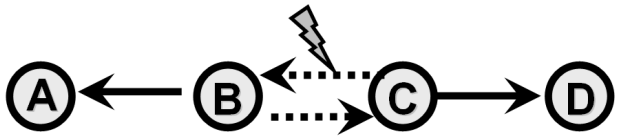


Figure 5.3: Collisions in IEEE 802.16 Mesh Mode due to Optimization: To optimize the utilization of the wireless medium, in IEEE 802.16 collisions can occur. Here node B and C transmit at the same time, accepting the collision. This is done, as the transmitted messages are received by the destination nodes A and D without collision.

Another method to avoid forwarding a packet without being detected can be achieved by controlling the transmission power. In this case an attacker B could adapt its transmission power so that the message he had to forward reaches the previous hop of the packet, which uses a watchdog mechanism, but not the next hop. The motivation for this attack is yet to be investigated. A selfish node does not gain any benefits from this behavior. Only a malicious node is willing to drop packets by investing as much energy as it is needed to forward the packet.

An effective attack on the watchdog scenario can be done by colluding misbehaving nodes. If selfish or malicious nodes cooperate and only pass their own packets, while dropping any other packet, this behavior can only be detected by the misbehaving nodes. If these nodes do not send any accusation,no other node detects the misbehavior. The authors do not discuss this scenario.

Another relevant scenario is not discussed by the authors: the Watchdog and Pathrater

system encourages selfish nodes to drop packets. Misbehaving nodes will be excluded from routes and can conserve energy, while they can still act as source or destination nodes of routes. Because of this selfish nodes benefit from the Watchdog and Pathrater system. Through conserving energy by not forwarding packets. They are awarded by excluded from routes.

Nevertheless the Watchdog and Pathrater system is the first contribution to observe misbehavior by checking the actions of the neighbors. This local observation of the behavior of neighbors is very efficient; it does not produce any additional transmission overhead.

5.2.2 CONFIDANT: Cooperation of Nodes - Fairness in Dynamic Ad-hoc Networks

In [BB02b] and [BB02a] Buchegger and Le Boudec analyze the effects of selfish behavior in MANETs. They conclude that selfish behavior will spread and dominate in a system without punishing mechanisms, this endangers the operability of the network. Therefore they propose the CONFIDANT system [BB02c], which extends the Watchdog and Pathrater concept by adding a punishing component to the system.

CONFIDANT consists of four components:

- The Monitor observes the behavior of neighboring nodes and tries to detect malicious behavior. If unconformable behavior is observed a notification is passed to the reputation system.

- The Reputation System maintains the reputations of other nodes. Own observations have larger effect on these reputations than second hand observations, i.e. reputations communicated by other nodes. If the reputation of a node drops below a certain threshold value then the path manager is notified.

- The Path Manager decides which nodes have to be avoided when choosing a route. Additionally to the avoidance of malicious nodes, these nodes are not supported in routing. Packets coming from these nodes are not forwarded.

- The Trust Manager sends and processes alarm messages indicating the detection of malicious nodes. Nodes communicate their observations using alarm messages, which are sent to neighbors not suspected for misbehavior. The Trust Manager consists of an alarm table listing the nodes that are expected to be malicious. It also contains a trust table listing the trust levels of other nodes and a list of friends to which alarms are sent when misbehavior is detected.

CONFIDANT advances the idea of Watchdog and Pathrater by introducing the concept of second hand reputations, which are reputations sent by other nodes. CONFIDANT also introduces the punishing mechanisms. When a node is detected to be misbehaving, then no service is granted to this node by the detecting node. If a misbehaving node is detected to be misbehaving by all of its neighbors, the misbehaving node is excluded totally from the network.

The authors show that CONFIDANT keeps the throughput in a network with 50 nodes and 60% of them being malicious at 80%. Unfortunately they do not discuss how the principle of exclusion from the network due to bad reputations can be used for attacks. The effects of bad accusations are only discussed in later works [Buc04]. Investigations on the processing of second hand reputations show consideration of second hand reputations always generates better performance of the network than ignoring them.

CONFIDANT has the same drawbacks as the other passive monitoring mechanisms: using only this type of overhearing can result in observations which are not clear without ambiguity (see the ambiguous collision problem in Figure 5.1).

CONFIDANT is advancing the idea of Watchdog and Pathrater, the authors show the necessity of methods for punishment to guarantee the operability of the network. The extension of second hand reputations quickens the detection of malicious nodes.

5.2.3 OCEAN: Observation-based Cooperation Enforcement in Ad hoc Networks

Observation-based Cooperation Enforcement in Ad hoc Networks (OCEAN) is an extension of DSR [JM96] to provide a mechanism to detect misbehavior in routing. The authors of OCEAN [BB03a] follow the idea of Watchdog and Pathrater and give an extension of how malicious nodes can be punished.

OCEAN consists of five components:

- NeighborWatch has a similar function as the watchdog mechanism. Its main task is to observe the neighboring nodes to detect misbehavior in forwarding packets. Each node keeps a short checksum of the packet it forwards to the next hop. This next hop node is observed for a period of time to see if the packet is forwarded or not. The result of this observation, i.e. either the packet has been forwarded or not, is communicated to the RouteRanker.

- The RouteRanker maintains the reputations of the neighboring nodes. Based on observations these reputations are modified. Misbehavior results in a decrement of the reputation, correct behavior in an increment of the reputation of the neighboring

node. The authors chose -2 and +1 as modification values. When the reputation of a node falls below -40, which is the threshold value for misbehavior, the node is added to the faulty node list.

- Rank-based routing is done by taking reputations during the establishment of routes into account. Since OCEAN is designed as an extension for DSR, route establishment is done by broadcasting route request messages to contact the destination node, which in turn unicasts the route reply containing the total route back to the route requesting node.

  A node performing rank-based routing appends a list of nodes to the RREQ. The list contains nodes, which shall not be used for routing. Any node receiving the RREQ, checks the list of nodes to avoid. First the current node checks whether the route contains a node which is listed on its faulty list. If it is so, the RREQ is dropped. If not, then the current node checks whether or not it can provide a valid next hop for this request. If the current node can forward the RREQ, it appends its own faulty list and re-broadcasts the message. So each intermediate node can decide, based on local observations, which other nodes are trusted to be part of the route.

- The Malicious Traffic Rejection component rejects all traffic coming from a node listed on the faulty list. This is done to punish the faulty node and to avoid that the faulty node adds fake source addresses and source routes to the packets pretending to be forwarding messages.

- The Second Chance Mechanism gives nodes on the fault list the opportunity to be provided with service again. Each node on the faulty list is observed for inactivity and after a fault timeout while being inactive the faulty node is taken from the faulty node list and the node is given the chance to repair its reputation.

OCEAN relies explicitly only on direct observations. The motivation for this decision is avoiding attacks based on false second hand reputations. The authors designed OCEAN as a misbehavior detection system extended with a punishment component. But OCEAN still contains some flaws which limit its effectiveness.
Any node may drop route request messages when the route contains nodes listed on the faulty list of the current node. This behavior is not punished, a selfish node can drop all route requests and relieve itself from the burden of forwarding. The authors suggest a credit based system like Sprite [ZYC02] called 'chipcounts' to keep track of the forwarding ratio of neighboring nodes. For each link the chip rate defining the forwarding ratio increases periodically. A trade-off between the application of selfish behavior and traffic contention must be found.

Another limitation of OCEAN is that the malicious traffic rejection component may generate serious contention due to mutual exclusion from service. As the detection mechanism may produce false results (see ambiguous collision problem in figure 5.1) it may occur that a good-natured node A is considered malicious by another node B and excluded from service. On the other hand Node A identifies node B as malicious node and excludes B from the network, too. After a while B gives A a chance to prove its 'goodness', but as A is avoiding to give service to B, this test will fail. A is rated bad again and re-entered into the faulty list. Now A performs the second chance mechanism for B and B fails. So using a timeout based second chance mechanism can lead to deadlocks, with nodes refusing to provide service for each other.

5.2.4 CORE: Collaborative Reputation Mechanism

Pietro Michiardi and Refik Molva propose in [MM01] [MM02b] [MM02a] a observation and reputation system called collaborative reputation mechanism (CORE). In CORE reputation represents the amount of a nodes contribution to network operations. To measure this contribution the authors give a model in which a node A can observe the execution of operations assigned to another node B. In our case the operation is forwarding a packet compliant to the routing protocol. The model defines three components:

1. The Requester, instructing a provider to execute a function *f()* giving a result.

2. The Provider executes the instruction and returns the result.

3. The Validation Mechanism evaluates the results returned by the provider.

The validation mechanism maintains the various reputations assigned to other nodes in the network. Three types of reputation are maintained. The requesting node adapts the subjective reputation of the provider based on evaluations of results returned by a provider locally. The indirect reputation of a given node is made up by the reputations reported by various other nodes in its neighborhood. The third type of reputation is the functional reputation which results from the quality of the answers given to various requests. The quality of these function results are weighted to produce the functional reputation.

CORE has two basic components to operate: a watchdog mechanism to detect misbehavior of neighboring nodes and a reputation table maintaining the various types of reputation for all known nodes.

If the watchdog mechanism detects that the next hop of a packet refuses to forward the packet, then the subjective reputation of the observed node is adapted in the observing node. In the other case, when a correct forwarding was observed, the subjective reputation

is adapted again, but additionally an acknowledgment message (ACK) is sent to the source of the node. The source of the node acts as a requester of forwarding the packet correctly to the destination node. Only positive ACKs are sent. Each node receiving ACKs adapts the indirect reputation of the forwarding node locally. These indirect reputations are made up by ACKs received, confirming the correct behavior of the node. They are combined with the subjective reputation to make up a global reputation (which is still individual for each node calculating this reputation).

This global reputation of a node defines whether this node is accepted as a requester by the node having calculated the global reputation. If the requester has a negative reputation from the providers point of view, the provider refuses to supply any service and transmits a Denial of Service (DoS) message to its neighbors. The neighbors of the provider check if this decision is acceptable by checking the reputations of the provider and the requester. If the DoS declaration is accepted the announced denial of service by the provider is not interpreted as misbehavior. If the DoS declaration is identified as inappropriate the reputation of the provider is devalued.

CORE is a framework enfolding a detection mechanism combined with a reputation system which analyzes the effects of indirect reputation in more details [MM02a]. But CORE, like the previous mechanisms, relies only on a simple watchdog mechanism which cannot give any information about the forwarding of a packet by the next hop under some circumstances. Another problem is given by the assumption that the global reputation of a node is similar in the nodes being located in the proximity of this node. Simulations have shown that this assumption is hard to fulfill. If the global reputations of a single node are differing then a node refusing to provide service to a malicious node is given a bad reputation by some of its neighbors. Based on the different local reputations they interpret this denial of service as an attack on a well-tempered node.

The authors give no information about the overhead generated by the additional ACK messages and how these could be secured against modifications. Since the concept does not include any authentication, forging and modification of CORE messages is still possible and needs additional security mechanisms.

5.2.5 Mobile Intrusion Detection System by Zhang/Lee

Yongguang Zhang and Wenke Lee propose in [ZL00] and [ZLH02] an intrusion detection system for Mobile Ad Hoc Networks. Intrusion detection system is a term describing systems used to detect anomalies or violations of rules in networks or on computers, they are installed on. The mobile intrusion system proposed by the authors is designed to detect misbehavior in Mobile Ad Hoc Networks. The system is consists of six components.

- Local Data Collection - is used to gather context information of user and system activities, as well as communication streams corresponding to this node and all communication in overhearing range.

- Local Detection Engine - evaluates the observed activities and passes the results to either the Local Reaction module if an instant reaction is necessary or to the Global Detection Engine to detect more complex attacks.

- Secure Communication - is a component to share the local evaluations with neighbors.

- Global Detection Engine - uses evaluation data received from other nodes and own observations to detect more complex attacks. Using a majority-based voting system, each node can investigate the consensus of its neighboring node regarding a suspicious node.

- Local Response - is activated when the Local Detection Engine gives the information that the network is under attack. A possible local response is the notification of the user. The reaction is limited to the reacting node.

- Global Response - is activated when the Global Detection Engine signals an attack. There is no specific reaction defined by the authors, but they present some suggestions. All nodes identifying a certain node as harmful can agree on a change of the used transmission channel or group key to exclude the harmful node from the network.

In following works [LX01] the authors investigate the principles of detecting anomalies in Mobile Ad Hoc Networks.

The components of the mobile intrusion system can be divided in local observation, evaluation and reaction and so called global counterparts, which take evaluation results of neighboring nodes into account. The detection criteria given by the authors in [LX01] are very complex and need training data. The applicability of the detection mechanism is doubtful. The definition of an initial starting data set which results in a high detection quality combined with a low false positives rate is hard to achieve. Still if detection would be manageable, the system is designed to detect anomalies related to a training set describing normal behavior. Detecting an anomaly does not always lead to its originator. Without identifying the originator an appropriate reaction is not possible.

5.2.6 Mobile Intrusion Detection System by Kargl

In [Kar03] Frank Kargl developed a Security Architecture for Mobile Ad Hoc Networks called SAM. After detailed analysis of current security solutions for Mobile Ad Hoc Networks [KS04], he comes to the conclusion that secure routing can only be assured when the following tasks are fulfilled:

- All participating nodes must be authenticated so that malicious or selfish nodes can be excluded from the network. In [KSK+04] [KK04] he introduces a node identification system called MANETID providing unique and immutable identities to MANET nodes. These node identifiers are derived from authenticated RSA Public Keys.

- The routing protocol used must be designed to prevent routing information to be modified without being detected. In [KG05] Kargl introduces Secure Dynamic Source Routing (SDSR) which relies on asymmetric cryptography to secure routing messages. A short introduction to SDSR is given in Subsection 5.3.2.

- To detect misbehavior like attacks not based on modifications but on service refusal a watchdog mechanism is needed. Kargl introduces a watchdog mechanism called mobile intrusion detection system (mobIDS).

In this subsection we give a short introduction to mobIDS as proposed by Frank Kargl. By scrutinizing current watchdog mechanisms the author denotes the detection of selfish and malicious nodes as design goals for mobIDS. The proposed system is completely decentralized and does not require any initial trust relationships between the nodes.

MobIDS does not rely solely on the observation forwarding conduct of neighboring nodes. In contrast to works of other authors mobIDS also takes the activity of the neighboring nodes into account. Following differentiation in the overhearing behavior is given:

- Promiscuous overhearing as introduced by Marti et al. in [MGLB00]. This mechanism only provides information about whether the neighboring node has forwarded packets as instructed or not. But upon detection of misbehavior the observing node does not get any information about the reason for misbehavior. As shown in figure 5.1 various circumstances exist which lead to false observations obscuring the real situation. If node A arrives at the conclusion that a packet has not been forwarded by the next hop node B, then A cannot be sure that this observation is based on misbehavior or on overhearing collisions.

- Activity based overhearing takes further activities of the observed node into account. Every node compiles lists for each of its neighboring nodes, listing the

transmission behavior of them. It is journalized whether the neighboring node has been silent, not sending anything, or if the node was transmitting messages.

- Hybrid overhearing combines both the observations of the forwarding behavior and the protocols of activity to result in more realistic assumptions on the behavior of neighboring nodes.

  If a node A detects that the neighboring node B does not forward packets the activity list compiled during previous observations is checked. If B has shown inactivity in the last time A concludes that B is out of reach or down, no misbehavior is assumed. Otherwise, if B has been active, especially sending only own messages, A concludes that B is misbehaving. The activity observations are only used to determine by which degree the reputation of the neighboring node shall be decreased when misbehavior is detected.

Using hybrid overhearing results in better detection rates [Kar03] than promiscuous or activity based overhearing only.

Additionally to the overhearing mechanism Kargl proposes an iterative probing mechanism for the detection of leaks on the route. Probes, as introduced in [AHNRR02], are testing packets which are regularly sent along the route defined by the source node (source routing is assumed). The probing messages contain an encrypted field denoting the destination of the probe packet. Each node on the route checks whether the packet is destined for it or not. The destination of the probe can only be revealed by the destination node itself, as a reaction the destination node has to send an acknowledgment message to the source node. From this acknowledgment message the source node learns, that the route is valid up to the destination node of the probe. Any node on the route dropping messages will drop probing messages destined for upcoming nodes on the route too. By this the source node can confine the set of nodes suspected for being malicious. But as probes defined by Awerbuch are special testing packets a malicious node can act inconspicuously during probing. The principle of iterative probing is shown in Figure 5.4.

Kargl introduces a system called iterative probing to detect malicious nodes on routes. He proposes an additional probing field in the header of normal data packets, so that probe requests can be hidden. Each source node S regularly sends probes to destination nodes D of its routes (S-K_1-K_2....-D). On non-appearance of acknowledgments a leak in the route is assumed and the iterative probing mechanism is started. For that each packet contains a probing field in its header keeping the identifier of the destination of the probe in an encrypted form. Probes are sent to all nodes on the route beginning from the destination node of the route iterating over all nodes up to the direct neighbor of the source node. When the first acknowledgment is received the source node learns, that

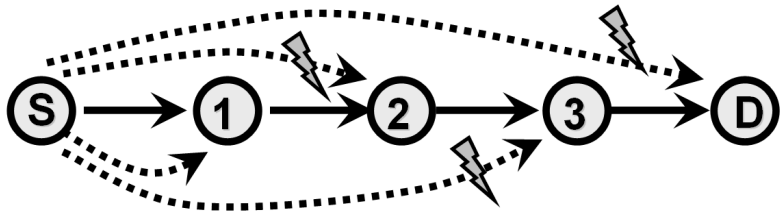


Figure 5.4: Principles of Iterative Probing: The source node S sends probes to every node on the route starting with the destination node D. As the malicious node (here: 2) drops every data packet, it drops the probes too. The first node that answers on the probe is either the malicious node itself or the node previous to the malicious node.

either the answering node K_i or the next node K_{i+1} on the route is malicious. By asking the predecessor K_{i-1} of the answering node K_i whether K_i is misbehaving or not the misbehaving node on the route is identified.

Based on observations the reputations of the neighboring nodes are compiled by each node. These local reputations take previous observations into account, but with less effect than fresh observations. Local reputations are periodically broadcasted to the neighboring nodes. All second hand reputations of a node are compiled to the global reputation describing the mean reputation of the node in its neighborhood. If a node A calculates a global reputation for one of its neighboring nodes B, which lies below a certain threshold called tolerance threshold then A tolerates that B does not get service from other neighboring nodes. If the global reputation falls below an even more lower threshold, called reaction threshold, then A does not provide any service to B until B's reputation raises over a threshold called rehabilitation threshold, where neutral reputation value $>$ tolerance threshold $>$ rehabilitation threshold $>$ reaction threshold $>$ worst reputation value possible.

The author suggests for punishment of malicious nodes the local exclusion from the network. This means that all neighbors of a malicious node N calculate the global reputation for N and conclude that this node shall be excluded. So every neighbor (or at least most neighbors of N) refuses to provide service for N. N cannot communicate any more as its messages are not forwarded.

Additionally any node calculating a reputation of N which lies below the reaction threshold sends a complaint message to a central station. This central station gathers the complaints about all nodes. If one node is constantly rated bad by various other nodes, then the owner of the node is contacted to clarify the reasons of the bad reputations. This strategy requires the owner of each node to be registered.

Kargl presents in his works an intelligent extension to the watchdog mechanism by taking additional activities of the observed nodes into account. As his simulation results in [Kar03] show, hybrid overhearing is a more efficient misbehavior detection system than simple promiscuous overhearing.

Iterative probing is designed for source routing algorithms to detect misbehavior overlooked by the watchdog system. As the probing mechanism demands an additional probing field in the header of normal data packets this requires an extension of the routing layer protocol (commonly IPv4) . Extensions to IPv4 are hardly realizable as it limits the interoperability of the protocol.

Further probing requirements are that the source node has exchanged cryptographic keys with each node on the route, so that they can decrypt the probe destination encrypted in the probe field of the header. SDSR (see Subsection 5.3.2) achieves this task but the overhead costs are significant.

By introduction of various thresholds describing a hysteresis function the author circumvents the problems caused by the usage of only one threshold (see Subsection 5.2.3). The implementation of a threshold based decision system is more intuitive than the usage of statistical models as in [Buc04]. So the authors effectively counteract possible deadlocks like mutual punishment based on punishment reactions.

5.2.7 Comparison and Conclusion

The watchdog mechanism first introduced by Marti et al. in [MGLB00] is well suited for wireless networks with a priori not trusted nodes. Assuming that the link between two nodes is not encrypted it can detect whether a packet sent to a neighboring node has been forwarded. This is done without any additional traffic overhead. But the fact that the forwarding of a packet is not observed can have several reasons as described in Subsection 5.2.1. Only one of them is caused by malicious intention. To get more detailed information on the reasons why no forwarding has been observed, a node should take other activities of the suspicious node into account [Kar03] or check the opinion of other nodes (second hand reputation).

The benefit of using second hand reputations is shown in [Buc04]. Only a few mechanisms are not using this additional information, like the original Watchdog and Pathrater system and OCEAN. Whereas Watchdog and Pathrater awards malicious nodes for refusing to forward packets by relieving them from the burden of forwarding, OCEAN can cause two neighboring nodes to get into a deadlock by mutually punishing each other for being punished.

Having compiled reputations for all neighboring nodes these reputations are used to

decide whether to provide service to a specific node or not and if this node shall be part of the route or not. As the watchdog mechanisms are designed as an extension to source routing algorithms, each node can effect during the establishment of routes that non of its suspicious neighbors shall be on the route (like in OCEAN).

The decision of using source routing algorithms as a base for watchdog mechanisms is doubtful, as these local observations are more suitable to be used in decentralized routing algorithms (see Section 4.2 and Section 4.3). It requires unnecessary additional overhead to inform the source node of leaks in the route. As shown in Subsection 4.3.7, it is more efficient to modify the route locally, circumventing the malicious node, than starting a new route request from the source node.

The costs for using watchdog mechanisms are higher energy consumption. Since listening in promiscuous mode when the node would normally sleep is energy consuming, the watchdog mechanism requires ca. 20% more additional energy (WLAN network interface, [FN01]). But compared to the energy savings resulting from direct detection and the avoidance of packet dropping simulations show that the energy investment is worth it.

Alternatives for detecting forwarding misbehavior require additional traffic overhead. Correct forwarding of a packet X is characterized by acknowledging to a node A that its neighbor B has passed X to Bs neighbor C. This can be done by so called passive acknowledgments (as entitled by Sonja Buchegger [Buc04]) which are the direct observation of the forwarding process using the watchdog mechanism. Active acknowledgments require explicit messages sent from C to A confirming the receipt of a packet X. This second method require significant additional overhead (as discussed in Subsection 5.1.2).

Based on the comparison of the watchdog based misbehavior detection mechanism discussed, in this section we conclude that there does not exists a more efficient mechanism that produces less overhead to detect misbehavior. We expect that it is more efficient to use the results of the local observations for distributed routing algorithms than for centralized routing algorithms, as unnecessary traffic overhead can be minimized. As the reputation varies over time one could constantly adapt the routing behavior to it. Distributed probabilistic routing algorithms as described in Section 4.3 seem to be optimal in combination with watchdog mechanisms.

Taking second hand reputations into account shortens the time needed to detect malicious nodes, therefore they should be used additionally. It is interesting to investigate whether relying on direct observations only, does produce similar detection results or not. Buchegger et al. have investigated the effects of rumor spreading in reputation systems [BB03b] but they assume that a malicious node shows malicious behavior to all of its neighboring nodes. Therefore further investigation is needed.

The watchdog mechanism requires the link between two neighboring nodes not to be

encrypted with individual per-link keys. Encrypted links result in the inability to detect whether the packet has been forwarded or not. Designing a watchdog mechanism for IEEE 802.16 Mesh Mode requires modifications so we are able to analyze traffic over encrypted links. Additionally the case in which the watchdog mechanism does not detect correct forwarding or produces false results due to collision have to be rare. Additionally, authentication of each node in the neighborhood is needed to guarantee that every attacker can be identified and punished. This requirement is essential for any security mechanism.

Kargl's suggestion to send complaints to the operator of the network requires only small additional traffic overhead, but has a great impact on the behavior of the nodes participating in the network. Complaints about nodes are complaints about the real persons owning this node. So for the price of giving up anonymity (which is also hindering accounting measures) punishment can be brought to the owner of misbehaving nodes. We expect that this reduces the motivation for misbehaving essentially.

Despite the advantages covered by watchdog mechanisms several issues on security in routing remain unconsidered. Authenticity is essential for watchdog mechanisms, as any concept of punishment requires that repudiation of the malicious behavior is impossible. Authenticity of the participating nodes has to be achieved by additional mechanisms. As the watchdog mechanisms are only passive components observing the forwarding of packets, various attacks remain undetected. Forging of routing or data packets cannot be detected. The integrity and authenticity of the routing packets which are crucial for the establishment and maintenance of routes must be guaranteed. Watchdog mechanisms are incapable of detecting forged packets.

An interesting challenge of wireless networks are colluding misbehaving nodes. A set of colluding misbehaving nodes can pretend to provide service to benign nodes but deny service as soon as benign nodes do not detect this misbehavior. To give an example: Let two neighboring misbehaving nodes A and B collude in a wireless network. Every packet not originated by A but being forwarded from A to B is dropped by B, and vice versa. A and B tolerate this mutual misbehavior. By this any neighbor of A assumes that A is forwarding the packets correctly (to B), so no attack is detected. A observes B dropping the packets, but A tolerates this since its packets are not dropped. In this simple scenario two nodes in combine to act as a black hole. Since this attack is not detect by any derivate of the watchdog mechanism additional security mechanism are necessary.

5.3 Asymmetric Cryptographic Mechanisms

In the previous section we discussed mechanisms to detect malicious behavior of nodes in wireless networks. Using these mechanisms a node in a wireless network can observe

whether its neighboring nodes are forwarding packets or not. One can also detect whether these packets have been illegally modified or not. But the watchdog mechanism and its derivates can only give information about the direct neighborhood.

Since the task of routing requires the protection goals to be attained for an end-to-end connection, additional security mechanisms to the watchdog mechanism are needed. The routing functionality in a network can be attacked using various methods (see Table 3.2. Attacking routing messages is one of the most common approaches to mount more effective attacks on the network or on specific nodes in the network).

As routing messages are crucial for the routing process they need special attention. Asymmetric cryptography can be used to achieve integrity, authenticity and confidentiality of routing messages. Especially spoofing, fabricating and illegal altering of routing messages shall be avoided.

In this section we discuss security mechanisms for routing in MANETs. They provide functionality to achieve end-to-end security and extend the detection of forging and invalid modifications of routing packets from the local neighborhood to the whole network.

5.3.1 ARAN: Authenticated Routing for Ad Hoc Networks

Security issues in Wireless Ad Hoc Networks are investigated by Sanzgiri et al. in [SDLBR02]. The authors focus on security flaws in DSR [JM96] and AODV [Per97] as representative routing protocols for wireless ant hoc networks. They come to the conclusion that these protocols fail to counteract attacks caused by invalid modifications and fabrication of the routing messages.

Authenticated Routing for Ad Hoc Networks (ARAN) is a routing protocol suggested by the authors to counteract these security issues. The design goals of ARAN are to provide authenticity, integrity and non-repudiation.

To provide authenticity a trusted certificate server is assumed. This trusted server is known to all of the nodes participating in the network. Before entering the network, each node requests a valid certificate from the certificate server. As response a valid certificate containing a current timestamp, the validity period of the certificate, the IP address and the Public Key of the requesting node is returned. With this certificate the new node can prove its authorization to participate in the network.

ARAN is similar to AODV, consisting of three main parts: route discovery, route setup and route maintenance.

5.3.1.1 Authenticated Route Discovery

For Route Discovery the source node creates a packet containing its IP address, its certificate, a nonce and a timestamp. This packet is signed by the source node with its Private Key and afterwards broadcasted to all of the neighboring nodes. A route discovery packet is expected to contain the certificate and signature of the source node and the node which has recently broadcasted the message. If both nodes are the same, then the route discovery message contains only one certificate and signature.

Every node receiving a route request message first checks the signatures and the validity of the certificates. If these checks succeed and the current node did not process this packet before, it removes the certificate and signature of the previous hop node and appends its own certificate and signature to the packet. After modifying the route request message, the current node signs and rebroadcasts the route request. The routing message does not contain a hop count; the authors argue that by this several vulnerabilities are avoided, as the changing hop count cannot be secured against malicious manipulation. Solutions like Ariadne [HPJ02a] and TESLA [PCTS02] have proved this statement wrong. These concepts are introduced in Section 5.4.

5.3.1.2 Authenticated Route Setup

When a route request message first reaches its destination, the destination node checks the signature of the previous hop node and the source node of the message. If the signatures are valid a Route Discovery packet is originated containing the IP address of the route requester, the certificate of the creator of the packet and the Nonce and timestamp taken from the route request message. Additionally the message is signed before unicasting it to the neighboring node from which the route request was received. The contents of the route discovery packet identify the destination of the packet and give information to which route discovery message this reply is corresponding.

The processing of route discovery messages is done the way like it was discussed for route discovery messages. Each intermediate node has to confirm the validity of the packet by appending a self-created signature and the certificate of the current processing node. The information appended by the preceding node is removed before forwarding the message to the node from which the corresponding route request message was received. By this a route is built up between the route requester and the route destination node.

5.3.1.3 Route Maintenance

Once routes are established they need no further maintenance. Actions are only taken when errors occur. If a node experiences link loss or data is to be forwarded on a route which had reached a timeout for inactivity, a route error message is generated. This message contains the IP address of the source node of the route and the IP address and certificate node, which originated the error message. In addition a Nonce and a timestamp as message identifiers are appended. This message is signed and forwarded along the path to the source node of the route. Upon receiving the error message the source node initiates a new route discovery.

5.3.1.4 Conclusion

ARAN is a reactive routing protocol assuring integrity and non-repudiation for the processing of routing messages. Authenticity and integrity is based on the availability of a trusted certificate authority, which assigns corresponding certificates to each participant in the network. Certificate revocation is done by broadcasting signed revocation messages by the certificate authority and the use of limited-time certificates. As any routing message sent during route establishment is signed by its originator and any node forwarding it, spoofing, forging and modifying of routing messages is not possible. Additionally, only authorized nodes may participate in the network.

All these benefits are achieved by signing each routing message twice and by appending in general two certificates to it. The validation of the certificates and the signatures require additional computational effort. As route requests are broadcasted each node within the hop limit of a requesting node has to contribute computational power. Information about the additional computational and traffic overhead are very interesting for the applicability of the security system, but the authors do not give any details about this.

Another disadvantage of ARAN is not providing any mechanism to detect the dropping, alteration and fabrication of regular data packets. A malicious node may drop data packets and route error messages undetected to keep a route up and act as a black hole.

5.3.2 SDSR: Secure Dynamic Source Routing

Frank Kargl researched secure routing mechanisms in Mobile Ad Hoc Networks in his dissertation [Kar03]. As a result he introduced a security framework consisting of an authentication system, a watchdog mechanism (see Subsection 5.2.6, [KK04]) and a routing algorithm called Secure Dynamic Source Routing (SDSR)[KG05].

The author identifies authenticity and non-repudiation being key elements to effective

punishment of misbehavior in routing. Each node, which wants to participate in the network, must provide a valid certificate received by a trusted certificate server. No node without valid certificate shall be served.

SDSR is a reactive source routing algorithm similar to DSR (see Subsection 4.1.1). In addition to the general goals of asymmetric cryptographic routing algorithms, SDSR provides full authentication of each node on the route from a source node to a destination node. The source node negotiates a secret key with each node on the route. This is done by following steps:

5.3.2.1 Route Request

The route request message consist of a typical message identifier, the IDs of the source node and the destination node, an unique identifier for the route request message and a secret key part for the Diffie Hellman key exchange algorithm (see Subsection 3.4.4.3). These contents must not be changed during the routing of the message to the destination node; therefore they are signed by the source node. In addition to this immutable part of the message, the self signed Public Key of the source node, a nonce N_1 and the mandatory source list only consisting of the source node are included. The route request message is broadcasted.

Each intermediate node I_i receiving a route request first checks whether this route request message has been observed before. If the request identifier is seen for the first time for this source/destination combination, the intermediate node appends its own network address to the source list. Additionally, the Nonce N_{i-1} is encrypted with a fresh secret key K_i which remains at node I_i and is not shared. Having modified the source list and the nonce, the intermediate node rebroadcasts the message.

After the destination node receives the route request, it first checks whether the signature of the immutable part of the message was generated by the source node. Afterwards a route reply message is created.

5.3.2.2 Route Reply

Since the destination node receives the same route request several times traversing over different paths, various replies could be sent back. The author suggests limiting the number of replies to a fixed value, i.e. 3. The route reply message contains the same immutable part the route request message, but the message ID has changed. Additionally the destination node signs the source route and the previous signature of the source node. By doing this integrity of the source route is guaranteed, any changes on the list of nodes will be detected.

A main goal of SDSR is to establish symmetrical secret keys between the source node and any node on the route. Therefore the route request message contains a secret key part DHK_S for the Diffie Hellman algorithm originated by the source node. The destination node also generates a Diffie Hellman secret key part DHK_D and calculates the secret key K_{SD} based on these two parts according to the Diffie Hellman algorithm. The destination node appends its part of the DH key and a signature for the common secret key K_{SD}. With it, the source node will be able to check whether the destination node has calculated the same secret key. Afterwards the destination node unicasts the route reply message along the path listed in the source route.

Each intermediate node I_i first checks the validity of the signatures of the source and destination node on receiving a route reply message. After that the current node generates a key part DHK_i for the Diffie Hellman algorithm and calculates the secret key K_{Si} shared with the source node. A signature of K_{Si} and the key part DHK_i are appended to the route reply.

To make sure that the route reply follows the path of the route request reversely, the previously encrypted Nonce N_i is decrypted with the secret key K_i. The encrypted Nonce must match the preceding Nonce N_{i-1}, which was stored while processing the route request. If the match fails, the route reply is discarded. Otherwise the messages is forwarded along the path listed in the source route.

On receiving a route reply message, the source node checks the signature of the destination node, verifying the integrity of the list of nodes on the route. By comparing the Nonce N_x in the message with N_1 the source node can affirm that the route reply message followed the path correctly in the reverse order the route request had traversed it. Afterwards the common secret keys shared with each node on the route are calculated and their signatures are checked. By this the source node is assured that the secret keys were negotiated correctly.

The author provides a second version of the protocol in which the Public Keys can be shared by appending them to routing messages if needed. The main idea is to have special fields in the routing messages which can be set by the current node if a certificate for a node on the route is not known. Each node appends its certificate to the routing message if the flag signaling the need for its certificate is set. Since routing packets traversing from the source node to the destination node twice back and forth it is sufficient for each node to signal which certificates from nodes on the routes are missing. The missing certificates are appended to the route request when the packet traverses the next time back or forth. By this certificates are only transmitted if they are missing, no additional bandwidth is wasted. For further details see [Kar03].

5.3.2.3 Route Maintenance

As SDSR is a reactive source routing algorithm, route maintenance is only needed when the route is not available any more. In case of link loss a signed route error message is sent to the source and the destination node of the route from both nodes having lost their links to each other. Route error messages contain a mandatory message identifier, the network address of the sending node, the current source route and a reason why the route error message was sent. The message is signed by the sending node so that this critical action cannot be repudiated afterwards and the integrity is guaranteed.

5.3.2.4 Conclusion

Frank Kargl provides a complex security framework for wireless Mobile Ad Hoc Networks. SDSR is a main component of this framework guaranteeing that only authorized nodes may participate in the network and that forging, illegal manipulation and spoofing of routing messages is detected. An advantage of SDSR is that it takes into account the distribution of the Public Keys and provides a mechanism to negotiate symmetric secret keys with each node on the route. This is necessary for Kargl's iterative probing mechanism to detect colluding malicious nodes (see Subsection 5.2.6). It is doubtful whether it is necessary to exchange a symmetric secret key with each node participating on the route if this mechanism is not used. Furthermore the large number of certificates transmitted during route establishment period results in a high traffic overhead.

SDSR requires broadcasting capabilities in the physical layer for broadcasting route requests. Furthermore, links between neighboring nodes have to be bidirectional and must not encrypted, so that MobIDS (see Subsection 5.2.6) can overhear the contents of packets sent by neighboring nodes. Concluding we denote that Kargl's security framework guarantees the security goals which are necessary for secure routing. Unfortunately the price is a quite restrictive set of assumptions and large routing messages.

5.3.3 SAODV: Secure Ad-Hoc On-Demand Distance Vector Routing

SAODV is based on AODV and has been introduced by Manel Guerrero Zapata in [GZ02]. It uses both asymmetric and symmetric cryptography. SAODV assumes a pre-deployed infrastructure for the Public Keys. This means that it is assumed that every node possesses the Public Key of every other node. Only the routing messages are secured in SAODV:

- The immutable parts of a routing message are signed by the source node.
- The varying parts (hop count) are secured using hash chains.

To apply security with hashchains, the last element of the chain, $h^{maxHopCount}(InitialValue)$, is included in the signed part of the routing message. When sending a route request message the $InitialValue$ is appended as *currentHopHash*, i.e. as variable part. Each intermediate node checks whether $hash^{maxHopCount-currHopCount}(currentHopHash) = hash^{maxHopCount}(InitialValue)$. If the equation is true, the hop count listed in the packet is true. Before forwarding the message, the intermediate node increases the hop count and substitutes *currentHopHash* with $hash(currentHopHash)$.

With this the fixed parts of the message are secured using a digital signature and for the hop-count it is guaranteed, that it can only be increased. SAODV provides a mechanism for intermediate nodes to answer on route requests, using information from their routing table. Therefore, to avoid attacks, the intermediate node signs the route reply and a fresh timestamp. Additionally and the destination nodes signature, which was received for authentication of the route, is appended to the route reply.

SAODV is well designed, using the appropriate cryptographic mechanisms to be efficient. Still the signature of the fixed parts of the route request is verified by every node during route discovery. And as AODV uses broadcasting during this period, many nodes in the network are stressed with unnecessary computational overhead.

5.3.4 Comparison and Conclusion

In this section we presented ARAN [SDLBR02], SDSR [KG05] and SOADV [GZ02]. All three use asymmetric cryptographic mechanisms to secure routing messages from forging and invalid modification. Asymmetric cryptography can be used to prove authenticity of messages without having to assume pre-deployed keys.

ARAN is based on DSR, most of its security benefits are achieved by signing each routing message twice and by appending in general two certificates to it. This causes a significant computational and traffic overhead, especially as the route request messages are broadcasted. SDSR is part of a security framework designed by Frank Kargl. In SDSR the deployment of the Public Keys is done by applying them to routing messages, if they are requested. On each route the source and destination node negotiate symmetric keys with each intermediate node, using the Diffie-Hellman algorithm. These symmetric keys are needed for the iterative probing mechanism, which detects malicious nodes on the route. SAODV uses for the fixed and the variable parts of routing messages different security mechanisms. Fixed parts are secured with digital signatures, variable parts with hash chains.

We conclude that for fixed parts digital signatures are a viable way to guarantee au-

thenticity, integrity and non-repudiation. Hereby it is important to limit the number of nodes having to verify the signature, to decrease the computational overhead. A secure routing algorithm needs to provide mechanisms to deploy the needed keys. Variable parts of routing messages cannot be secured in a feasible way with asymmetric cryptography. Other solutions for this problem have to be investigated.

5.4 Symmetric Cryptographic Mechanisms

In this section we present solutions for securing routing discussed in literature, that do only use symmetric cryptography. Symmetric cryptography, as described in Subsection 3.4.2 needs less computational power than asymmetric mechanisms. The drawback is that the secret keys have to be deployed, kept secret and are bound to a fix group of users, i.e. cannot be reused to deploy a secret with other nodes. Symmetric cryptographic solutions provide mechanisms to guarantee authenticity and integrity, but the drawback is that not all nodes in the network are able to verify this, only nodes which possess the same keys. Signatures based on asymmetric cryptography and providing authenticity and integrity could be verified by any node, which has the Public Key of the sender. Using symmetric cryptography, this is not possible. Kai Inkinen has investigated in [Ink04] the flaws of asymmetric cryptographic mechanisms used in MANETs and discussed the benefits of symmetric cryptographic solutions.

5.4.1 TESLA: Timed Efficient Stream Loss-tolerant Authentication

TESLA is a mechanism for source authentication in broadcasting, introduced by Perrig et al. in [PCST01] and [PCTS02]. TESLA provides integrity in authenticity of broadcasted messages with delayed verification. Therefore all nodes in the neighborhood of a sending node need to be loosely time synchronized. Time is the basis for the security mechanism. TESLA works as follows:

- Each sender splits time in a chain of defined time intervals.
- Additionally each sender calculates a "one-way chain of self-authenticating values", one authentication value per time interval.
- The authentication values have the property, that every current value can verify the correctness of older authentication values.
- Authentication values are undisclosed periodically after a defined time period.

- A HMAC (see Subsection 3.4.2.2) with the current authentication value as key is appended to every packet which is broadcasted.
- Every receiver verifies that an undisclosed key has been used and buffers the packet.
- Upon disclosure of the current authentication value, the receivers check the validity of the HMACs of the buffered packets.

TESLA provides security features like asymmetric cryptographic mechanisms, but using only symmetric cryptography: Every neighbor of the sending node can verify the authenticity and integrity of messages without sharing a common key, like it is needed in symmetric solutions. Forging and modification of messages is not possible. Still the drawback is, that TESLA provides only one hop authentication and is only designed for securing broadcasts. Various works base on TESLA solving these drawbacks. We present Ariadne and TIK in the next two subsections.

5.4.2 Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks

Hu, Perrig and Johnson have introduced Ariadne in [HPJ02a]. Ariadne is a routing algorithm based on DSR, foreclosing invalid modification and forging of routing packets. Therefore three mechanisms are integrated:

- Digitally signing every routing packet provides authenticity and integrity.
- HMAC authentication can be used. This requires that all nodes have pair wise shared keys. The authenticity and integrity of packets is verified at the destination node. This mechanism is used in SRP too (see Subsection 5.4.4).
- Ariadne with TESLA is innovative in compare to the other two solutions.

Ariadne with TESLA works as follows: A sender initiates a route request like in DSR to a destination node. Each intermediate node checks the packet for freshness and validity and appends its HMAC and current authentication value. As the packet arrives at the destination, all HMACs are verified using the authentication values of each intermediate node. The destination node computes the HMAC for this packet, clears the list of authentication values and returns the packet as route reply message. Now each intermediate node waits until the next hop on the source list discloses and verifies its previous authentication value. If this is done and the HMAC of the next hop in the route reply is valid, then the route reply is processed and forwarded.

Ariadne provides a solution to provide integrity and authentication without using asymmetric cryptography. But with this, several drawbacks have effect. Routing packets increase in size with every hop dramatically, as every intermediate node appends an HMAC. Having to wait in every hop for the disclosure of the authentication value of the next node in the source list increases the delay of the routing packets. Still Ariadne is a fair solution to provide desired security with symmetrical cryptography.

5.4.3 TIK: TESLA with Instant Key Disclosure

Hu, Perrig and Johnson introduced in [HPJ01] and [HPJ02b] a concept to anticipate wormhole attacks. In a wormhole attack two malicious nodes have a faster connection to each other than the other nodes have to each other, i.e. using wired instead of a wireless communication. By this they connect two parts of the network by a shortest path and attract all traffic between these two parts of the network to use this connection. Having a high fraction of traffic between the two sub-networks going over a single link between two malicious nodes is harmful.

To thwart this attack the authors introduce TIK, which stands for TESLA with Instant Key Disclosure. TIK provides two mechanisms to detect wormholes:

- Geographical leashes providing information about the geographical distance a packet has covered.
- Temporal leashes describing the time a packet has traveled.

When using geographical leashes, the sender appends to every message its geographical position and the time it sent the packet. Every intermediate node on the route to the destination node verifies whether the distance covered by the packet is acceptable or not. If the packet traveled to fast, then a wormhole attack is assumed. The nodes participating in the network need loosely synchronized clocks.

Temporal leashes use a similar idea to counteract wormhole attacks, but they need tightly synchronized clocks with a maximum deviation of a few microseconds. A sending node includes in every packet it sends the time at which the message is sent. The destination node calculates the time needed between sending and receiving the packet. With the time the packet needed to travel and the common speed of transmissions using the wireless link, the receiving node calculates a radius the packet may have covered. If the packet seems to have traveled too far, then a wormhole attack is assumed.

TIK is a mechanism to detect wormhole attacks by identifying packets that travel to fast when assuming wireless communication. The main limitation of this solution is its dependency on very tight clock synchronization. Various conditions have effect on the

time needed to process and forward a packet. Especially the capabilities of the device used, the number of its neighboring nodes, even the content of the message has effect on the processing time, if cryptographic mechanisms are used additionally. Due to the criticality of the processing in combination with its unpredictability, TIK does not suit our needs.

5.4.4 SRP: Secure Routing Protocol

The Secure Routing Protocol [PH02] has been introduced by Papadimitratos and Haas in 2002. SRP is a source routing protocol and very similar to DSR. The main idea for providing security is to use HMACs for authentication and verifying integrity of routing messages. The protocol assumes that the source and destination nodes of a route have already negotiated a secret key. The sender appends to all routing messages a HMAC using the secret key. Intermediate nodes on the route act like in DSR, they perform no security related actions. The destination node of a packet checks the authenticity and integrity of the packet by verifying the included HMAC.

SRP is a very simple routing protocol, providing minimal security. Many security issues are not solved. The negotiation of the secret keys is not described, data packets are not secured and black hole attacks can be done without being detected. The simplicity of SRP comes with unsolved security problems, which need to be solved in our scenario.

5.4.5 SEAD: Secure Efficient Ad Hoc Distance Vector Routing Protocol

SEAD has been introduced by Hu, Johnson and Perrig in [HJP02]. SEAD is based on the Destination-Sequenced Distance-Vector Routing (DSDV) protocol. In DSDV each node sends periodically a list of destination nodes and distances to its neighboring nodes. By this its neighbors can determine the hops needed for reaching the destination nodes over the transmitting neighbor.

Authenticity of the packets is added by hash chains. Each route update message contains one hash value for each destination entry. Any node receiving the route update can check its authenticity by verifying the current hash value with hash values observed in older route updates. As SEAD uses hash chains, an attacker cannot advertise a route update having a greater sequence number, than the sequence number currently used by the destination node. By this an attacker cannot advertise invalidly fresher routes for other destination nodes. Another benefit of the hash chain principle is, that nodes can only worsen the metric of a route advertisement, which they have received. They cannot increase the quality metric invalidly.

SEAD uses symmetric cryptography to secure the distance metrics contained in route update messages. A drawback of the protocol is that route update messages may get very big in size. To every destination entry a hash value is appended, a common hash value has 160 bits. Having a quite big traffic overhead, SEAD secures only the distance metrics, the rest of the routing message and data packets are not secured.

5.4.6 Comparison and Conclusion

TESLA, as introduced in [PCTS02], provides a mechanism of self authenticating packets. The authenticity of packets is revealed after a defined period of time. Ariadne [HPJ02a] is a routing protocol based on DSR, which can use digital signatures, HMACs and TESLA for authenticating routing packets. Using TESLA results in a source routing algorithm with delayed authentication of the packets used in the route discovery phase. TIK [HPJ01] is a mechanism to detect wormhole attacks by verifying that the distance a packet has covered is tolerable. To measure the distance a packet has traveled geographical leashes or temporal leashes are applied. Both concepts have limitations, which results in very strict constraints. SRP [PH02] is a routing protocol based on DSR, with minimalistic security features. SRP assumes pre-established secret keys between every pair of nodes in the network. With these keys HMACs to the routing messages are computed. SEAD [HJP02] uses hash chains to guarantee that the distance values broadcasted in route update messages are not decreased invalidly.

We conclude that symmetric cryptographic mechanisms are more efficient than asymmetric mechanisms. But symmetric mechanisms come with the drawback that every node knowing a secret key, can verify content signed with this key, but also sign content itself. So symmetric keys can only be used for authentication when two nodes share a single secret key. With this constraint, we have the problem that every pair of nodes in the network have to negotiate an unique secret key.

For authenticating a routing message at every intermediate node on the route and at the destination node, a single secret key have to be shared among these nodes. But then authentication is not possible, as every of these nodes could have sent the routing message. So we cannot use secret key mechanisms for authentication in a feasible way. Hash chains can be used to guarantee that metrics may only be increased, but not decreased.

Chapter 6

A Security Framework for Organic Mesh Networks

Overview

6.1 Bootstrapping Security in the Network Entry Phase 91
6.2 Neighborhood Establishment 97
6.3 AntSec - Secured AntNet 100
6.4 Mechanisms to Detect Misbehavior 110
6.5 Leak Detection Mechanism 119
6.6 Reputation Management 122

Abstract

In this chapter we present our security framework for secure routing in mesh networks. First, we present a solution how new nodes can enter the network while bootstrapping security. Having a basis of certificates proving the authorization and authenticity of each node, our routing algorithm, AntSec, is introduced. AntSec is a probabilistic proactive multipath routing algorithm which is resilient against forging, modification and dropping attacks. Our routing algorithm is supplemented by a watchdog mechanism coping with encrypted links, this mechanism is called WatchAnt. Additionally, we present a leak detection mechanism in order to detect leaks in the route due to colluding malicious nodes. To our knowledge both WatchAnt and the Leak Detection Mechanism are the first contributions to the problems they solve using no cryptography.

As we have seen in the previous chapter, IEEE 802.16 does not provide mechanisms to secure the routing functionality of the network in the presence of malicious nodes. The security mechanism provided by the standard is designed for a client-server scenario and not for the infrastructureless ad hoc topology which is characteristic for Mesh Mode networks.

Therefore new security mechanisms have to be suggested to fulfill the needs for secure routing in IEEE 802.16 Mesh Mode. The new security mechanisms should fit the premises defined by MAC layer of IEEE 802.16, i.e. broadcasting is not possible, links between two nodes may be encrypted.

In this chapter first we take a look at the node entry procedure, which is initiated when a new node wants to join the network. During this phase the new node is authenticated and receives credentials to be able to prove its authorization to other nodes in the network. The components of our security framework are shown in Figure 6.1.

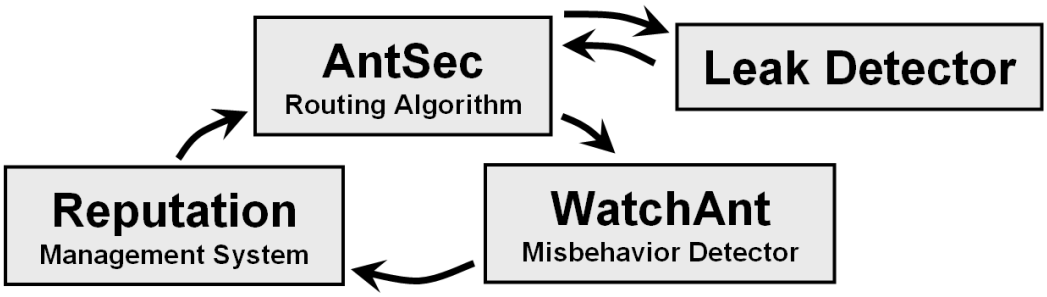


Figure 6.1: Components of our Security Framework: AntSec, a secure routing algorithm. WatchAnt, a mechanism to detect misbehavior. The Reputation Management System, combining the observations with the routing strategy. And Leak Detector, which detects malicious colluding nodes on the network.

After the node entered the network and established links to its neighboring nodes it is ready to participate in routing. We introduce a stigmergy based proactive routing algorithm called AntSec which is mainly AntNet 1.1 (see Subsection 4.3.4) with several security extensions to prevent spoofing, forging and invalid manipulating of critical routing messages.

Additionally, our routing algorithm is equipped with a leak detection mechanism, which can detect colluding malicious nodes on the route. To our knowledge this is one of the first mechanisms to detect colluding malicious nodes suggested for routing algorithms in wireless ad hoc or mesh networks. To detect misbehavior during the forwarding opera-

tion we suggest two mechanisms. The first mechanism is applicable for wireless networks with unencrypted links between neighboring nodes. In this case each node can listen in promiscuous mode to the communication of neighboring nodes (although exceptions to this rule exist, see Section 5.2.1). A watchdog mechanism like discussed in Section 5.2 is applicable.

In the second case we look at wireless networks in which links between neighboring nodes are encrypted. Due to these constraints watchdog mechanisms like in Section 5.2 cannot be used because these watchdog mechanisms compare the content of the packets forwarded by the neighboring node with the contents of the packets which were sent to this neighbor. If links are encrypted, it makes no sense to compare a packet encrypted with the secret key K_1 with another packet encrypted with K_2. Encrypting the same plaintext with different keys result never in the same cipher text and encrypting different plaintexts using different keys result in different cipher texts, too. Watchdog mechanisms based on promiscuous overhearing do not work when links between neighboring nodes are encrypted.

We suggest a second mechanism based on challenge-response, which is introduced in Subsection 6.4.2. Its main idea is to request a list from a neighboring node denoting its forwarding behavior: over which nodes had it forwarded which packets. The answer is sent in such a way, that each of its neighboring nodes receive the answer. The requesting node and the referenced neighboring nodes can check the validity of the answer. Due to the fact that the answering node cannot lie to the report requesting node and the referenced nodes, misbehavior can be detected.

In order to consider the behavior of neighboring nodes in future routing decisions, we suggest a reputation management system maintaining the reputations of the neighboring nodes. We provide details on the implementation and discuss the consequences of using second hand reputations.

6.1 Bootstrapping Security in the Network Entry Phase

In this section we present how the net entry process of IEEE 802.16 Mesh Mode works; we look at the flaws and derive the requirements for our security system, which will build on these. The Network Entry process is described in Section 6.3.9.14 of [ICS04] and consists of 8 parts starting with finding the network to becoming a full member of it. We look at each of these 8 steps and identify the vulnerabilities which may be a danger for the upcoming layers building on this initial setup phase.

6.1.1 Scanning and Coarse Synchronization to the Network

Each node wanting to join the network (initially or due to signal loss), shall search for MSH-NCFG messages to acquire coarse synchronization with the network. To achieve this, the node shall use the last known operational parameters. Or upon failure the node shall scan the possible channels of the frequency band of operation until a valid network is found. Upon receiving a MSH-NCFG message, the node shall sync to the timestamp field of the message.

At this point there is no realistic vulnerability. Sending false timestamps in the MSH-NCFG messages could be a possible attack, but this is not realistic. Because a correct timestamp is needed to participate in the network, so the attacker would soon be excluded from the network.

After synchronizing on the PHY layer, the MAC layer shall attempt to acquire network parameters and simultaneously build a list of physical neighbors.

6.1.2 Obtaining Network Parameters

MSH-NCFG messages are periodically sent by each node in the network, so while receiving these messages the synchronization is intact. The node willing to enter the network shall accumulate MSH-NCFG messages until all physical neighbors are known, from the established physical neighbor list. Then the node shall pick a potential Sponsor Node and synchronize its time to the potential sponsor assuming no propagation delay.

After choosing a potential Sponsor Node, the Candidate Node shall send a MSH-NENT:NetEntryRequest including the Node ID of the potential sponsor to indicate that it wishes the potential sponsor to assist during the Network Entry Phase, acting as a tunneling node.

After choosing a Sponsor Node, the Candidate Node shall contact the sponsor to negotiate basic capabilities and to perform authorization. The Sponsor Node is used for tunneling the messages of the Candidate Node into the network; therefore a sponsor channel is opened for direct communication between Candidate and Sponsor Node.

6.1.3 Open Sponsor Channel

In the last step the Candidate Node requests the candidate Sponsor Node to establish a temporary schedule for further communication between them. The temporary schedule requested is termed Sponsor Channel.

Upon receiving the MSH-NENT:NetEntryRequest from the candidate Sponsor Node matching its own Node ID, the node shall either open the Sponsor Channel or reject the

request.

The candidate Sponsor Node responds in the next MSH-NCFG message. If it chooses to reject, the Candidate Node shall pick another potential Sponsor Node and repeat the procedure. If there is no answer in the following MSH-SPONSOR-ATTEMPTS, the Candidate Node interprets this as rejection and retries with another candidate Sponsor Node.

If the candidate Sponsor Node chooses to accept the request, it advertises the Candidate Node's MAC address in all its MSH-NCFG messages until the sponsorship ends and it sends a MSH-NENT:NetEntryOpen message to the Candidate Node, which is acknowledged by a MSH-NENT:NetEntryAck.

Afterwards the Candidate Node performs fine time synchronization considering the "Estimated Propagation Delay" from the NetEntryOpen message. After sending the NetEntryAck message the channel is ready to be used and will only be terminated after the upcoming authorization process, which ends with the admission of the Candidate Node to the network. By sending a MSH-NENT:NetEntryClose the Candidate Node terminates the entry process, which shall be responded by a MSH-NENT:NetEntryAck.

This part of the network entry process offers some vulnerabilities. A selfish node may reject any sponsorship, frequent rejections have to be detected and punished. The reasons for rejecting a sponsorship have to be declared, any aberration has to be examined.

The MSH-NENT:NetEntryOpen may contain a false "Estimated Propagation Delay" field, which may lead to asynchrony, so this field has to be checked. The valid delay should be calculated. At this point of the network entry, none of the participating nodes is authenticated, so there may still be some unauthorized attacker.

Now that a Sponsor Channel has been established, the Candidate Node can communicate with any node in the network using the Sponsor Node as gateway. So now the authorization and registration process with a distant node may take place.

6.1.4 Negotiate Basic Capabilities

At this point in the protocol *"basic capabilities shall be negotiated after a logical link has been established between two nodes"* [ICS04].

6.1.5 Perform Network Entry

In this step, the Candidate Node uses its link via the Sponsor Node to the network to fulfill several tasks: The Candidate Node acquires the authorization, the registration and an IP address and the accurate time. For this the standard mentions separate servers, but for security reasons we propose the usage of only one trusted server.

6.1.5.1 Node Authorization

In this step the Candidate Node shall perform authorization at the authorization node according to the PKM protocol described in Section 7.2 of [ICS04]. This procedure is described in detail in the master thesis of Damir Sarac in [Sar06]. The Candidate Node acts as the SS, the authorization node as the BS verifying the certificate of the Candidate Node. The Sponsor Node shall tunnel the messages from and to the Candidate Node as described in Section 6.3.16 in [ICS04].

The Candidate Node (CN) authorization starts with the CN sending a strictly informative Authorization Information message containing its certificate issued by the CN's manufacturer. This message is strictly informative and may be ignored by the authorization node.

The next step is to send an Authorization Request to the Authorization Node (AN) including a manufacturer-issued X.509 certificate, the cryptographic capabilities of CN and the CN's basic Connection Identifier (CID). The AN checks the certificate of the CN, chooses a crypto suite from the suggested cryptographic suites, activates an Authorization Key (AK) and sends the Authorization Key encrypted with the Public Key of the Candidate Node to the CN. Further information like the sequence number and the lifetime of the AK, as well as a set of Security Association IDs are appended.

The AK is needed for authorization of the CN, it is only relevant for communication with the BS. For further details see Subsection 3.5.2.

6.1.5.2 Node Registration

Upon receiving the authorization the node is allowed to join the network. In this step the registration node assigns a Node ID to the new node. The Node ID enfolds 16 Bits (see Section 6.3.1.2 in [ICS04]) and is smaller than the IP or MAC address in size. It is more efficient to use the Node ID as identifier of nodes in the network, therefore we apply a mechanism to avoid multiple usage of node identifiers in the network. The mechanism is presented in Subsection 6.1.5.6.

6.1.5.3 Establish IP Connectivity

In this step of the protocol the standard defines that the new node shall use DHCP to acquire an IP address. Hereby it is important to communicate only with trusted nodes in the network.

6.1.5.4 Establish Time per Day

As the node has been authorized to participate in the network, it shall obtain the time of day using the Time Protocol described in IETF RFC 868 [IETF83] over UDP.

6.1.5.5 Transfer Operational Parameters

After obtaining the information described in the previous chapters, a file containing additional parameters shall be obtained using TFTP as described in Section 6.3.9.12 in [ICS04]

6.1.5.6 Security Considerations

This section needs special attention, because here the Candidate Node communicates with a distant node, so we have to consider security issues that occur.

During the node authorization phase integrity of the messages is not guaranteed. An attacker might replay or drop the Authorization Request sent by the CN. So, to avoid a replay attack, we propose to send the certificates with a timestamp, secured with a digital signature. Another vulnerability is the usage of several servers (DHCP-, Time-Server), they all may be compromised and the authentication of all of them is not feasible. Therefore our concept uses only one trusted server, which provides all the necessary information and services.

Another important issue is the design of the authorization certificate. It will be used for further authentication with other nodes and has to be small in size to reduce overhead.

We have identified two scenarios, the open subscription-based and the open subscription-less based network. We focus on open subscription-based mesh networks, assuming the existence of a trusted third party (the network provider). Each node which wants to gain access to the network has to be authorized out-of-band, so that each user is identified and misbehaving can result in consequences for the user.

Assuming a trusted third party, we can also assume a pre-shared secret between each node and the network provider. If this is not given we assume that the certificate from the node issued by the manufacturer is known to the network provider (from out-of-band registration), the Public Key of the network provider is known to the node, too. Based on this, we perform several authentications. Before communicating any security-related information via the Sponsor Node, its identity has to be proved.

The Sponsor Node sends its certificate received from the trusted third party, proving the authorization to participate in the network and the validity of the authorization in this current time period. The Candidate Node sends its manufacturer issued certificate and a signed timestamp to prove its identity (the Sponsor Node can download the manufacturer's

Public Key from the trusted third party via the network). After this the Sponsor and the Candidate Node are mutually authenticated.

In the next steps, the Candidate Node performs its authentication and receives the certificate of authorization, which indicates that it is authorized to participate in the network for a limited time period (e.g. 1 day). The certificate contains the information about the Node ID, IP address it assigned (see RegCert in the following).

To avoid Man-in-the-Middle attacks, we use digital signatures. Due to not being compatible to the idea of several participating servers, we assume a central server for authorization, registration, IP address and time of day determination.

1. The Candidate Node sends the Authorization Information message, which is strictly informative. The Authorization Information contains a timestamp and the manufacturer-issued certificate of the Candidate Node. The information message is signed with the Candidate Node's Private Key.

2. The Authorization Request sent from the SN to the AN over the SN contains the X.509 certificate issued by the CNs manufacturer, CN's cryptographic capabilities, CN's Basic Connection ID (CID) and a timestamp. The Authorization Request is signed by the CN proving the identity of the creator. With this we avoid an attacker trying to enter the network impersonating another node's identity and thus stressing by that other node's reputation.

3. The Registration Node checks the identity of the requester and creates three certificates:

   (a) Registration - Certificate (RegCert) containing the CN's MAC, its new assigned IP, Node ID, (Hash of PubKey), validity period.

   This certificate is small and can be used by the CN to prove its authorization to other nodes in the network.

   (b) Initialization Certificate (InitCert) containing the timestamp, hash of the message received from CN and hash of the RegCert.

   (c) The Identity Certificate (IdentCert) for the node is issued during the out-of-band registration of the customer using this node. The Identity Certificate contains the Public Key of the Candidate Node.

At the end of the net entry process the following is achieved:

1. The Candidate Node now maintains a list of its neighbors in a 2-hop-neighborhood fed by information from MSH-NCFG messages.

2. The Candidate Node performed mutual authentication with the Authorization Node and the Sponsor Node.

3. The Candidate Node gained following certificates:

   - Certificate of Registration (RegCert): [Node-ID, IP address, MAC, hash(PubKey)] issued by the Registration Node.
   - Certificate of Device (ManuCert): [MAC, Public Key] issued by the manufacturer of the device.
   - Certificate of Identity (IdentCert): [Public Key] issued by Registration Node

Concluding this network entry phase has been secured using digital signatures against Man-in-The-Middle attacks. Timestamps were used to prevent replay attacks. Each participating node has been authenticated. Based on this, the upcoming protocol steps can rely on these properties.

6.2 Neighborhood Establishment

In the last section the Candidate Node has been authorized to join the network, using mutual authentication with the Sponsor Node and the central server. At this point of the protocol, only these three nodes know whether CN is authorized or not. In the next step, we propose mutual authentication with a selection of the direct links (k out of n). In contrast to the establishment of links to each neighbor, we expect that this optimization leads to better utilization of the bandwidth as links to neighbors are only established if needed.

We assume the following prerequisites:

1. Each node maintains a list of its 2-hop-neighborhood fed by information taken from MSH-NCFG messages.

2. Each node has the following certificates:

   - Certificate of Registration (RegCert): [Node-ID, IP address, MAC, hash(PubKey)] issued by the Registration Node.
   - Certificate of Device (ManuCert): [MAC, Public Key] issued by the manufacturer of the device.
   - Certificate of Identity (IdentCert): [Public Key] issued by Registration Node

Every connection to a direct neighbor means the reservation of at least one minislot (see Section 6.3.6.6 in [ICS04]), which than may not be optimally utilized. Therefore, we propose a selection of the neighbors to which MAC layer links will be established to optimize throughput and delay.

A new node needs to perform mutual authentication with each of its neighboring nodes. First we investigate what kind of solution the standard offers. The standard says:

> Section 7.2.2.2 , page 275: *"For each neighbor and each SAID identified in the Authorization Reply a separate TEK state machine is started.*
>
> *Key Request to neighbor, then Key Reply (6.3.2.3.9.5) containing BS's active keying material for a specific SAID and the remaining lifetime of the TEKs. 2 active sets of keying material per neighbor have to be maintained by each node."*

So the Key Request/Reply in Mesh Mode is only for "tunneling" the communication to the BS, which is the only node which can create new SAIDs and new security groups.
The standard lacks authorization mechanisms between two adjacent nodes. Thus we have to implement a mutual authorization mechanism. We have to accomplish two different requirements:

1. Mutual authentication of the adjacent nodes.

2. Proving the neighbor relation to authenticated nodes, i.e. checking whether the given node is really spatially near to the checking node and the messages are not tunneled by a malicious node.

Without solving the second point the following attack would be feasible:
$Nbrhood(A) \longleftrightarrow A \longleftrightarrow bC \leftarrow wiredtunnel \rightarrow Ca \longleftrightarrow B \longleftrightarrow Nbrhood(B)$
The nodes bC and Ca are both belonging to an attacker C, bC echoes each packet B transmits and Ca echoes every packet A transmits. The tunnel is only opened for communication between A and B, bC and Ca refuse communication to other nodes pretending disinterest. Relying on our first requirement only, the mutual authentication of the adjacent nodes would not detect the given case. Standard authentication mechanisms are designed for usage in a distributed environment, where the messages have to be routed over several hops and therefore the location is indefinite (but within a certain hop range). In this scenario bC and Ca would perform the mutual authentication process by echoing the messages of A and B respectively, exchange symmetric encryption keys and so guarantee confidentiality and integrity. But bC and Ca may decide to act uncooperative and misbehaving. Due to the watchdog mechanism this would be detected, but the wrong

nodes would be rated and punished. So the location of the nodes has to be derived from other factors. We state that the location of a node is precisely defined by its neighborhood. We suggest as solution proving the neighbor relation to a common neighbor by signing a list of Node IDs of adjacent nodes.

Now we present our mutual authentication protocol considering the spatial near of the node to be authenticated.

1. $A \longrightarrow B : RegCert_A, IdentCert_A$
2. $B \longleftarrow A : RegCert_B, IdentCert_B, ((DH_B,\ NonceB,\ ListOfNbrs)SigB\ )encPK_A$
3. $A \longrightarrow B : (NonceB{+}1, DH_A, (NonceA)encDH_{Key},\ (ListOfNbrs)SigA\ )encPK_B$
4. $B \longleftarrow A : (NonceA + 1)encDH_{Key}$

Now will look at each step and discuss the achievements of this 4-way-handshake.

1. The first message is for information which identity (A) shall be authenticated (by B). B checks whether the MAC address and the Public Key of the sender is matching the MAC address and the hash of the Public Key in RegCert. B also checks the IP address and the validity of the certificate.

2. B sends its certificates in return and additionally a message encrypted with A's Public Key containing Bs part of the Diffie-Hellman Key, a number-used-once(Nonce) and a signed list of the Node IDs of its neighbors. Sending a list of its neighbors prevents an attacker from spoofing B in another location. The signature proves B to be the creator of this message.

3. A proves the possession of the Private Key belonging to the Public Key identified in the certificates of A by decrypting NonceB (and sending it back). Now additionally A affirms its neighbor relations by sending a signed list of the Node IDs of its neighbors. A sends its part of the Diffie-Hellman Key. A calculates their shared Diffie-Hellman Key. NonceA is encrypted using the new Diffie Hellman Key and the whole message is encrypted using the Public Key of B

4. The last message proves B's possession of the Private Key corresponding to the Public Key in B's certificates (and so the identity of B). NonceA+1 can only be calculated if B decrypted NonceA using the shared Diffie-Hellman Key of A and B.

Therefore after this mutual authentication the identities of both nodes are approved and a symmetric key was exchanged for further usage. This mutual authentication was done during link establishment with a selection of nodes from the direct neighborhood, so our new node is now fully integrated into the network and ready to participate in routing.

6.3 AntSec - Secured AntNet

6.3.1 Design Goals and Assumptions

In the last section we described how a new node in the network establishes links to a selection of direct neighbors performing mutual authentication and exchanging a symmetric "session" key guaranteeing Perfect Forward Secrecy (PFS) [IETF00]. In the following step, the new node shall participate in routing and is free to use the network as full member. We propose a routing algorithm called AntSec, which is based on AntNet1.1 and supplemented by a watchdog mechanism.

In [ZE02] Weilin Zhong and David Evans have explored some of the security issues for stigmergic systems. Observations made in their work will be considered. They point out that monitoring, fabricating, modifying, and deleting of ants is possible. Replay attacks are possible as well. As this is a valid observation for any routing protocol without special security mechanisms, we focus on the AntSec specific attacks. Zhong and Evans argue that modifications on the metric describing the quality of each link can be done without being recognized. But since a malicious node can only lie about properties of the link between itself and its successor, this can be counter checked. The authors suggest, that each node processes only BANTs whose corresponding FANTs have been processed. We add, that in this step the entries of the BANT should be verified, too. The work of Zhong and Evans is the first in literature to study security vulnerabilities of stigmergic systems. Our work has to be seen in this context.

The design goals for our new routing algorithm were the following points:

1. Correct routing functionality: loop free routing, up to date routes.

2. Verify authenticity and authorization of the source and destination node to all nodes in a route.

3. Verify authenticity and authorization of each intermediate node to their neighboring nodes.

4. Forging of routing messages in the name of other nodes shall have no effect on routing.

5. Invalid manipulations on routing messages shall be detected.

6. The routing algorithm shall detect data leaks on a route in case the watchdog mechanism fails or is tricked.

7. If possible avoid the usage of asymmetric cryptography due to its calculation complexity.

8. Routing overhead shall be kept as low as possible, unnecessary transmissions have to be avoided.

9. If possible decisions shall be based on local information.

In the following subsection we introduce our routing algorithm and provide a detailed review of the list of attacks it prevents, additionally we describe how active attacks can be detected. We discuss the necessity of using asymmetric cryptography in some steps of our algorithm. The usage of non-local components will be illustrated. Our work enfolds the implementation and simulation of AntSec, the performance is compared to the standard routing algorithm DSR.

We have to make the following assumptions:

- Bidirectional links - if A is in the transmission range of B, then B is in the transmission range of A. This is essential for the communication between two nodes.

- The authorization of each node is checked locally, no unauthorized node is allowed to be served. After link establishment mutual authentication between direct neighbors is indispensable.

- For authenticating distant nodes we assume a feasible way to link the IP address of a node to its Public Key. This can be done by a trusted instance providing a look-up table or by certificates issued by a trusted third party approving the link of the IP address and the corresponding Public Key (RegCert).

- We assume that the devices corresponding to the nodes in the network are capable of performing common cryptographic calculations (symmetric and asymmetric cryptography and hashing).

- We present two watchdog mechanisms. For the first watchdog mechanism, we additionally assume that the links between neighboring nodes are not encrypted.

- For the second watchdog mechanism, called WatchAnt, we assume that no loss of packets occur on physical and data link layer. Or even when loss occurs, that there exist a mechanism for retransmission. So that on routing layer all packets that have been transmitted are received.

6.3.2 AntSec - Routing Messages

In this subsection we present a stigmergy based proactive routing algorithm called AntSec. Each node maintains a routing table keeping a list of its neighboring nodes and probabilities to use them as next hop to all known destination nodes. So, for each combination of destination node and neighboring node a routing probability is maintained denoting the probability of choosing this neighboring node as the next hop for a packet destined to the destination node.

We have three different kinds of routing messages which are used for route establishment and maintenance:

6.3.2.1 Discovery Forward Ant

Discovery Forward Ants (DFANT) are periodically sent to random destinations to find and establish new routes. They contain the registration certificate authenticating the source node and a path list containing all visited nodes. They are forwarded in a way that allows them to find new and yet undiscovered routes.

Optimized version (38 bytes + n * 8 bytes):

- AntID.RegCert: Registration Certificate of the source node proving its authorization to participate in the network and giving information about the IP address(4 bytes), MAC address(4 bytes), NodeID(2 bytes) and Public Key (20 bytes, hash) of the node. The certificate has a limited lifetime (4 bytes). The total size is of the Registration Certificate is 34 bytes.
- AntID.timestamp: Timestamp (4 Bytes) denoting the generation time of the DFANT. The AntID combined with the network address of the destination node is used as unique identifier of the Ant.
- Flag, signaling the need of the Registration Certificate of the destination node (1 Bit).
- Flag, signaling the need of the Public Key of the destination node (1 Bit).
- List of visited nodes: each list element contains the network address of the node on the route (4 bytes) and QoS-information for the link from its predecessor (4 bytes for the queue length).

Current version (60 bytes + n * 9 bytes):

- AntID.RegCert: Registration Certificate of the source node proving its authorization to participate in the network and providing information about the IP address(4

bytes), MAC address(4 bytes), NodeID(2 bytes) and Public Key (20 bytes, hash) of the node. The certificate has a limited lifetime (4 bytes) and carries a signature of a trusted certificate authority (20 bytes). The total size of the Registration Certificate is 54 bytes.

- AntID.timestamp: Timestamp (4 bytes) denoting the generation time of the DFANT. The AntID combined with the network address of the destination node is used as unique identifier of the Ant.
- Flag, signaling the need of the Registration Certificate of the destination node (1 Bit).
- Flag, signaling the need of the Public Key of the destination node (1 Bit). This flag should not be set without setting the previous flag.
- List of visited nodes: each list element contains the network address of the node on the route (4 bytes) and QoS-information for the link from its predecessor (4 bytes for the queue length).

6.3.2.2 Maintenance Forward Ant

Maintenance Forward Ants are periodically sent to keep the current routes between a source node and a destination node active. These routing messages are used to reinforce the active routes and to adapt the routing probabilities to the current state of the topology.

Current version (26 bytes + n * 9 bytes):

- Hash of AntID.RegCert (20 bytes): As the Registration Certificate of the source node has already been passed to each node on the route during route establishment it is sufficient to send an unique identifier of it.
- AntID.timestamp: Timestamp (4 bytes) denoting the generation time of the MFANT. The AntID combined with the network address of the destination node is used as unique identifier of the Ant.
- Total traffic for this route (2 bytes): This field is set by the source node and denotes the amount of data (in bytes or packets) which has been sent during this session to the destination node (for further details subsection see 6.5).
- List of visited nodes, each list element contains the network address of the nodes on the route (4 bytes) and QoS-information for the link from its predecessor (4 bytes for the queue length). Additionally, each node on the path specifies the fraction

(1 byte) of traffic (in comparison to the total traffic for this route), which has been received from the current predecessor in the visited nodes list.

6.3.2.3 Backward Ant

Backward ants (BANT) are sent by destination nodes upon receiving forward ants. Critical parts of the forward ant are signed and added to the BANT. Additionally the Registration Certificate and Public Key of the destination node is added if the corresponding flags in the forward ants are set.

Current version (28 bytes + n * 9 bytes + (0 or 54 or 182)) = (28/82/210 + n*9):

- Hash of AntID.RegCert of the source node (20 bytes).
- AntID.timestamp of the forward ant (4 bytes).
- Total traffic for this route indicated by the source node (2 bytes).
- Complete list of visited nodes: Each list element contains the network address of the node on the route (4 bytes), QoS-information for the link from its predecessor (4 bytes for the queue length) and the fraction (1 byte) of traffic (in compare to the total traffic for this route) which has been received from the current predecessor in the visited nodes list.
- Registration Certificate of the destination node, if the flag requesting it was set (0 or 54 bytes).
- Public Key of the destination node, if the flag requesting it was set (0 or 128 bytes (= 1024 bits)), the Public Key is only sent in combination with the Registration Certificate (128+54 = 182 bytes).

6.3.3 AntSec - Route Establishment

As AntSec is a proactive routing algorithm: routes are established before they are needed for data transmission. To build up routes, each node (S) periodically sends Discovery Forward Ants (DFANTs) to random destinations (D). To quicken the learning period, in the beginning neighbors in the 2-hop-neighborhood have a higher probability to be chosen as destination for these FANTs.

The DFANT messages contain the Registration Certificate of the source node, a current timestamp, flags set according to whether the certificates of the destination node are in the Certificate Repository and an empty visited nodes list. As first hop of the message, either a node is picked from the source node's neighborhood which was not chosen before

for this route, or, if there are none such, then a neighbor according to the probabilistic routing table is chosen.

Upon receiving a Discovery Forward Ant following processing is done:

- First the Registration Certificate in the AntID is checked for validity. If the check fails, the packet is dropped and the previous hop gets a bad rating for forwarding invalid routing messages instead of dropping them. But if the check succeeds, the Registration Certificate is added to the Certificate Repository, which lists all known certificates.

- Then is checked whether the current node is already listed on the list of visited nodes. In this case, the packet is dropped and the previous hop node's reputation is decreased, since no packet is allowed to be forwarded to a node, which is already in the visited nodes list.

- In order to detect invalid manipulations of the TTL field of the IP-Header, it is also checked whether the TTL field and the number of nodes listed in the visited nodes list match.

- Having done these security checks, the next step consists of checking whether the message is addressed to the current node. If not, and the current node is just an intermediate node, then the following is done:

  – A hash of immutable parts of the DFANT is generated enfolding the complete AntID, both the source node's and the destination nodes network address and the contents of the field denoting the total traffic sent for this route. Each node maintains a list of Forwarding Ants having processed. At the current step of processing this list is examined to see whether this message has already been processed before. Therefore the hash of immutable parts of the Forward Ant mentioned above and the entries in the list of already processed FANTs is inserted.

    In this case the visited node's list does not contain the current node's network address. This means that some node on the route has maliciously modified the Forward Ant. Being aware of an attack, the message is still processed as normal, because this is not harmful. But if the FANT has not been processed before, the hash of immutable parts of the Forward Ant is added to the list of already processed FANTs.

  – The next step is to check whether the Time To Live (TTL) of the packet is in a state signaling that the message is not needed to be forwarded any more. Then

the packet is dropped.

- Reaching this point means that the current intermediate node has to forward the Forward ANT. The current node applies its own data to the path-history. It adds its network address and the current length of its data queue.
- The current node has to check, whether the Public Key of the destination node is known to it, if not, the current node has to set the CertOfDestNeeded_Flag. The same is done for the Registration Certificate of the destination node.
- Afterwards, a hash is generated containing the immutable parts of the FANT and the path-history up to this current node. This digest is stored locally at the current node, since on further processing of the DFANT these parts of the message must not be changed. Additionally, the state of the flags is stored, to know upon receipt of an answer whether the RegCert or the Public Key of the destination node has been requested or not.
- The next hop of the modified message is chosen according to the current node's routing table, except there are some direct neighbors, which have not been tested for their ability to route packets to the given destination node, those neighbors are preferred. For choosing the next hop normally the probabilities in the routing table are used. But with a small probability uniform distribution is used. During this phase some neighbors are removed from the pool of possible next hops. The path-history is checked, to avoid sending the packet to a node it has already passed and thereby generating a loop. The current node shall avoid sending the packet into a back-end, i.e. to a node having only one neighbor, the current node itself. It is neither allowed to forward the Forward Ant to a node whose reputation signals malicious behavior.

  These cautious checks have to be considered, because the dropping of routing packets is strictly forbidden (only packets from selfish/malicious nodes may be dropped, or when reaching the maximum hop count).
- The adapted FANT is unicasted to the chosen neighbor.
- Note that the Forward Ant is not secured against attacks on authenticity (spoofing) and integrity (modifications). An attack can perform any modifications on the Forward Ant without being detected. We show in the description of the processing of BANTs and in Chapter 7 that these attacks have no negative effect though.

- If a FANT reaches the destination, it is turned into a Backward Ant (BANT) and the path-history is signed so that it cannot be altered anymore. The FantID is kept

as BantID, the RegCert of the source is removed and the RegCert of D is added if the flag CertOfDestNeeded was set. Additionally the IdentCert is added if the flag IdentCertOfDestNeeded was set. The BANT is sent back to the node from which the corresponding FANT came. Each intermediate node on the route back to the source node is processing the BANT in this way:

- Each node receiving the BANT checks whether the hash on the immutable parts of the BANT is known and the path to S stored in the BANT matches with the path-history in the current node's memory. Only if this test is positive the BANT will be processed. This is done to prevent the processing of fabricated (unknown FANT-ID) or modified (false path-history or modified fields which must not have changed) FANTs. Any invalid modification of the FANT result in a BANT which cannot be associated to its corresponding FANT. Such BANTs are dropped. We see that FANTs, which have been modified invalidly, are detected here.
- In the next step, the active node checks the signature of D. If it is invalid, the packet is dropped and the previous hop is rated bad for forwarding invalid BANTs.
- If D added its Registration Certificate, this is checked too. The same is done for the Public Key of the destination node, if it is contained in the BANT. Its hash has to match with the hash listed in the Registration Certificate. In all these cases the packet is dropped upon a failure, as the BANT cannot be processed with false information.

  If either the Registration Certificate or the Public Key is missing, although they were requested the BANT is dropped. On a failure during this step, the previous node's reputation is decreased for forwarding a BANT with invalid or missing (previous node did not check for validity or existence) contents to the current node.
- Having reached this point means that the BANT has been sent in reaction to a valid FANT. The visited nodes list of this FANT has not been modified in the entries up to this current node. The Registration Certificate and Public Key of the destination node are available and the signature of the BANT is correct. So the information of this BANT can be used for the routing table.

  As the visited nodes list contains not only the network addresses of the nodes on the route but also the size of the queues, this information is used to determine the quality of the route. Let N_i be the current node on the route S, N_1, N_2, ..., N_i, N_{i+1},...D. Then the sum of all queue lengths from the entries of

N_{i+1}, N_{i+1}, ..., D defines the quality value (Q_{val}) of the route to D. Q_{val} is set in relation to the best quality value measured for this destination node up to now. Based upon this relation the probability to chose N_{i+1} as next hop to D is increased. The closer Q_{val} to the best quality value measured the higher the increase. The probabilities to choose other neighbors are decreased proportionally to their current probabilities, so that the sum for all neighbors is 1 again.

This reinforcement of the path the FANT took models the stigmergy behavior of the routing algorithm, as following messages for this route are influenced to use this link more often.

- The only exception to the rule of assigning probabilities to the next hops is when the destination node is a neighboring node. In this case the probability to use the destination node as next hop is set to 1 and the probabilities to use other neighbors is set to 0.
- In case of being the destination of the BANT message there is nothing more to do. But, if the current node is just an intermediate node, the BANT has to be forwarded to the next hop denoted in the path-history like in source routing algorithms.

- It may be that a FANT does not reach the destination node as FANTs are forwarded to probabilistically chosen nodes. In this case no route is established. But as Discovery Forward Ants are sent regularly some DFANT will find a route (if one exists) after a certain amount of attempts.

6.3.4 AntSec - Route Maintenance

Route Maintenance is only done for routes which are actually used. Although route discovery is still performed for all known destination nodes, routes in use should be updated more frequently. Therefore Maintenance Forward Ants (MFANTs) are used which are smaller in size than Discovery Forward Ants. This smaller routing message size and the need for more frequent route updates justify the introduction of this additional routing message type. The processing of both kinds of Forward Ants differs only in a few points:

- Maintenance Forward Ants are also generated periodically, but they are only destined to nodes to which data is actually sent.

- As these routes have already been established, MFANTs do not need to exchange Registration Certificates again, so the MFANTs are smaller in size.

- Intermediate nodes receiving a MFANT perform the same security checks (except concerning certificates) as for DFANTs, but act differently when choosing the next hop of the MFANT. During route maintenance no exploration of new routes is done. This means that MFANTs are forwarded strictly according to the routing table.
- MFANTs reaching their destination nodes are processed like DFANTs: a Backward Ant is generated. Only if the leak detection mechanism is used, slightly different procedure is chosen. In this case the route is examined for leaks resulting from colluding malicious nodes. (for further details see 6.5).

6.3.5 AntSec - Route Error

AntSec is a probabilistic multipath routing algorithm in which each node locally decides which neighbor to pick as next hop for a packet. Upon occurrence of link failures, there are two cases: the current node has an alternative link or not. If there is an alternative link, the probability for the broken link is set to 0 and the probabilities for all other links are increased proportional to their current probabilities. If no other link exists (except the link the packet was received from) then the probability for the broken link is still set to 0 and for the probability for using the last remaining neighbor is increased to 1.

As the list of current neighboring nodes is advertised in the NCFG messages, the only remaining neighbor realizes that the current node has no other neighbor and therefore no messages are routed over this current node anymore (since dead-ends in routing are prohibited). So, the management of broken links does not need additional traffic overhead. The nodes solve the problem locally by adapting the routing probabilities.

6.3.6 AntSec - Routing of Data Messages

The actual payload is routed from the source node to the destination node in a distributed manner based on local decisions. On the arrival of a data packet at an intermediate node the next hop is chosen from a pool of valid neighbors according to the probabilities listed in the route table of the current node. The sum of the probabilities of all neighboring nodes to be chosen may be smaller than 1 as some neighbors may be excluded from the pool of acceptable next hops. In this case the probabilities used are normalized, but the original probabilities are not affected.

Following neighbors are excluded from the pool of valid next hops:

- The source node of the data packet.
- The neighbor from which the data packet was received. This is done to prevent an immediate returning of the packet.

- Neighbors assumed to be malicious. Neighbors are assumed malicious, if their reputation lays below a specific threshold value (see Section 6.6).

In the case that no valid neighbor remains after this procedure the constraints to the pool of valid next hops are canceled. Now neighbors from the list above are allowed to be next hops. We consider it reasonable to give the data packet a try to reach the destination, even if the next hop may be a bad choice, instead of dropping it.

Note that loops may occur on the route of data packets, as it cannot be checked which route a data packet has taken up to now. But the probability for this case is rather small as it is not the optimal route. Ants leaving the optimal route result in a worse quality value and have therefore a lower probability to be chosen.

6.4 Mechanisms to Detect Misbehavior

In the previous section AntSec was introduced, which guarantees the integrity and authenticity of routing messages. AntSec uses asymmetric cryptography to secure the BANTs and a crosscheck mechanism to verify the authenticity and integrity of FANTs, which are not secured with cryptographic mechanisms. Route establishment and maintenance cannot be interfered with forged, spoofed, replayed or modified routing messages. And since routing adapts to the topology due to its probabilistic behavior and multipath routing is used additionally, false route errors have only a small effect on routing.

In this section we introduce two mechanisms to detect misbehavior in forwarding routing and data packets. Due to the differences in the assumptions we cannot unify the mechanisms. The first mechanism is called Watchdog and is inspired by the mechanisms introduced in Section 5.2, assuming unencrypted links. The second mechanism is called WatchAnt and has to cope with encrypted links, which complicates the monitoring of the behavior of neighboring nodes.

These misbehavior detection mechanisms are to be used to control whether the neighboring nodes are forwarding packets correctly or not. The following kinds of attacks shall be detected:

- Routing messages must be forwarded and must not be dropped (only exception: time to live limit reached). As we have seen in the previous section, there are various cases in which routing messages are dropped. These cases all occur only when the previous hop of the routing message has made a mistake, therefore it is rated bad.

- Some fields in routing messages have to be left unmodified during the forwarding

from the source to the destination node. Although this is additionally checked by AntSec itself, the watchdog mechanism can do this, too.

- Data packets must not be modified.
- Data packets must not be dropped but have to be forwarded.
- Misbehaving nodes shall be detected and identified, so that various punishment methods can be applied.

Both mechanisms which are introduced in the next two subsections fulfill these tasks, but they differ in the assumptions they make.

6.4.1 Watchdog: Overhearing Mechanism for Unencrypted Links

For the first mechanism, Watchdog, we make the following assumptions:

1. Communication is done over a shared wireless medium
2. Communication channels are bi-directional (Each node is capable of receiving and sending)
3. The transmission ranges for all nodes are similar, so for any node A that is in the transmission range of a neighboring node B counts, that B is in the transmission range of A too.
4. Nodes can spy on the shared medium and overhear the transmissions of neighboring nodes (promiscuous mode).
5. The communication links between two neighboring nodes are not encrypted.

Whereas assumptions 1, 2 and 3 are also stated for the next detection mechanism, assumptions 4 and 5 limit the application field of this mechanism. In literature these 5 assumptions are made often (see all overhearing mechanisms in Section 5.2).

6.4.1.1 Watchdog Mechanism

The main idea of the mechanism is to store an identifier for every packet forwarded to a neighbor and listen to it afterwards, checking whether this identifier is recognized among the transmissions of this neighboring node in the future.

Therefore each node maintains a list for packets that have been forwarded to each of its neighboring nodes. When a new packet arrives at node A and is not destined to A, A processes it according to its routing algorithm (AntSec) and chooses a next hop (B). If

B is not the destination of the packet, A stores the packet and a timestamp in its list of packets forwarded to B.

On every transmission of B that is overheard by A, A checks whether this currently transmitted packet is matching one of the packets that B has to forward (and which has previously been forwarded from A to B). For routing packets the immutable parts are compared and it is also checked whether the mutable parts contain valid entries (TTL decreased by only one, no certificate requesting flags reseted). Data packets are checked for having been modified invalidly or not.

For optimization reasons one can alternatively store hashes of the immutable parts of a packet instead of the whole packet. But in this case one cannot check whether the mutable parts of the FANTs (TTL, flags) have been modified correctly. But this is considered to be not harmful since the TTL field is checked during the processing of FANTs, and invalid flags are detected if the returning BANT does not contain the certificates. It is then assumed that the previous hop is the malicious node, as it forwarded the BANT even though a flag was set incorrectly.

Three cases may occur on monitoring the behavior of the neighbor:

- The packet is recognized to be forwarded correctly. In these cases the reputation of the monitored node is increased by the monitoring node.

- The packet to forward is recognized but it has been modified invalidly. This can only occur during the forwarding of routing messages, as a modification of a data packet cannot be detected as such (it may be a valid forwarding of a packet received from a third node). Routing messages can be identified uniquely by the AntID and the destination address. On detection of invalid modifications of routing messages the reputation of the neighboring node is decreased. When applying the optimization and storing only hashes, this misbehavior cannot be detected.

- Not forwarded or invalidly modified packets are most commonly detected by a time-out signaling, that the packet, which had to be forwarded, has not been recognized during a specific time window. The failed recognition can have several reasons. Besides invalid modifications and dropping of the packet also problems on the physical layer may be a reason (see Subsection 5.2.1). Nevertheless the monitored node's reputation is decreased by the monitoring node.

With this simple mechanism we can detect whether a neighboring node forwards packets correctly or not. One limitation is that in the case of not forwarding the packet, we cannot clearly say whether the packet was dropped, manipulated or not perceived due to

collisions. But as the decrease of the reputation, when no correct behavior was recognized, is only one reputation among many, one false rating does not have that big effect. But when a node is constantly malicious these bad reputations accumulate and it is detected as being malicious (for further details see Section 6.6).

6.4.2 WatchAnt: Challenge-Response Mechanism for Encrypted Links

In this subsection we introduce WatchAnt, a mechanism fulfilling the task of detecting misbehavior at forwarding under the following assumptions.

6.4.2.1 Assumptions for WatchAnt

1. Communication is done over a shared wireless medium
2. Communication channels are bi-directional (Each node is capable of receiving and sending)
3. The transmission ranges for all nodes are equal, so for any node A that is in the transmission range of a neighboring node B counts, that B is in the transmission range of A, too.
4. For each packet node N_2 receives, it has to know from which neighbor (N_1) this packet was received and from which node N_1 has received the packet (N_0).
5. A mechanism exist to broadcast a message so that all neighboring nodes receive the message (collision free and unencrypted).
6. A mathematical function *hashsum* with following properties:
   - Let B = 0,1, $H = B^{160}$ and $h_i \in H$ for $i \in \mathbb{N}$.
   - hashsum : $\{H\}^* \longmapsto H$
   - Associativity: hashsum(hashsum(h_1, h_2), h_3) = hashsum(h_1,hashsum(h_2, h_3))
   - Commutativity: hashsum(h_1, h_2) = hashsum(h_2, h_1)
   - (Optional) One way function: Let X = hashsum(h_1,h_2) , then for given X and h_1 one cannot calculate y so that X = hashsum(h_1, y) with feasible effort.

When comparing the assumptions, we see that in this second case encrypted links are permitted. This complicates the task of detecting misbehavior in forwarding, as one cannot directly check whether the neighbor is re-transmitting the packet or not.

To our knowledge, there is no solution for the detection of forwarding misbehavior in wireless distributed scenarios with encrypted links. WatchAnt is the first solution suggested to cope with this problem.

We give a brief introduction to our solution and discuss the assumptions stated afterwards. WatchAnt is a challenge-response mechanism in which each node (A) periodically asks its neighbors (for example B) to report where the packets, which have previously been forwarded from A to B, have been forwarded to. The asked node (B) answers with a list, reporting which packets have been forwarded to which of its neighbors. Now A checks whether the list contains all packets that had to be reported and whether the neighbors listed are authorized (their Registration Certificates are in A's Certificate Repository). Additionally, each neighbor (C) of B which was listed in the answer checks whether the packets reported of having been sent to C have been actually received. The security in this mechanism arises from the fact that B cannot lie to both A and C pretending correct behavior.

Two of the assumptions need further discussion. Assumptions 4 and 5. First, our solution requires that each packet carries the information about the last hop before it was forwarded. Let $N_0 \longrightarrow N_1 \longrightarrow N_2$ be the part of a route. When N_1 forwards a packet to N_2 then N_2 has to know that this packet has previously been forwarded from N_0 to N_1. This is important for our mechanism which we demonstrate with the following example and Figure 6.2:

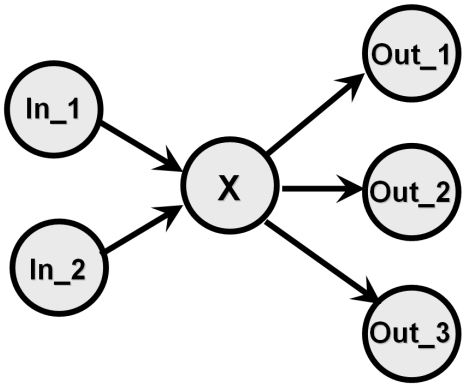


Figure 6.2: Observation radius of a node when using WatchAnt

Let In_1 and In_2 be the node from which a Node X receives packets. X forwards these packets only to its neighbors Out_a, Out_b and Out_c. As packets arrive from In_0 and In_1 the node X forwards them according to its (probabilistic) routing table to Out_a or

Out_b, Out_c is not used in this case. This may have several reasons as being a dead end or providing low quality and therefore having only a small probability to be utilized.

Now In_1 asks X to report where the last n packets were forwarded to. As X is benign and has forwarded everything correctly it computes the answer and sends this answer to all of its neighbors. In_2 and Out_c ignore this report as they are not listed in it. Out_a (and Out_b) has to check whether this report is correct, i.e. it only lists packets which were actually forwarded by X to Out_a (and Out_b respectively). Now our assumption number 4 becomes relevant. As X has forwarded packets both from In_1 and from In_2 to Out_a (and Out_b respectively) the listed nodes have to be able to differentiate which packets had been forwarded from In_1 over X to them. This differentiation has to be done to be sure that the packets are associated to the correct route.

Assumption 5 is motivated to assure that the asked node sends its report to all relevant nodes (requester and listed nodes). An alternative to collision-free and unencrypted broadcasting would require the excessive use of asymmetric cryptography. The asked node would have to prove to the asking node by digital signatures of the listed nodes, that the report has been sent to listed node, too. This alternative is not feasible; Hence a mechanism for collision-free and unencrypted broadcasting has to be assumed.

6.4.3 Adapting WatchAnt to IEEE 802.16 Mesh Mode

To fulfill the requirements of assumption 4 we could extend the packet header on routing layer or on MAC layer. As IPv4 [IETF81] is commonly used on the routing layer, for any modifications to it one can assume that it will not be used. This is for sure as even the adoption of its successor IPv6 [IETF98] is done very reservedly. So the additional information of the previous hop of the packet has to be transmitted using the MAC layer. As the IEEE 802.16 standard is quite young there is still room for improvements and extensions.

We suggest the following solution:

In 6.3.2.2.5 of the standard [ICS04] the Mesh subheader is introduced, which always follows the generic MAC header (specified in 6.3.2.2). The Mesh subheader only contains one information: *Xmt Node Id* (16 Bits), denoting the node ID of the transmitting node.

We suggest extending the Mesh subheader as follows: Besides the *Xmt Node Id* (16 Bit) it shall contain additionally a field for *Prev Hop Node ID* (16 Bits). The Node ID of the previous hop can be resolved to a network address or MAC address using the Registration Certificate of the previous hop from the Certificate Repository. If the certificate is not available (which may occur on entering a network) a route request can be started to receive it. In the time when the certificate is not available no answers can be counter-

checked. This time is rather short as AntSec prefers to establish routes to nodes in the two-hop-neighborhood.

We continue our example from the beginning. X denotes the Node ID of the previous hop (In_1 or In_2) in *Prev Hop Node ID* for each packet forwarded to Out_a, Out_b or Out_c.

Assumption 5 requires a mechanism to broadcast collision-free and unencrypted. This is no real limitation when applying the mechanism to IEEE 802.16 Mesh Mode as the control subframe is transmitted collision free and unencrypted. So IEEE 802.16 provides a mechanism as required. It may seem contradictory, since in 4 we have argued that routing protocols using broadcasting are not applicable to IEEE 802.16. This is still true, because the broadcasting opportunities provided by using the control subframe are still rare and have constraints on the size of the message that has to be broadcasted. They can be used to rarely send a small WatchAnt Reply but are not suitable to be used for transmitting large messages (large due to security requirements) in a frequent manner (as required for routing).

We suggest the following solution:

- MSH-NCFG messages (specified in 6.3.2.3.35 in the standard [ICS04]) have an extra field for $MSCH - NCFG_embedded_data()$.

- $NCFG_embedded_data()$ is specified in 6.3.2.3.35.3. It contains a type identifier (4 bits), the length (8 bits) of the current Embedded Information Entry (variable length) and a flag ($Extended\ embedded_data$, 1 bit) signaling whether the current Embedded Information Entry (variable length) is followed by another one.

- As the type identifier provides space for up to 16 different types, from which only 6 are specified, we add an additional type: *WatchAnt Information Entry*.

- We introduce two new Information Entries which can be sent as embedded data in NCFG messages:

  - WatchAnt Request (22 bytes): Contains the Node ID of the neighbor to check (16 Bits) and the hash of the relevant packet (160 Bits).

  - WatchAnt Reply ($3 + n \cdot 23$ bytes): Contains the Node ID of the requester (16 bits), the number of report entries (8 bits) and the report entries. A report entry contains the Node ID of the listed node (16 Bit), the number of packets sent to it (8 bit) and the accumulated hashes of the packets sent to this node (160 bits).

6.4.4 The WatchAnt Mechanism

After having discussed the assumptions we made for our mechanism. We give now a detailed introduction to how WatchAnt works. Each node maintains two lists for each neighbor: a list of incoming packets (coming from the neighbor) and a list of leaving packets (sent to the neighbor). In the further description we call these lists inList and outList. Further we denote the hash of immutable parts of a packet P with hash(P). Data packets and Backward Ants are hashed completely whereas the flags and the visited nodes' list is left out when hashing Forward Ants.

An entry in inList contains hash(P) for the corresponding packet P, the Node ID of the node the packet was received from and the Node ID of the previous hop. Additionally, a field where the next hop of this packet can be entered is provided. An entry in outList contains hash(P) and the Node ID of the node to which this packet was sent to.

InList and outList only contain a finite number of entries. Either a maximum number of entries can be set or a timeout mechanism may be implemented. Both result in the same behavior. Setting a maximum number of entries has some advantages, one can be sure to have a minimum number of entries (after an initialization time) and the memory usage is constant.

The WatchAnt mechanism works as followed:

- On generation of a packet each node sets its own Node ID in the *Prev Hop Node ID* of the Mesh subheader. Upon forwarding of a packet, the Node ID of the previous node is set in this field.

- When a node (A) sends a packet P to a neighbor (B) the sender A stores hash(P) and the Node ID of B in its outList. This is not done if the packet is destined to B, because B does not have to forward the packet in this case.

- On receiving a packet a node (B) stores the following information in the inList: hash(P), the Node ID of the previous hop (A) and the Node ID of the hop before the previous hop. Its Node ID is denoted in *Prev Hop Node ID* of the Mesh subheader.

- Upon forwarding a packet to a neighbor (C) the current node (B) stores the node ID of the next hop (C) in the entry corresponding to this packet in the inList.

- Each node (A) periodically sends a WatchAnt Request to a randomly chosen neighbor (B). The WatchAnt Request contains the Node ID of the challenged neighbor and the hash of a packet P (hash(P)) which was previously sent to this neighbor.

  This hash(P) is to be interpreted as a marking pointing in the inList. The question the requesting node A addresses to its challenged neighbor B is: Where did you

forward the n packets to, which were sent to you before the packet corresponding to hash(P).

- The challenged node B investigates its inList and filters out the entries which are associated to the n packets which were received from the challenging node A before the marking packet P.
- B separates these entries according to the next hops the packets were forwarded to and a generates a WatchAnt reply.
- The WatchAnt reply contains the number m of referenced neighbors, their IDs (j), the number of packets forwarded to them (j) and hashsum(P_i1,..,P_ij), the hashsum of all packets forwarded to each referenced neighbor.
- On receiving the answer the requesting node calculates hashsum(hashsum(Nbr1), hashsum(Nbr2), ..., hashsum(Nbrj)), the result is called $hash_{answer}$. Additionally the requesting node calculates the hashsum for all n packets sent to the challenged neighbor before the packet P. The hashsum of these packets is called $hashsum_{request}$.
- If $hash_{answer}$ is identical to $hashsum_{request}$ and the node IDs of the referenced nodes are all known (authenticated by their Registration Certificate), then the challenged neighbor has passed the test, and its reputation is increased. Upon a failure, the reputation is decreased significantly.
- On receiving of a WatchAnt Reply each node checks whether it is listed as a reference node or not. Being listed, the hashsum for the last j packets coming from the requesting node (A) over the challenging node (B) to them is calculated. If the calculated hashsum matches the hash listed in the WatchAnt Reply, the reputation of the challenged node is increased, otherwise it is decreased.

As we have seen benign nodes can give correct reports on their forwarding behavior. A malicious node dropping, manipulating or forging packets will be detected as its reports will not match the estimated traffic flow at the checking nodes.

A malicious node can follow three strategies. It can try to be honest, pretend to be good towards the requesting node and pretending to be good towards the nodes it references.

Being honest leads to false or missing hashes which result in mismatching Hasheem's. This strategy reveals its misbehavior towards both, the requesting and the referenced nodes. Pretending to be benign towards the requesting node can be done by answering

with Hashim's based on hashes of packets received from the requesting node. As these packets are modified before forwarding or dropped the neighboring nodes detect the mismatch in the reply. But when the malicious node answers with hashes based on modified packets the hashes may match at the referenced nodes but not during the check of the requesting node.

Furthermore based on assumption 6 malicious nodes cannot forge packets with the property that the hashsum of these forged packets match the hashes of the modified or dropped packets. Anyhow the misbehavior is detected.

6.5 Leak Detection Mechanism

As our review of watchdog and security mechanisms for MANETs has shown colluding malicious nodes are a critical threat to wireless networks. The main idea of a watchdog mechanism is to detect misbehaving neighbors, which do not forward packets as they should. But when malicious nodes are cooperating and tolerate the malicious behavior of each other detection is difficult.

We give following motivation for the investigation of a leak detection mechanism detecting colluding malicious nodes which are dropping packets of others. Consider following scenario in which a common watchdog mechanism (as presented in Section 5.2) fails:

- S is sending packets over A to D. S recognizes that A is forwarding all packets to B.
- A forwards all packets received from benign nodes (in this case to B).
- B drops all packets which were not generated by malicious nodes but received from a colluding malicious node (A)
- A detects the misbehavior, but as A and B are colluding A does not react.
- All nodes participate in forwarding and processing routing messages (AntSec does not detect any failure).

We see that securing the routing algorithms designated to prevent forging, invalidly modifying and dropping of routing messages, is not sufficient to detect this misbehavior. Watchdog mechanisms lead to a detection of misbehavior, but in this scenario the detecting node is colluding with the dropping node. Therefore the security framework needs a leak detection mechanism for routes.

We now present an extension to AntSec which fulfills this demanding task. The main idea of the leak detection mechanism is that the destination node in a route builds up a virtually graph modelings the various paths from the source node to the destination node. Based on traffic information transported with Maintenance's Forward Ants the destination node learns the ratio of incoming and outgoing traffic corresponding to this route for each participating node. By basic graph theory traffic leaks can easily be identified. Paths containing malicious nodes are not enforced during route maintenance in AntSec. The detailed description of the leak detection mechanism is as follows:

- Each source node in a route maintains a traffic counter denoting the amount of traffic in bytes which has been sent to the destination node.
- Maintenance Forward Ants (see Subsection 6.3.4) have two fields which are relevant for this task: emftotal_traffic_for_this_route (2 bytes) and for each visited nodes a field denoting the *fraction_of_traffic* (1 byte) in comparison to the total traffic sent by the source node.

  The 2 bytes of the *total_traffic_for_this_route* field are separated into a 2 bit part denoting the magnitude of the amount of data and a 14 bit enfolding value part, which describes the amount of data being sent according to the presented magnitude. We introduce following magnitude-codes: 00 = bytes, 01 = kilobyte, 10 = megabyte, 11 = gigabyte.
- As the MFANT sent from source node S to destination node D is forwarded from an intermediate node A to its neighboring node B, B calculates the following: B has to append its own node info to the visited nodes list. In this entry the *fraction_of_traffic* field has to be set. B calculates the amount of traffic received for the S-D route from its neighbor A. This amount of traffic is set in relation to the total traffic for this route (denoted in the *total_traffic_for_this_route* field of the MFANT). This relation is representing the fraction of traffic for this route sent from A to B. B sets this in the *fraction_of_traffic* of its visited nodes entry. As the *fraction_of_traffic* field has the size of one byte the granularity of the fraction (from 0 to 100) is $100/255 = 0.4$.
- The destination node collects the traffic information from MFANTs and virtually creates a graph. Each vertex represents a node participating in a route from S to D. The directed edges between two nodes X and Y represent a MFANT having first passed X then Y on its path from S to D. Additionally, the destination node can see from these edges the amount of data that has been sent from X to Y corresponding to this route.

- If D recognizes that the number of bytes received differs significantly from the number of bytes originated by the source the leak detection mechanism is enabled. The optimal threshold for the difference between incoming and outgoing traffic is to be found empirically. Various parameters such as the importance of the traffic, the tolerance ratio of the user and device capabilities have to be considered.

- The graph is further maintained and the amount of incoming traffic and outgoing traffic is calculated for each node. If the values of a specific node M differ significantly due to the outgoing traffic being far less than the incoming data, the destination node D assumes that M is malicious.

  We have used following criteria to determine misbehavior: The difference between the outgoing and the incoming has to be at least 5% of the total traffic for this route. Additionally at least 50 packets are needed to pass the current node analyzed to provide a minimal set of observations. Whereas these previous two criteria are needed to start the leak detection procedure, the next decides whether an analyzed node is to be seen as malicious or not. The criteria is, that the incomming traffic is higher that 5% of the total traffic and the corresponding outgoing fraction is at least three time as high as the incomming. By this we can determine nodes with a drop ratio of at least 30%. As the parameters we have chosen are only needed for a proof of concept (see Chapter 8), there is room for optimization.

- Since Maintenance Forward Ants are generated frequently to reinforce the current routes, following is done when assuming that one of the nodes on the route is malicious: the destination node does not answer with a Backward Ant. By this the route containing the assumed malicious node is not reinforced and so alternative routes get a chance to be more utilized.

- Due to the fuzzy characteristic of this mechanism we can reset the traffic counter more easily. As the graph represents the network state only using old partial information (Forward Ants), we can be sure that the fractions are not perfectly true. But this is not necessary, as our leak detection mechanism reacts only when loss is detected.

  Resetting the traffic counter to an initial value (0) is necessary to avoid a node which has cooperatively participated in routing first not being detected after changing to malicious behavior, because its previous good conduct dominates the information in the graph. Resetting the counter is done periodically (for example every 10 minutes). In a time window of 30 seconds each node resets its internal traffic counter for the current route to 0. Any Forward Ant is then treated as normal, but expected

to have a total traffic value of 0. The destination node of the route resets the internal graph and rebuilds a new one.

Our leak detection mechanism provides various characteristics that suit well to our design goals. First of all the detection quality is very good, benign nodes are almost never detected as malicious. Malicious nodes are detected when they are utilized in routing frequently. Malicious nodes used only rarely in routing do not give enough ambiguous information to be detected. Another benefit of this mechanism is its sliding quality scale. Using fractions of the total traffic sent provides us with a broad information range of the current state of the network. This range gives space for intelligent strategies on how to handle detected colluding misbehaving nodes.

An important point of our leak detection mechanism is that it suits well to AntSec. As information about the traffic states are sent in insecure FANTs they can be modified and provide a false picture of the network to the destination node. But as such FANTs are returned as signed BANTs, they traverse the path back to the source node and any node can check whether the information in the BANT is equal to the information they included in the FANT. So concluding malicious nodes can modify the traffic describing information, but this leads to a detection and they are utilized less on the route.

No cryptography is used at all, especially no asymmetric cryptography. This saves first computational power and also traffic overhead.

6.6 Reputation Management

In the last section we illustrated AntSec - Secure AntNet, a routing algorithm which prevents common attacks or at least detects them. A great advantage of AntSec is that a single ANT does not have a big impact on the network, but it is sufficient to detect misbehavior. Any attacker will have to misbehave several times and make the detection easier. Afterwards we introduced WatchAnt and the leak detection mechanism to detect misbehavior in the forwarding behavior of the neighboring nodes and to detect malicious colluding nodes on the route which are not detected by WatchAnt.

In this chapter we introduce our Reputation Management concept. Detection of malicious or benign behavior makes no sense without concepts to maintain a view on the current state of neighboring nodes and to provide strategies on how to cope with misbehavior. First we introduce our solution to maintaining reputation information about neighboring nodes based on results from WatchAnt. In the next step describe strategies to share local reputation information with other nodes in the network to increase the detection rate. At last we give an outlook at how central servers or trusted sub-networks can be

involved in the maintenance of the reputations.

6.6.1 Direct Neighbor: First Hand Reputation

Our concept relays on local observations and local decisions, so each node maintains its reputations locally using a reputation table similar to the routing table. For each neighboring node the current reputation and a list of reputations provided by other neighboring nodes is maintained (for latter see Subsection 6.6.2). The reputation value is a real number ranging from *-60* to *+25*.

We have identified the following actions that have effect on the reputation of a neighboring node:

- Loops in FANTs can be detected, but the previous hop should have avoided it.
- FANTs should not be forwarded to a node which has no other neighbor than listed in the FANT. Dead ends have to be avoided.
- It may occur that a loop is detected and the list of visited nodes in the FANT is modified too, so that the previous hop could not see that it is creating a loop.
- As BANTs contain the information of the corresponding FANT, there should not be modifications. If there are, the previous hop's reputation is decreased.
- BANTs are signed and a BANT with an invalid signature should have been dropped by the previous hop.
- As BANTs are only accepted when the corresponding FANT has passed, invalid BANTs are dropped and the previous hop is rated bad, since the neighboring node should have dropped the BANT itself.
- A WatchAnt request may be not answered, which can be interpreted as an action to hide misbehavior.
- A WatchAnt reply may be malformed.
- A WatchAnt reply may be false, i.e. provide a wrong answer or an unknown reference outgoing node.
- A WatchAnt reply may contain illegal or no entries.
- A WatchAnt reply may be received even though there none were requested. This might be a try to improve reputation by sending valid WatchAnt replies frequently.

- On the other hand a referenced outgoing node can receive a WatchAnt reply which is malformed or contains illegal information.
- Referenced outgoing nodes may also find a false information describing the amount of data received from the answering node(false reply).
- There are only two cases in which reputation rises: The referenced outgoing node receives a WatchAnt reply with correct information.
- The WatchAnt requesting node (starting the examination process) receives a WatchAnt reply containing authenticated nodes and correct information about the forwarded packets. In this case the reputation rises, too.

The changes in reputation are event triggered. On each action mentioned above (or at least after every reception of a WatchAnt reply) the reputation of the inquired node is adapted.

6.6.1.1 Usage of Reputations

As every node gathers and maintains the reputation information about each of its neighboring nodes, we point out how these reputations are used in routing in this subsection. The range of reputations goes from -60 (which is very bad) to $+25$ (which is very good). The initial reputation value is 0. Since WatchAnt requests are generated regularly, events to trigger changes in reputation occur regularly too.

We define three thresholds in the range of the reputation values to define differing strategies of how to cope with a specific neighboring node. Following thresholds for reputation values are defined:

- 25 Maximum reputation value
- 0 Initial reputation value
- -25 No-reputation-bonus threshold
- -40 Punishment threshold
- -60 Minimum reputation value

In the range of -25 to 25 the neighboring node is treated normally as it is done in typical routing algorithms. This range we call normal behavior. In the range lower than -25 positive reputation modifications are not applied. This means that the reputation of the observed node can only get worse or recover by time. In this case the observed node is not given the chance to raise its reputation by some benign action.

When the reputation of a neighboring node falls under -40 two changes take effect: First the node is excluded from routing, i.e. its probability of being chosen as next hop to a certain destination node is set to the minimal value. Its probability value is shared among the other neighboring nodes, with respect to their current probabilities. In this process no other malicious node takes benefit, as every node considered being malicious is excluded from sharing the probability.

Example: Node X has 3 neighbors A, B and C. with the probabilities 0.40 , 0.35 and 0.25 to be chosen as next hop on the route to another node D. When the reputation of A from X point of view (stored locally in X) falls below -40, then X sets the probability of A to 0 and raises the probability of B to $0.58333 = 0.35 + \left(0.40 \cdot \frac{0.35}{0.35+0.25}\right)$ and of C to $0.41666 = 0.25 + \left(0.40 \cdot \frac{0.25}{0.35+0.25}\right)$.

To avoid selfish nodes using this reaction to be excluded from the burden of routing, additionally malicious nodes are excluded from being served. This means that any message generated by this malicious node is dropped by its neighbors that consider it to be malicious.

This behavior can be dangerous for the quality of routing, as it may happen that benign nodes are accidentally excluded from service. They can in turn exclude their neighbors too. So two benign nodes can consider each other as malicious and therefore exclude each other from being served. In reaction to not being served malicious behavior of the neighboring node is assumed.

To break this circle a node does not send WatchDog Requests to neighboring nodes considered malicious. By this both reputations can recover and normal service is provided afterwards.

6.6.1.2 Second Chance Mechanism

Second chance mechanisms are characteristic for misbehavior detection mechanisms, as the relevance of observations fades with time. Inactivity (for example when not being utilized in routing) is not to be considered harmful. We introduce a second chance mechanism giving nodes, which have previously been considered being malicious, a chance to recover their bad reputation and get a new chance of participating in routing. Therefore periodically the reputations of all neighboring nodes are modified to convert to the initial reputation value (0). The Following function is used:

- If current reputation value $oldRep$ is positive: $newRep = 0.9 \cdot oldRep$
- If current reputation value $oldRep$ is between -40 and 0: $newRep = 0.98 \cdot oldRep$
- If current reputation value $oldRep$ is less than -40: $newRep = 0.99 \cdot oldRep$

With this reputation fading mechanism, inactivity leads to rehabilitation (in the case that one was considered malicious). As with other parameters, these values are open for optimization.

As we have seen in this subsection reputations are maintained locally. They represent the subjective view of one node observing its neighbors. The reputation values of a single node maintained by two of its neighbors can be completely different. This single node can behave differently with respect to its neighboring nodes. Each node decides based on its local reputation table, how to cope with each single of its neighboring nodes. As investigations of Sonja Buchegger [Buc04] have shown the detection speed can be increased if subjective observations are spread as second hand reputations. We discuss this topic in the following subsection.

6.6.2 Direct Neighbor: Second Hand Reputation

In this subsection we point out the benefits of the usage of second hand reputations and how this concept can be adapted to our security framework. We identify the following three concepts to deal with reputations:

1. No second hand reputations is used. By keeping the principle of local decisions based on direct observations, the simplicity of the system is saved.

2. Periodically send the reputations-list based on direct observations to the neighbors.

3. Use a distributed storage system (like in P-Grid [Abe01]) for maintaining 'global' reputation.

The usage of only direct observation is the easiest solution, but as Buchegger has shown in [Buc04], the usage of second hand reputations always provides better detection results. The third solution would be to use the direct observations to build up a 'global' reputation table and to use periodically retrieved snapshots of it as basis for local decisions. This solution is feasible but it has several drawbacks. First, it is difficult to find a set of trusted nodes in the network which can be utilized for this task. If no a priori trusted set of nodes exists, then a set of peers has to be determined to fulfill this task. To avoid one node of this set being able to behave harmful, strategies like threshold cryptography [Sha79] have to be applied. But this task requires smart organization of the peers and additional traffic and computational overhead. We do investigate any further on this topic, as it is a research topic of its own.

We focus on the second solution: Using second hand reputations to improve the detection quality. This solution requires two tasks:

- The observations / reputations have to be spread among the neighbors.
- The received second hand reputations have to be accumulated with the direct observations to build a basis for local decisions.

Broadcasting is the most efficient way for a node to spread information to all of its neighbors, as the wireless medium only has to be utilized once. In IEEE 802.16 Mesh Mode broadcasting mechanisms are rare and only native for the transmission of control messages. Two types of information have to be spread:

1. An identifier for the specific neighbor, for which the reputations is publicized.
2. The reputation value of this neighbor.

As node identifier the network address, the MAC address and the NodeID are available. We use the NodeID as its size is only 16 bits. In comparison to this the network address is 32 bits and the MAC address is 48 bits long. As each node in the 2-hop-neighborhood of a node is authenticated by receiving a FANT from it, we additionally get to know the NodeID, the network address and MAC address of the authenticated node from the RegCert in the FANT (see Subsection 6.1.5.6). So all identifiers are connected and known and we can use the shortest identifier of these without loosing any information.

As broadcasting capabilities in IEEE 802.16 Mesh Mode are very scarce we want to base our solution on already existing broadcasts and enhance them with minimal modifications. We introduce the following solution for broadcasting reputation values to neighboring nodes:

- MSH-NCFG messages (specified in Subsection 6.3.2.3.35 of the standard [ICS04]) have an extra field for $MSCH - NCFG_embedded_data()$. See Figure 2.2.
- $NCFG_embedded_data()$ is specified in Subsection 6.3.2.3.35.3 of the standard [ICS04]. It contains a type identifier (4 bits), the length (8 bits) of the current Embedded Information Entry (variable length) and a flag ($Extended\ embedded_data$, 1 bit) signaling whether the current Embedded Information Entry (variable length) is followed by another one.
- As the type identifier provides space for up to 16 different types, of which only 7 are specified (six in the standard, the seventh in Subsection 6.4.3: *WatchAnt Information Entry*. For the spreading of second hand reputations we add an additional type: *Reputation Information Entry*.

- As MSH-NCFG messages already contains a list of the current neighbors (Nbr Node ID (16 bits)) we can use this information to shrink the size needed for reputations.
- Reputation Info (n bytes): Containing the reputation (8 bits) of all n neighbors listed in the NCFG message. The order of the reputations matches the order of the neighboring node IDs.

With this mechanism each node can periodically spread its reputations table to its neighboring nodes. The strategies of how these second hand reputations are accumulated with reputations based on direct observations are numerous. In [Buc04] and [BB03b] Sonja Buchegger evaluates different approaches of processing second hand reputations. She evaluated following strategies:

- Using only first hand reputations.
- Using second hand reputations with a simple mechanism to detect lies (ignoring contradictory reputations).
- Using a Bayesian approach to determine whether a neighboring node is sending truthful or false reputation values.

Simulations [BB03b] reveal that the Bayesian approach offers a faster detection time of misbehavior. To cope with false or erratic second hand reputations while still providing good detection results a specific form of Byzantine robustness [Per88] is necessary.

As the handling of second hand reputations is a wide research topic, we do not focus on it. In this subsection we introduced a mechanism to spread second hand reputations in IEEE 802.16 Mesh networks.

6.6.3 Central Reputation Management

Central Reputation Management strategies are characterized by having a central server (or server cluster) in the network, which is responsible for the maintenance of global reputations.

The benefit of this concept is that global reputations exist, since only one instance is storing all reputation values. Nevertheless the drawbacks are more numerous and important. First the reputation server is a single point of failure, being down would disable all security mechanisms. Additionally, storing reputation value on a distant server, needing several hops to retrieve / update the crucial reputation values, is very ineffective with respect to the traffic overhead. As this server is frequently contacted by a huge number of

nodes in the network, it is to be seen as a bottleneck. Furthermore servers maintaining reputation values need to be trusted by any participating entity in the network. All in all using a central server (or server cluster) for the maintenance of reputation values offers more disadvantages than benefits.

Nevertheless central servers can be utilized in our concept in addition to provide an incentive to participate in routing: As each node has to be authorized to participate in the network, authorization can be refused when the node requesting authorization has been considered as malicious by numerous other nodes in previous session. Therefore each node in the network stores the identity of neighboring nodes it considers being malicious and transfers this information irregularly, when reliable fast connection is available, to a central server of the network provider (Reputation Node).

This single server is not responsible for maintaining the reputations. It is used to store complaints about nodes in the network only. If numerous different nodes in the network send a complaint about one single node X and this node X is regularly considered as malicious, the network provider can contact the owner of this node to discuss the reasons for the malicious behavior. To shift the responsibility for misbehavior from a node to its real-life owner provides an incentive to participate in the network, as non-repudiation is guaranteed.

The drawback of this concept is that malicious nodes can cooperate to discredit a specific node in the network. But this can be detected, since the specified victim node is not rated bad when being in interaction with non-malicious nodes.

This simple addition provides a good psychological incentive to participate in routing for very little costs. Memory efficient storing is possible, since only node identifiers and time periods have to be stored. Transmitting the complaints to the Reputation Node can be done infrequently when network utilization is low and no regular traffic is disturbed.

Chapter 7

Specific Attack Tree

Overview

7.1 Analysis of Attacks Committed by Selfish Nodes 132

7.2 Analysis of Attacks Committed by Malicious Nodes 134

Abstract

In this chapter we examine AntSec, WatchAnt and the Leak Detection Mechanism in reaction of all attacks we found applicable to routing algorithms in mesh networks. We see that the combination of the presented security mechanism is able to detect and prevent all relevant attacks.

We give now a survey on possible attacks, separated according the motivation of the attacking nodes: being defective, selfish or malicious. As defective nodes do not work properly they may be considered as black holes (dropping all packets) or gray holes (dropping some packets). Defective nodes can only do attacks that selfish or malicious nodes can also do intentionally. Therefore we do not look further at attacks done by defective nodes, these attacks will be considered at the analysis of the attacks done by selfish or malicious nodes.

7.1 Analysis of Attacks Committed by Selfish Nodes

Tree 1: Conserving resources. Selfish behavior.
OR 1.1 No participation in the routing process (no routing, no data)
OR 1.1a No forwarding of routing packets (FANT, BANT)
1.1a.1 No forwarding of FANTs
1.1a.2 No forwarding of BANTs
1.1a.3 Setting of Hop-Limit / TTL to 0 or 1
OR 1.1b Modification of Routing Information
1.1b.1 Worsen the quality metrics in the FANT
1.1b.2 Modifying the Backward Ant
OR 1.2 Do not forward data packets (no data)
1.2a Drop data packets
1.2b Set Hop-Limit / TTL to 0 / 1
1.3 Do not list all neighbors in MSH-NCFG messages

Table 7.1: Specific Attack Tree Part A: Selfish Behavior

For the following investigations we discuss the attacks listed in Table 7.1.

1.1a.1 / 1.1a.2: No Forwarding of FANTs / BANTs

This attack is detected by WatchAnt. Routing messages and data packets are both taken into account by the WatchAnt mechanism, so that loss of these packets is detected.

1.1a.3 / 1.2b: Modifications on the Hopcount / TTL

This attack is harder to detect, still it is done by WatchAnt. WatchAnt has to store the hashes of each packet in a way that the hash can later be recognized. The problem with the TTL-field is that it changes with each hop. In contrast to this the hash value of a packet changes with modifying the TTL. This is taken into account, by storing various hashes of a packet with varying valid TTLs. WatchAnt detects invalid modifications.

1.1b.2: Modifying the Backward Ant

This attack is not effective in AntSec, as the BANT is signed. Any modification to the BANT will be detected as the signature is not valid anymore.

1.2a Drop data packets

Dropping of data packets is detected by WatchAnt, as WatchAnt has been designed to counteract this attack in special.

1.3 Do not list all neighbors in MSH-NCFG messages

By this a malicious node could try to signal that it has not enough neighbors to forward messages. Any node getting a MSH-NCFG knows that the sender is in reach and so the receiver should be in the MSH-NCFG too. An attacker cannot perform this attack as it has to synchronize with neighboring nodes to schedule access to the control subframe. So if there is a violation, the violating node may be considered as a malicious node.

7.2 Analysis of Attacks Committed by Malicious Nodes

Now we discuss the attacks listed in Table 7.2.

2. Malicious nodes (attacks on the Operability/Efficiency of some/all nodes)
2.1 Destruction of some/all data packets OR 2.1a Disturbing of the whole frequency-band (traffic jamming) 2.1b Producing collisions with minimal effort (send a short packets for collision)
2.2 Generating an overload on bandwidth, CPU, energy OR 2.2a Overload of direct neighbors 2.2b Overload of arbitrary nodes on a route in the network OR 2.2b.1 Generate a route discovery storm (many useless route requests) 2.2b.2 Replay old BANTs / FANTs 2.2b.3 Replay old data packets 2.2b.4 Route packets over bad (not optimal) paths 2.2b.5 Generate routing loops 2.2b.6 Forwarding as many packets as possible over a specific neighboring node 2.2c Overload by claiming all free time slots for own usage
2.3 Disturbing the accurate routing OR 2.3a Packet loss/dropping (Black Hole Routing) 2.3a.1 Send the packets to a non existent neighbor 2.3a.2 Drop any packet, be a black hole 2.3b Modify routing packets 2.3c Forge routing packets
2.4 Participate unauthorized in the network OR 2.4a Use the same Node ID at different locations
2.5 Avoid being detected by WatchAnt OR 2.5a Send false information in the WatchAnt Reply (lie) 2.5b Lie to every neighbor in same frequency 2.5c Drop with a very low frequency
2.6 Use the Reputation Management System for attacks OR 2.6a Effect that benign nodes suspect each other as malicious 2.6b Broadcast false reputations
2.7 Avoid being detected by the Leak Detector OR 2.7a Tell false about information passing traffic 2.7b Drop data packets and forge invalid data packets additionally

Table 7.2: Specific Attack Tree Part B: Attacks of Malicious Nodes

2.1: Destruction of some/all data packets by forging arbitrary invalid packets

Handling with attacks on the PHY level are hardly to antagonize. Detecting permanent collisions in reserved time slots could be a signal for such an attack. Still attacks on PHY

layer are out of scope for our work.

2.2a: Overload of direct neighbors

In IEEE 802.16 there is no easy way to overload neighboring nodes. For both nodes in a communication link on MAC layer the efforts needed to communicate are similar. Both need to invest transmit and receive time, both nodes have to process a transmitted packet. The bandwidth of each link is negotiated either using centralized or distributed scheduling. Using distributed scheduling each node can request bandwidth to neighboring nodes and grant bandwidth upon receiving such a request. Upon detection of malicious behavior of a neighboring node, observations on the routing layer can be passed to the MAC layer and bandwidth to this node can be decreased.

2.2b.1: Route discovery storm

In contrast to various routing algorithms AntSec does not use any broadcasting. In many other routing algorithms route discovery storms are a problem, as with a single request the network can be flooded. In AntSec a route request causes the generated FANT to be routed using random walk to a specific destination node. Still in AntSec Discovery Forward Ants are sent quite frequently (400ms, see Table 8.1). Therefore any node processes Discovery Forward Ants from any of its neighbors only if these are sent in an interval higher or equal 400ms.

2.2b.2 / 2.2b.3: Replay of old packets (FANT / BANT / data)

Replay of old packets can be seen as forgery of new invalid packets. The same solution counts here too. The WatchAnt mechanism checks the correct transmission of a set of packets. If packets were forged or replayed, then this the set of forwarded packets contains invalid packets and the WatchAnt reply becomes false. By this forged and replayed packets the reply is wrong and malicious nodes are detected.

2.2b.4: Route packets over bad (not optimal) paths

This attack effects longer paths and even loops may occur. As AntSec is a probabilistic routing algorithm, forwarding packets over not optimal paths may occur and cannot be detected as attack. The effect of this attack is limited, as a malicious node can only chose the next hop for a packet, not the complete route.

2.2b.5: Generate routing loops

Rooting loops cannot be generated by one single node. This is due the fact that AntSec is a decentralized routing algorithm, each node decides about the next hop of a packet. A single node can generate a loop only by sending a back over the hop it came. But this attack is detected immediately. And when alternatives to the previous hop of a packet exist, it is not allowed to send the packet back.

Malicious colluding nodes can generated routing loops by passing packets to each other until the TTL limit is reached. This attack is equal in its effect to the dropping attack of colluding malicious nodes. The counteraction to this attack is defined by the Leak Detector, which detects leaks on routes caused by colluding malicious nodes.

2.2b.6: Forwarding as many packets as possible over a specific neighboring node

A malicious node may forward any packet it receives to a specific neighboring node to stress it. This cannot be detected as misbehavior, as it may be done by a benign node, too. Each node has only local sight on the network, so due to the topology it may be that some neighboring nodes are stronger utilized in routing than others.

2.2c: Overload by claiming all free time slots for own usage

A malicious node can try to reserve all free time slot for it self and decrease by this the remaining bandwidth for all other nodes. This attack is performed on the MAC layer. With our security framework we cannot detect this. The MAC layer should provide mechanisms to detect invalid reservations.

2.3a.1: Send the packets to a non existent neighbor

It is ineffective for an attacker to denote to have sent packets to a node, which is not existing in reality. The WatchAnt Reply may contain the Node ID of this non existing node, but the WatchAnt requesting node checks whether the listed nodes are known and authenticated. If not, the WatchAnt Reply is invalid. A non existent node cannot be authenticated as there is no Registration Certificate existing. Malicious nodes may try to use a Registration Certificate of a distant node to imply that this node is in its near. To avoid a malicious node using the Registration Certificate of a distant node we suggest that a signed list of neighboring nodes shall be transmitted upon request, by this the authentication and location of a node is determined. In our implementation we did not focus on this attack.

2.3a.2: Drop any packet, be a black hole

The dropping of routing and data messages is detected by WatchAnt. Additionally AntSec rates a neighboring node bad when a FANT was sent over this neighboring node and no BANT arrives.

2.3b: Modify routing packets

The modification of FANTs is not detected on the route to the destination. But as the signed BANT returns on the same path, which the corresponding FANT used, the modification is detected since there is no FANT corresponding to the received BANT. BANT cannot be modified, as there are signed and any modification will be detected immediately.

2.3c: Forge routing packets

Forward Ant can be forged undetected. A FANT does not contain any cryptographically mechanism assuring its authenticity and integrity. A forged or spoofed FANT is processed regularly. The corresponding BANT is also processed as usual. But the malicious effect is limited. The effect of a forged packet is equal to the effect of a FANT sent from the malicious node. The processing of FANTs and BANTs does not take into account the sender of the packets. Processing a BANT modifies the route to each node on the path to the destination node, but the source node is irrelevant.

To forge BANTs does not make sense. Spoofing other nodes is impossible, as their signature cannot be created by any other node. The malicious node may create a BANT to a non existent FANT to reinforce specific paths. But this BANT is dropped by the first benign node, since no corresponding FANT is known.

2.4a: Use the same NodeID at different locations

If different nodes use the same Registration Certificate, i.e. same Credentials, and misbehave, then complaints about this node are faster and in larger amount reported to the network provider. The problem occurs when nodes using these multi identities are well behaving and do not behave suspiciously. In this case malicious nodes can participate with this identity on several places in the network. Still this attack does not make sense, routing cannot be guaranteed for this malicious node. Packets addressed for this identity used multiple can be routed to any copy of this identity, dependable communication is not possible.

2.5a, 2.5b: Avoid being detected by WatchAnt by lying in the WatchAnt Reply

A malicious node may drop all data packets and report in the WatchAnt Reply to have sent all packets to one single neighbor of it. The node, which has sent the WatchAnt Request assumes that all packets have been forwarded correctly. The only node detecting the misbehavior is the node listed in the WatchAnt Reply.

The malicious node can use two different strategies: Either the malicious node accepts being detected by a neighboring node which is not important for it or it tries to be detected by any neighbor. In the first case, the malicious node lists every time the same neighboring node as next hop for all packets received. This neighboring node detects the misbehavior, may even denial any service for the malicious node, but as the detecting node is not relevant for the malicious node, no real punishment is done. In the second case, the malicious node reports every time another neighbor of having received all packets. In this case the local reputation of the malicious node is decreased periodically in each neighbor by a little, but with recovery the negative effect may be reduced to none.

In both cases WatchAnt can be optimized by modifying the parameters used in Table 8.1 to be more sensitive on frequent false WatchAnt replies. Still using second hand reputations can improve the detection quality.

2.5c: Drop with a very low frequency

By dropping with a very low frequency a malicious node may try to drop packets to which no report is requested. Nevertheless, to avoid dropping packets in a requested window, the malicious node has to drop with a very low rate. Even when only every second WatchAnt Requests detect misbehavior, misbehavior is rated stronger than good behavior. This attack is not effective, as avoiding detection requires a very low (irrelevant) drop rate. This case is investigated in simulation Scenario B in Section 8.3.

2.6a: Effect that benign nodes suspect each other as malicious

The strongest punishment done by nodes in the network is to deny service to a neighboring node, which is suspected of being malicious. A malicious node may try to effect, that benign nodes suspect each other of being malicious and mutually deny service. Very few actions can be done to lower the reputation of a node locally in another node. The WatchAnt mechanism, which has the greatest impact on the reputation of a node, cannot be used to discredit a neighboring node from the view of another node. Invalid actions on FANTs and BANTs are detect and punished. So there is no feasible way to worsen the reputation of neighboring nodes from other nodes point of view.

2.6b: Broadcast false reputations

Broadcasting false reputation values can be used to worsen the reputation of neighboring nodes. Taking second hand reputations into account needs mechanisms to detect invalid second hand reputations coming from malicious nodes. Sonja Buchegger presents in [Buc04] various strategies to use for routing second hand reputation values, which may be invalid.

2.7a: Tell false information about passing traffic

Using the Leak Detector requires the nodes to tell which amount of traffic on the route they have received from the previous hop of the MFANT. By giving a false information, malicious nodes may try to make the previous hop responsible for packet dropping. The destination node does not detect this attack, the information is used for the Leak Detector. But still the attack is detected. As the BANT is sent back by the destination node, it is forwarded using the route, on which the corresponding FANT came. As the BANT arrives at the node which was the previous hop for the FANT before the malicious node, the invalid modification is detected. The previous hop assumes that up to its successor no modification has been detected (as the BANT was not dropped). Therefore the previous node drops the invalid BANT and decreases the reputation of its successor: the malicious node.

2.7b: Dropping and forging of data packets by colluding malicious nodes

Sending of invalid or corrupted packets with spoofing of source addresses is a common problem. The main function of routing messages is to transfer data packets from one node to another, not regarding the content of the packet. Still WatchAnt detects forged data and routing packets, as for each packet the previous hop has to revealed when forwarding the packet to a next hop. Any lying at this point will be detected by WatchAnt, due to the fact that the forged packet is mapped to the wrong stack (identified by a different pre hop-next hop) maintaining the received and sent packet hashes. The problem is when malicious nodes collude to trick WatchAnt and additionally forge for every packet they drop a new packet to trick the Leak Detector, then this misbehavior is not detected by our security systems.

Chapter 8

Simulation Setup and Results

Overview

8.1 Implementation 142
8.2 Simulation Setup: Scenario A 145
8.3 Simulation Setup: Scenario B 148
8.4 Simulation Setup: Scenario C 150
8.5 Results: Scenario A 152
8.6 Results: Scenario B 173
8.7 Results: Scenario C 180

Abstract

In this chapter we describe which tools we used to implement the mechanisms and how the simulation setup was chosen to evaluate our solution. First we give an introduction to JiST (Java in Simulation Time), SWANS (Scalable Wireless Ad hoc Network Simulator) and the MoBQoS project at the Multimedia Communications Lab (KOM) at Technische Universität Darmstadt.

After this we present three scenarios. First we compare the performance of AntSec (with security mechanisms enabled / disabled) with DSR under varying ratio of malicious nodes in the network. In the second scenario we simulate 6 nodes, one of them being malicious and varying its drop rate. Here we want to see which effect the drop ratio of malicious nodes has on the detection quality. The third scenario shall reveal the effects of colluding malicious nodes on the network and show the detection quality of the Leak Detector.

8.1 Implementation

We have implemented our solution on top of JiST - SWANS - MOBQoS framework. JiST (Java in Simulation Time) [BHvR05b] is a discrete event simulator written in Java, running on a Java Virtual Machine. It provides a platform to run programs written in Java and use additional functionality which is needed for simulations: simulation time. By the introduction of simulation time, concurrent operations can be modeled. Additionally Java programs can use JiST specific commands to schedule the operation of instructions. This enables the construction of complex, time-based software.

SWANS [BHvR05a] is a collection of software components, providing functionality of a network simulator. These software components are written in Java and are designed to run on JiST. SWANS contains the implementation of various network layers enabling the users to enhance current implementations or to add new components. *"Its capabilities are similar to ns2 and GloMoSim"* as the developers of SWANS say.

MoBQoS is a project of the Multimedia Communications Lab (KOM) at Technische Universität Darmstadt. In this project the MAC layer of IEEE 802.16 Mesh Mode [ICS04] is implemented (for more details see [Bli06]). Additionally the routing protocols QSRM, AODV and Ant-Routing are implemented and evaluated (see [Zil06]). Besides the aspects regarding the Quality of Service on the MAC layer and network layer in IEEE 802.16 Mesh Mode, security issues are also investigated. In [Sar06] Damir Sarac investigates security mechanisms for the MAC layer of IEEE 802.16 Mesh Mode. The main focus of [Sar06] is to provide mechanisms for secure network entry and bootstrapping authenticity. This thesis investigates security issues in routing and provides a security framework to counteract misbehavior of malicious nodes in the network.

The implementation of our solution comprises 4 main components:

- CryptoClass and Certificates: We implemented a class providing cryptographic functionality and a set of certificates (RegistrationCertificate, IdentityCertificate and ManufacturerCertificate).

- RouteAntSec contains the security framework with its 4 main components: AntSec, WatchAnt, Reputation Management System and Leak Detector. AntSec uses the Ant-Routing algorithm described in [Zil06] as basis. Parameters that are not security related are set as described in the diploma thesis of Andreas Ziller [Zil06].

- Slight modifications to the MAC layer that are required for WatchAnt and to enable the sending of reputations to the neighboring nodes have been implemented.

- A statistics module to log the simulation results has been integrated in the code.

In the upcoming simulation setups we refer to the protocol *AntSec*, when all security features, except the Leak Detector, are enabled. With *Ant* we denote our routing protocol with all security features disabled.

In Table 8.1 we give a brief description of the parameters used for Ant and AntSec. A description of the parameters which determine the values by which reputations are modified is given in Section 6.6.1. Furthermore the naming of the parameters is self-describing.

Parameter	Value
Routing protocol related:	
AS_Maximal_hop_count_for_packets	16
AS_Hash_algorithm	"SHA1"
AS_Maintenance_Forward_Ant_send_interval	500 ms
AS_Discovery_Forward_Ant_send_interval	400 ms
AS_Number_of_simultaneous_Maintenance_Forward_Ants	3
AS_Number_of_simultaneous_Discovery_Forward_Ants	1
AS_Maintenance_Forward_Ant_send_interval_per_dest.	5 sec.
AS_Observerd_Fants_Buffer_Timeout	100 sec.
AS_Fraction_of_Prob._value_spread_among_benign_Nbrs_if_detected	0.8
AS_Fraction_by_which_a_nexthop_prob._is_decreased_if_no_Bant_arrives	0.15
WatchAnt related:	
WA_Number_of_hashes_to_ask_for	12
WA_Ask_only_for_hashes_older_than_this	50
WA_Number_of_hashes_to_maintain_per_nbr_nbr_combination	100
WA_Reply_timeout	15 sec
Reputation Management System related:	
RM_Reputation_Initial_value	0
RM_Reputation_Minimum	-60
RM_Reputation_Maximum	25
RM_Reputation_below_this_no_positive_bonus	-25
RM_Reputation_below_this_no_Service_and_no_routing	-40
RM_Second_hand_reputation_broadcasting_interval	10 sec
RM_Reputation_normalization_value_if_positive	0.9
RM_Reputation_normalization_value_if_negative	0.98
RM_Reputation_normalization_value_if_malicious	0.99
RM_Routing_probability_reserved_for_each_neighbor	0.02
RM_Normalization_Interval	1 sec
Modification values based on observations:	
AS_Loop_Detected	-5
AS_Dead_End_Cannot_Forward	-1
AS_Invalid_BAnt_signature	-5
AS_FAnt_has_false_Path_History_Went_In_Loop	-5
AS_No_FAnt_For_BAnt_Observerd	-5
WA_ReferencedNode_CheckAnswer_False	-5
WA_ReferencedNode_Received_WA_Reply_Incomplete	-5
WA_RequestingNode_WA_Request_Remained_Unreplied	-5
WA_RequestingNode_Receiving_Unrequested_WA_Reply	-5
WA_RequestingNode_Receiving_Illegal_WA_Reply	-5
WA_RequestingNode_Received_WA_Reply_Incomplete	-5
WA_RequestingNode_CheckAnswer_False	-10
WA_RequestingNode_CheckAnswer_Correct	+5
WA_ReferencedNode_CheckAnswer_Correct	+2

Table 8.1: Parameters used for Ant and AntSec

8.2 Simulation Setup: Scenario A

The goal of the first scenario is to compare the performance of AntSec, Ant and DSR in a network with varying fraction of malicious nodes. We compare AntSec with security features enabled, AntSec with security features disabled (called Ant) and Dynamic Source Routing (DSR) [JM96]. The DSR protocol is used as reference. An optimized version is used, as IEEE 802.16 does only provide limited broadcasting probabilities.

We evaluate the efficiency of the routing protocols in a network with 50 nodes. The placement of the node is random mesh topology. Figure 8.1 shows the characteristics of the random mesh topology when using the criteria defined by us.

The figure shows 110 nodes to demonstrate the properties characteristics of the random mesh topology, for our simulations we used 50 nodes.

The following metrics are considered:

- Throughput

  This metric is very important, as it is a design goal of AntSec. We set the number of received packets in relation to the number of originated packets. Malicious nodes in the network drop data packets not generated by themselves and not destined to them. The most interesting part of the simulations is whether AntSec performs better than Ant and DSR. If this is the case, it is to be shown by which amount AntSec performs better than Ant and DSR.

- Overhead

  Although AntSec requires additional computational power, we ignore the computational overhead and focus on traffic overhead. We measure the overhead first in the number of routing packets and the fraction of traffic amount (in bytes) needed for routing in relation to the total amount of traffic.

- Detection quality

  We evaluate the quality of detection by analyzing the number of detected nodes in various cases. Here it is interesting how many malicious nodes are detected and by how many of their neighbors malicious nodes are detected. Is the detection rate increasing or decreasing over time?
  We also look at the ratio of false positives and false negatives. False positive means that a benign node has falsely been suspected as malicious by one of its neighbors. False negative means, that a malicious node is not detected as malicious by its neighbor.

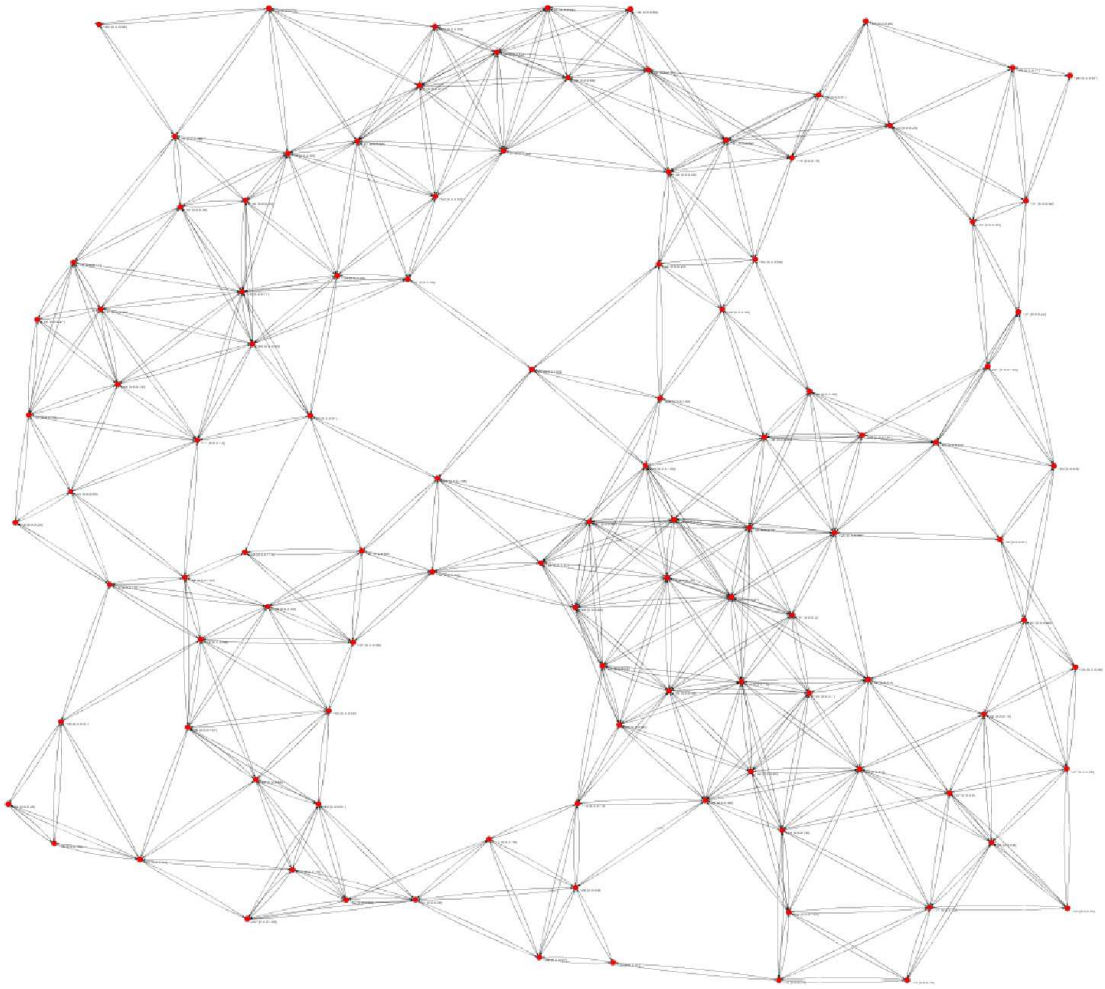


Figure 8.1: Sample mesh network topology containing 110 nodes. This figure shows the characteristics of random mesh topology. For our simulations we used 50 nodes.

- Average number of hops for different packets

  We analyze the average hop number of DFANTs, MFANTs, BANTs and data packets when using Ant and AntSec. By measuring the average hop number of packets, we can see whether the length of the routes increase with higher ratio of malicious nodes in the network.

The complete parameter list is given in Table 8.2.

Parameter	Value
Number of nodes	50
Area	5500m x 5500m
Radio range	627m
Placement	Random mesh
Movement	none, fixed positions
MAC	802.16 Mesh Mode
Number of flows	25
Flow pairing	random, but not to self
Sending rate per flow	10 packets per sec.
Flow type	Constant Bit Rate (CBR)
Packet size	512 bytes
Route discovery time	300 s
Simulation time	1000 s
Number of replications	20
Varying parameter	Ratio of malicious nodes in the network
	Ratio: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5

Table 8.2: Parameters for Scenario A

We explain why this parameter set was chosen. 50 nodes are simulated as this amount of nodes is well chosen to have a network that is large enough to provide alternative paths to routes with malicious nodes. Furthermore it is common for evaluation of routing protocols in networks with malicious nodes (see [Buc04], [Kar03]). The area is chosen large enough so that the network can grow organically during the creation of the random mesh topology. We used IEEE 802.16 Mesh Mode [ICS04] as MAC layer, the revision of the standard we referenced considers only static fixed node, so that there is no mobility in our simulations. 25 flows are applied, source and destination are chosen randomly. Only it is not allowed for a node to be source and destination for the same route, since we want to analyze routing in the network. Constant Bit Rate has been chosen for traffic to clearly separate the effects of the routing algorithm from effects caused by the MAC layer. 10 packets per flow and second has been chosen to generate significant traffic, but also to avoid congesting the network.

1 MBit per second is generated in the network. We chose 512 bytes as packet size for data packets. It has been shown that the predominant traffic on the backbone (90% of the packets, 85% of the bytes) uses packets with sizes ranging between 400 and 500 bytes [Abr99]. UDP packets with a size of 512 bytes are transmitted in our simulations.

We vary the fraction of malicious nodes in the network. Malicious nodes drop all data packets not related to them. Routing packets are processed compliant to the routing protocol.

8.3 Simulation Setup: Scenario B

In Scenario B we want to evaluate the quality of detection with varying the drop ratio of the malicious nodes. We use a small fixed topology of 6 nodes, in which one of the nodes is malicious. The topology used is shown in Figure 8.2. 2 flows are existing in the network: from A to B and from B to A. The malicious node drops data packets with a ratio from 0.0 to 1.0, increasing in 0.2 steps with every simulation setup. Further the malicious node is dropping packets only between 100s and 900s of total 1000s simulation time. This is done to see the time needed for detection and recovery. So in the worst case the malicious node drops 80% of the traffic, as it is only misbehaving in 80% of the time. With a smaller drop ratio, the fraction of packets droped in worst case (i.e. when all traffic passes the malicious node) is even less.

The main interest using this scenario is to find out which effect varying drop rates have on the quality of detection. In Table 8.3 we give a complete list of the parameters used.

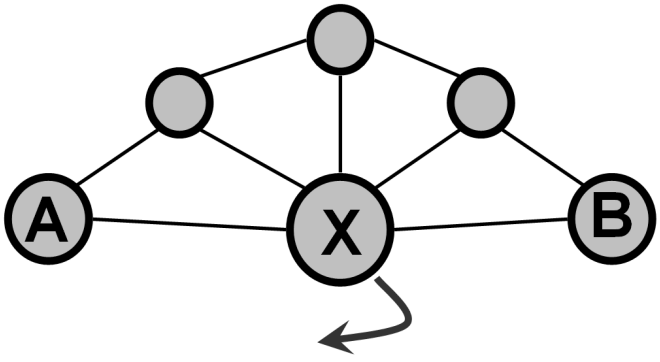


Figure 8.2: Topology for Scenario B: Two flows, from A to B and from B to A, pass the network. The node marked with X is malicious. As this malicious node is on the shortest path from A to B and also connected to all other nodes, the impact of its behavior is expected to be enormous.

Parameter	Value
Number of nodes	6
Radio range	627m
Placement	fixed topology, see 8.2
Movement	none, fixed positions
MAC	802.16 Mesh Mode
Sending rate per flow	10 packets per sec.
Application	Constant Bit Rate (CBR)
Number of flows	2
Flow pairing	fixed: $A \rightarrow B$, $B \rightarrow A$
Packet size	512 bytes
Route discovery time	100 s
Simulation time	1000 s
Number of replications	20
Varying parameter	drop rate of malicious node in the network
	Ratio: 0.0, 0.2, 0.4, 0.6, 0.8, 1.0

Table 8.3: Parameters for Scenario B

The following metrics are considered:

- Throughput, dropped data packets.

  This metric is important to measure the effect of varying the drop ratio of malicious nodes on the detection quality of WatchAnt. In this case it is interesting to see, up to which drop ratio detection quality is low.

- Detection time and recovery time

  The malicious node is only misbehaving from 100s to 900s of the total 1000s simulation time. By having a period of well behavior before and after the period of misbehavior we can see how long it takes the neighboring nodes in average to detect misbehavior. From 900s to 1000s the previously malicious node gets benign. Here we want to show how fast the recovery mechanism rehabilitates the previously malicious node.

8.4 Simulation Setup: Scenario C

In Scenario C we want to show the effects of colluding malicious nodes in the network. We chose a small fixed topology and placed 2 malicious nodes on the shortest path between the nodes A and B, which have flows to each other. In Figure 8.3 we present the topology. Flows exist between A and B. Nodes marked with X are malicious.

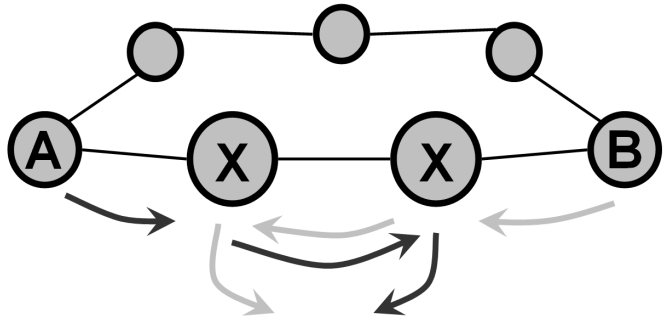


Figure 8.3: Topology for Scenario C: 2 flows, from A to B and from B to A, pass the network. Nodes marked with X are malicious nodes. They are on the shortest path between A and B.

In this scenario we simulate four cases:

- C.1) In the first case the nodes marked with X are benign, they are not malicious. The results from this case can be used as reference for the upcoming cases.
- C.2) In the second case the nodes marked with X are malicious (as in case 3 and 4, too). All security mechanisms in AntSec are disabled.
- C.3) The third case is similar to the second case: nodes marked with X are malicious. But here the security mechanisms of AntSec are enabled.
- C.4) In the fourth case the security mechanisms are also enabled, but in this case the malicious nodes collude. They drop all packets that are generated by a benign node and received from another malicious node. In this case the Leak Detector is enabled.

The complete list of parameters for Scenario C is denoted in Table 8.4.

Parameter	Value
Number of nodes	7
Radio range	627m
Placement	fixed, see Fig. 8.3
Movement	none, fixed positions
MAC	802.16 Mesh Mode
Sending rate per flow	10 packets per sec.
Application	Constant Bit Rate (CBR)
Number of flows	2
Flow pairing	fixed: $A \rightarrow B$, $B \rightarrow A$
Packet size	512 bytes
Route discovery time	200 s
Simulation time	1000 s
Number of replications	20
Varying parameter	See cases C.1 to C.4

Table 8.4: Parameters for Scenario C

The following metrics are considered:

- Throughput, dropped data packets.

  This metric is important to measure the effect of two things: First AntSec's detection quality demonstrated in a small topology. Second the effect of colluding behavior of malicious nodes on the detection quality of WatchAnt.

- Valid and invalid detects by the Leak Detector in case C.4.

  We want to evaluate the quality of detection of the Leak Detector. Hereby it is most important to see whether colluding malicious nodes are detected, and if they are detected how many valid detections are made in relation to the number of false positives.

8.5 Results: Scenario A

In this section we look at the results from the simulations related to Scenario A.

8.5.1 Throughput

A very important metric in the design of a secure routing protocol is the throughput. In a network with malicious nodes it should provide higher throughput than regular routing algorithms.

Figure 8.4 shows the throughput of AntSec, Ant and DSR in a network with varying fraction of malicious nodes. With 00% malicious nodes in the network all three routing algorithms provide throughput values very near to 100%. Ant and AntSec do not have reached 100% since the IEEE 802.16 Mesh Mode MAC Layer in the MobQoS project was concurrently implemented and occasionally collisions occured. These problems have been solved for DSR. So the results shown for Ant and AntSec are slightly penalized due to the incomplete implementation of the IEEE 802.16 Mesh Mode MAC layer. In the following states, with a fraction of malicious nodes ranging from 20% to 100%, throughput decreases. Ant and DSR have quite similar results differing not more than 5.2 percentage points. Ant distributes the traffic load by its multipath approach. By this approach it is more likely that also paths without malicious nodes are used.

Routing Algorithm	00%	10%	20%	30%	40%	50%
AntSec	97.882%	81.178%	65.347%	46.737%	35.248%	25.209%
Ant	97.294%	68.378%	48.650%	32.849%	26.982%	21.867%
DSR	100.0%	63.199%	44.399%	30.736%	23.199%	17.646%
Standard deviation						
AntSec	1.077%	11.790%	6.158%	8.194%	7.784%	6.074%
Ant	0.801%	11.884%	8.436%	7.507%	6.987%	6.193%
DSR	0.0%	13.893%	12.708%	10.332%	8.370%	8.007%

Table 8.5: Average throughput in Scenario A measured over 1000 seconds

The most important part in Figure 8.4 is that AntSec outperforms both Ant and DSR in throughput by up to 19 percentage points, as the Table 8.5 shows. Looking at the throughput in the last 100 seconds of the simulation, a time when AntSec had 900 seconds time to detect malicious nodes, we see a difference of up to 38.7 percentage points between AntSec and DSR. Here we show that our security mechanisms effectively detect malicious nodes in the network and exclude them from routing. In Table 8.6 the through-

put in the last 100 seconds is shown. Here we see clearly that the throughput of AntSec has increased over time, whereas the throughput using DSR or Ant show does not change much.

Routing Algorithm	00%	10%	20%	30%	40%	50%
AntSec	97.24%	85.57%	73.08%	50.46%	37.95%	24.74%
Ant	97.21%	68.32%	48.58%	32.80%	26.96%	21.87%
DSR	99.93%	63.16%	44.38%	31.39%	23.19%	17.26%

Table 8.6: Average throughput in Scenario A measured over the last 100 seconds of 1000 seconds simulation time

Throughput varies in each simulation as the standard deviation shows. In the case when 10% of the nodes in the network are malicious, the deviation has its maximum (up to 12,7 percentage points). Results vary as the topology of the network and the placement of the malicious nodes in it has a big effect on throughput. Malicious nodes in central positions result in high loss, whereas if placed at the edge of the network throughput will not decrease that strong.

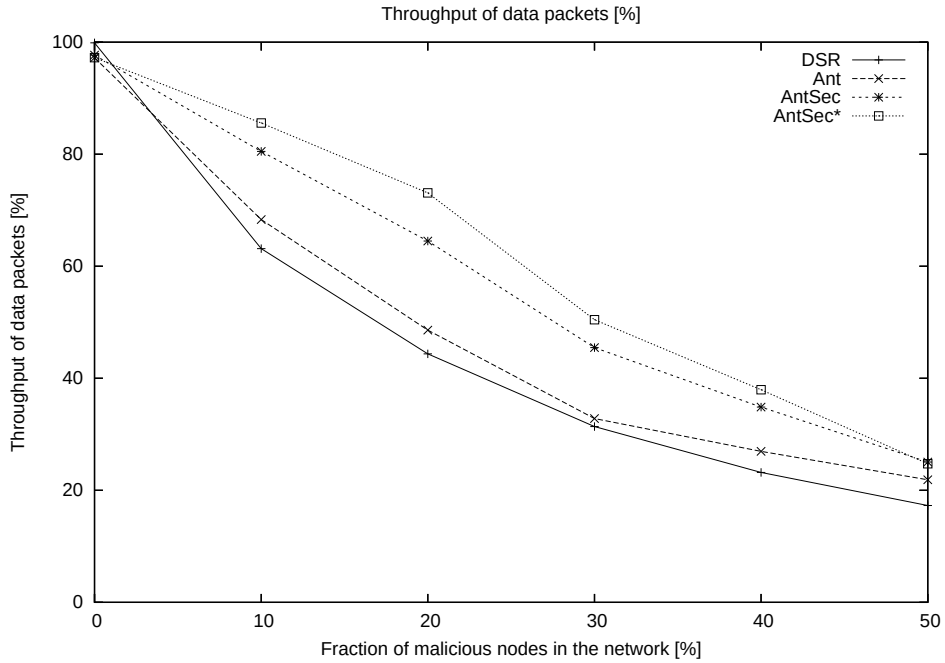


Figure 8.4: Throughput of data packets under varying fraction of malicious nodes in the network. DSR in comparison to Ant (security mechanisms disabled), AntSec (security enabled, average of throughput over 1000 seconds) and AntSec* (security enabled, average of throughput over the last 100 seconds of the simulation)

8.5.2 Overhead

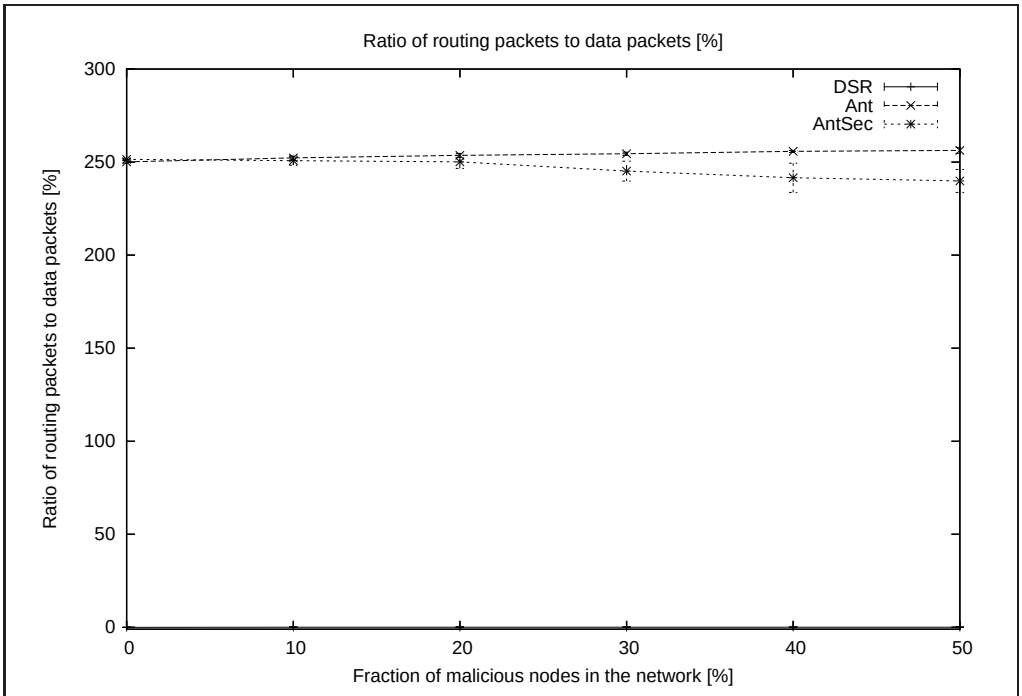
The Figures 8.5(a) and 8.5(b) show the traffic overhead measured in packets. In the first figure the number of routing packets is set in relation to the number of data packets transmitted. Hereby every transmission is counted, i.e. origination and forwarding of a packet. In the second figure the number of routing packets is set in relation to all packets transmitted in the network. DSR has almost no traffic overhead. This is due to the reactive character of DSR. Routes are only established when they are requested, routes are note maintained. Ant and AntSec are proactive routing algorithms; they discover routes to any known destination in the network. Additionally they maintain these routes by sending MFANTs. In DSR no route maintenance is done, routes are refreshed when they break. But as we do not simulate any movement or breaking links, no route breaks. DSR shows very good traffic overhead characteristics. Ant and AntNet produce quite a lot routing overhead. About 70% of the packets in the network are routing packets. Table 8.1 shows the parameters used for Ant and AntSec. For optimization of the traffic overhead the sending rate of DFANTs and MFANTs can be decreased.

The Figures 8.6(a) and 8.6(b) show the traffic overhead measured in bytes. In the first figure the amount of bytes transmitted for routing purposes is set in relation to the amount of data bytes transmitted (originated or forwarded). In the second figure the number of routing bytes is set in relation to all bytes transmitted in the network.

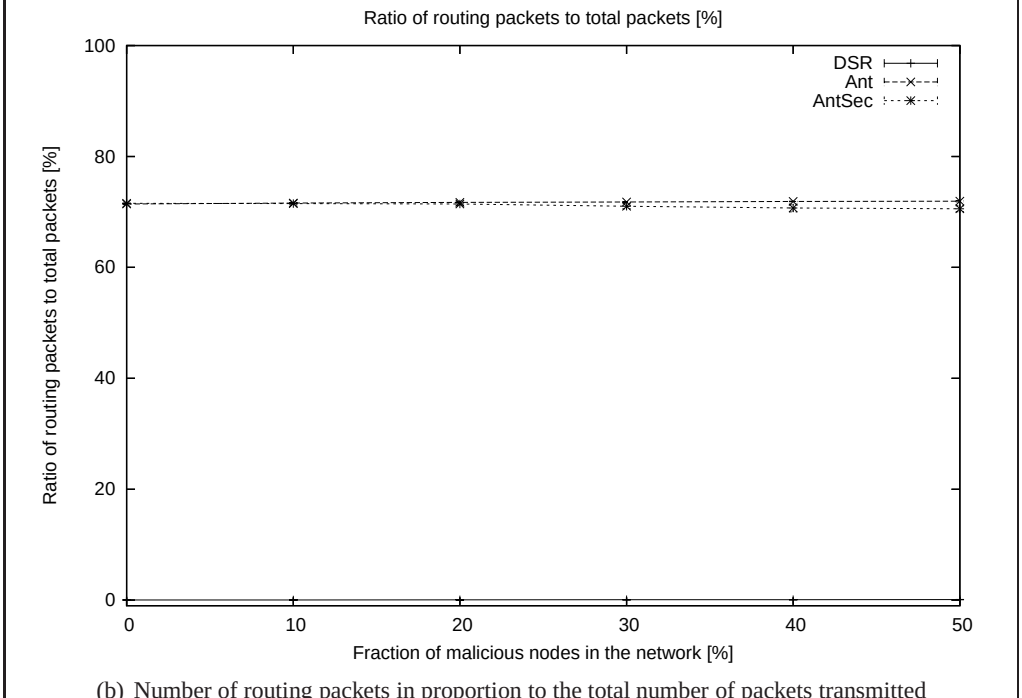
In Figure 8.6(b) it is shown that AntSec leads to about 35% traffic overhead in bytes. Ant results in about 20% traffic overhead in bytes. As the sizes of the Ants are optimized, traffic overhead can only be decreased by sending less routing messages or by sending more data messages.

The number of routing messages can be influenced with the parameters *AS_Discovery_Forward_Ant_send_interval* and *AS_Maintenance_Forward_Ant_send_interval* introduced in Table 8.1. They determine the sending interval of DFANTs and MFANTs.

Ant and AntSec are proactive routing protocols. With increasing of the intervals for sending of Forward Ants the traffic overhead resulting from the steady maintenance of routes is decreased. Nevertheless each proactive routing protocol causes a specific traffic overhead for route discovery and maintenance.

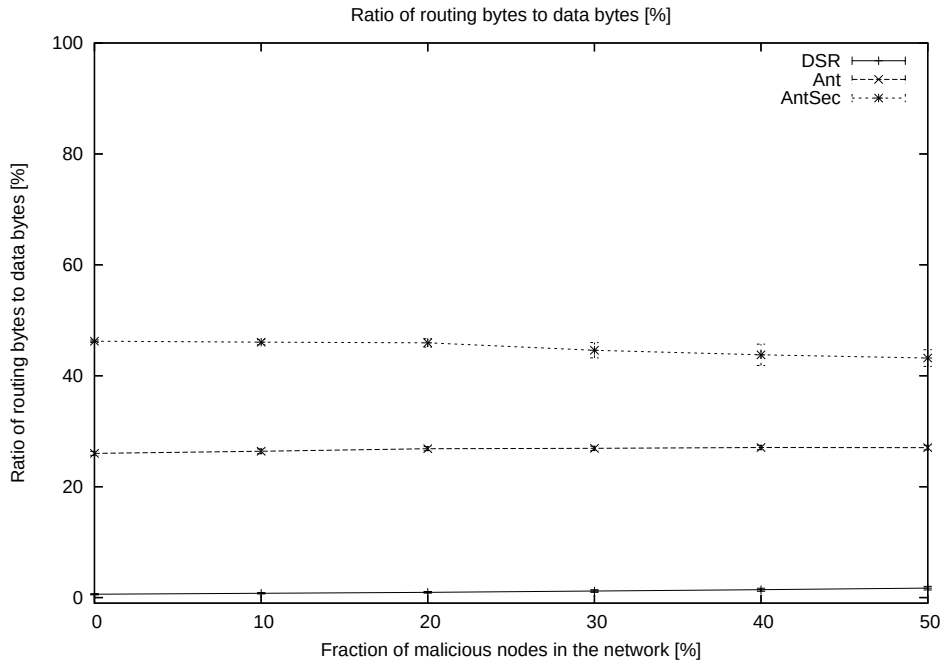


(a) Number of routing packets in proportion to the number of data packets transmitted

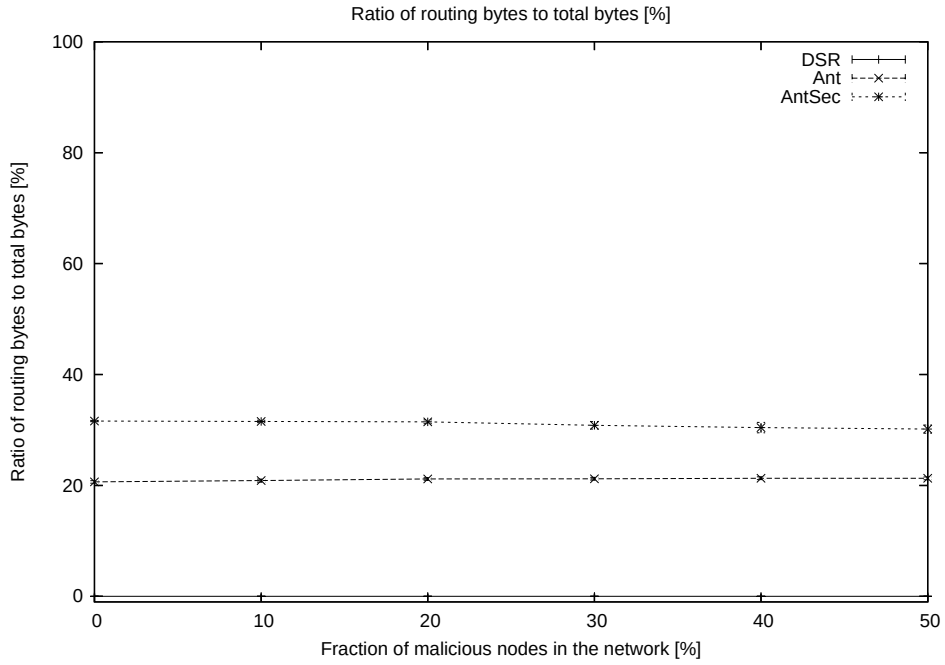


(b) Number of routing packets in proportion to the total number of packets transmitted

Figure 8.5: Traffic overhead measured in number of packets



(a) Number of bytes transmitted in routing messages in relation to the number of bytes transmitted in data messages



(b) Number of bytes transmitted in routing messages in relation to the total number of transmitted bytes

Figure 8.6: Traffic overhead measured in bytes

8.5.3 Average Number of Hops for Various Packet Types

Figure 8.7 shows the average number of hops for data packets. Routing packets have longer paths when using AntSec. As routing packets are not forwarded by malicious nodes, benign nodes may decide to circumvent malicious nodes in routing. Data packets are able to use longer paths when using AntSec. The average hop count for data packets is 1.308 hops higher in average when using AntSec instead of Ant.

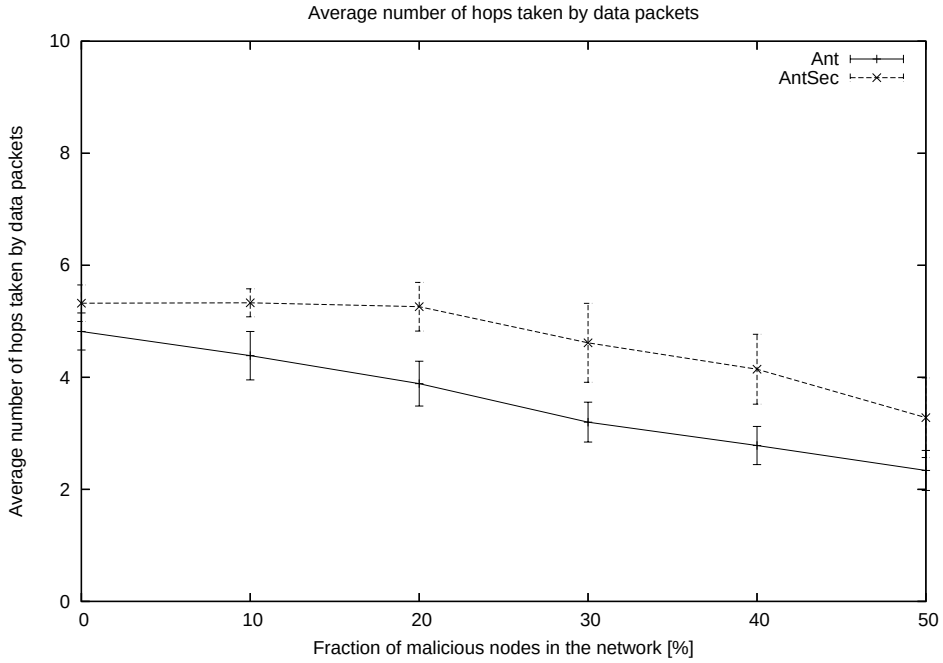
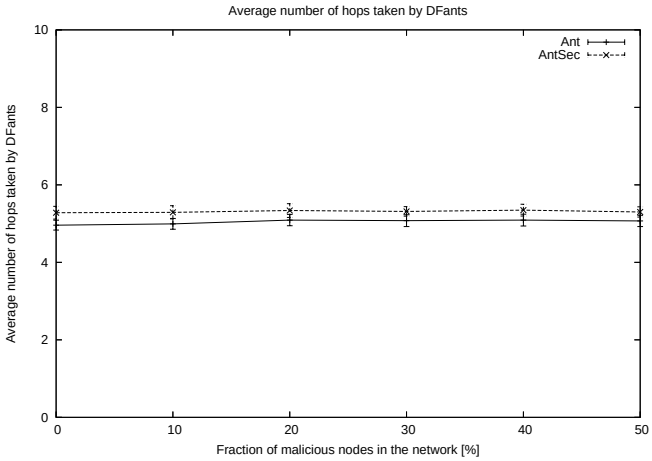


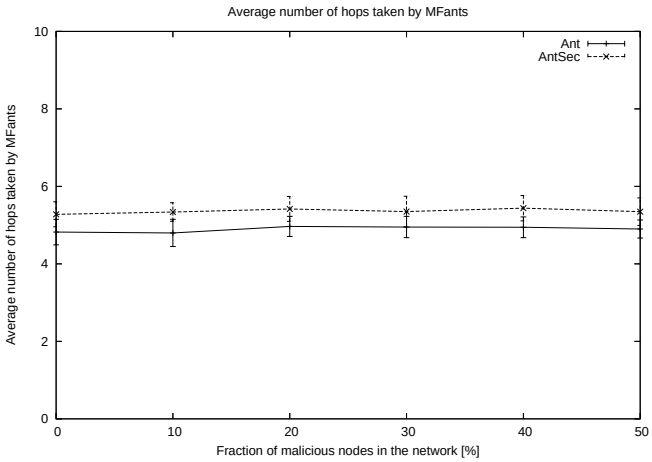
Figure 8.7: Average number of hops taken by data packets

Figures 8.8(a), 8.8(b) and 8.8(c) show the average number of hops for Discovery Forward Ants, Maintenance Forward Ants and Backward Ants. As routing packets are processed by malicious neighbors their average hop count is only affected by the benign nodes sending these routing packets to nodes not considered as malicious. The average hop count for MFANT is 0.56 hops higher when using AntSec instead of Ant. MFANTs are interesting as they are only transmitted in routes that are actually used and in which WatchAnt can detect misbehavior.

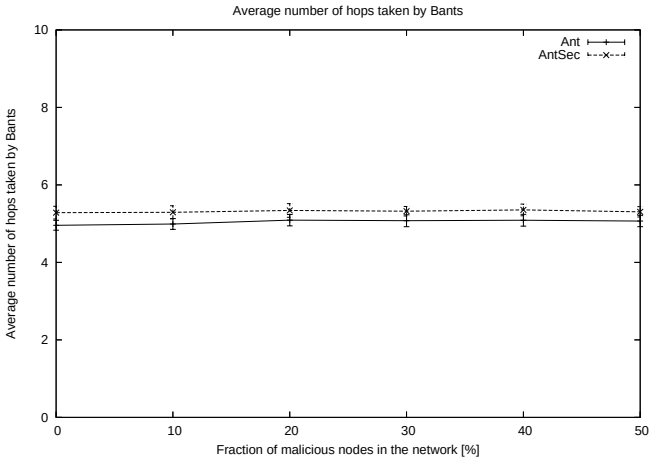
With AntSec the length of the routes increases, malicious nodes are circumvented. This effect is demonstrated in Figure 8.9 with the amount of packets dropped for reaching their maximum hop count. With increasing number of malicious nodes in the network and



(a) Average number of hops taken by Discovery Forward Ants



(b) Average number of hops taken by Maintenance Forward Ants



(c) Average number of hops taken by Backward Ants

Figure 8.8: Average number of hops taken by Ants

with detecting these malicious nodes, these nodes are not chosen as next hops in routing. By this routes get longer and may contain loops. The number of packets, which do not arrive at their destination but are dropped due to their maximum hop count limitation, rises. In DSR loops can be avoided, as the source node determines the route in advance. Loop-free routing protocols require that to any packet, that is routed through the network, it can be determined which path it traversed to avoid loops. In distributed routing algorithms there are no such concept.

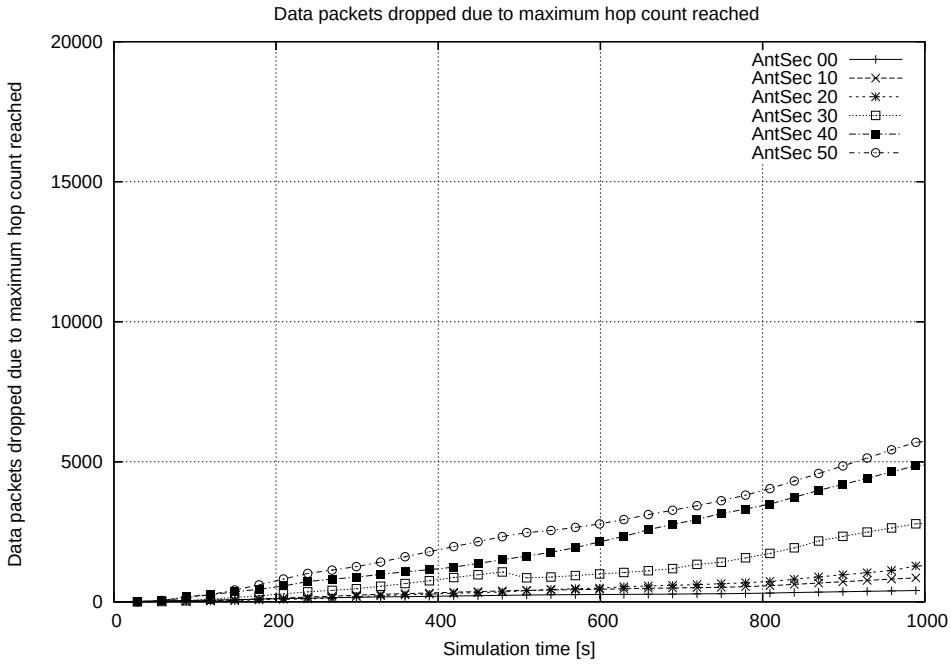
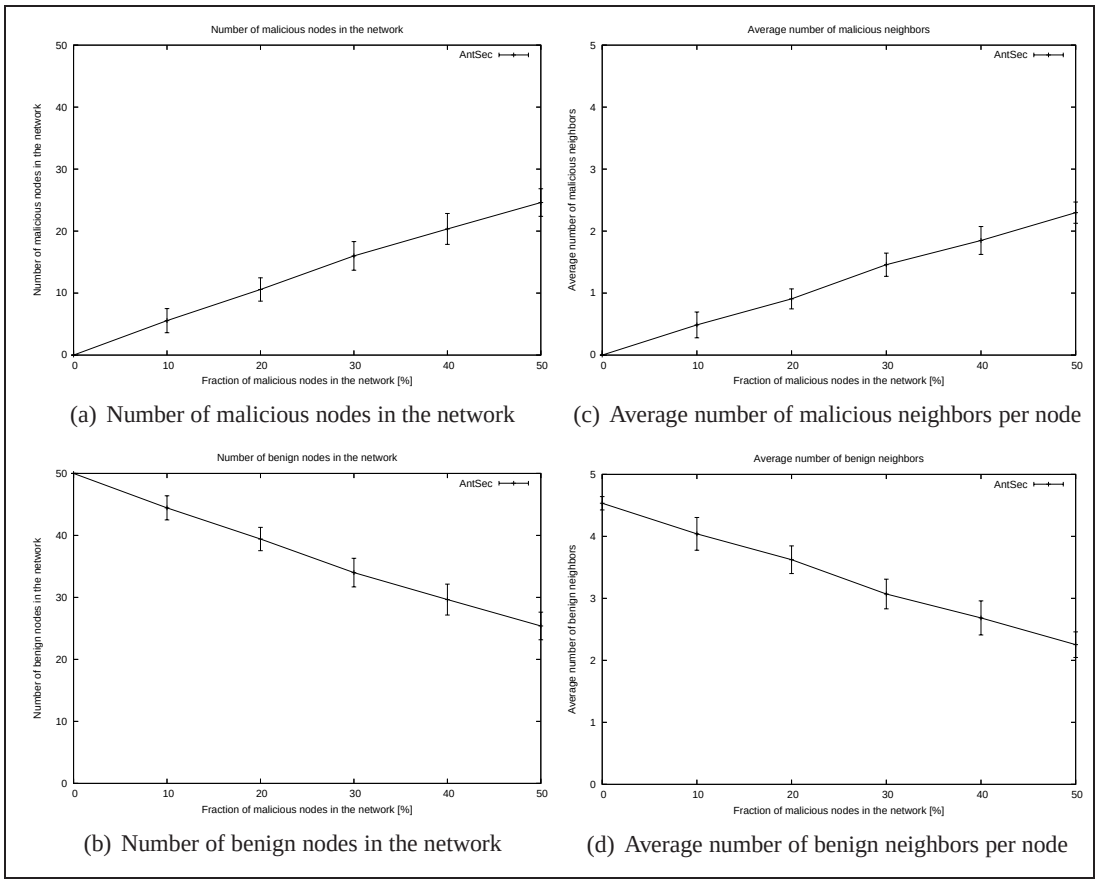


Figure 8.9: Data packets dropped over time due to maximum hop count reached. 250 packets originated per second.

8.5.4 Detection Quality of AntSec

Figure 8.10 shows the number of malicious nodes in the network. With increasing the number of malicious nodes in the network, the average number of malicious neighbors in the network is also raising. Related to this, the number of benign nodes and the average number of benign neighbors in the network is shown in Figure 8.10, too.

Figure 8.10 is introduced as reference for the upcoming figures, in which the number of malicious/benign nodes/neighbors should be kept in mind.



(a) Number of malicious nodes in the network

(c) Average number of malicious neighbors per node

(b) Number of benign nodes in the network

(d) Average number of benign neighbors per node

Figure 8.10: Characteristic numbers of malicious and benign nodes in the network (50 nodes in total)

Figure 8.11 shows the ratio of malicious nodes detected by at least one neighbor during simulation time. For the malicious node ratio 10% to 40% detection quality is high, in each case higher than 95%. So a great part of the malicious nodes is detected by at least one neighbor. The observations made could be communicated using the second hand reputation mechanism, to increase the detection quality of WatchAnt.

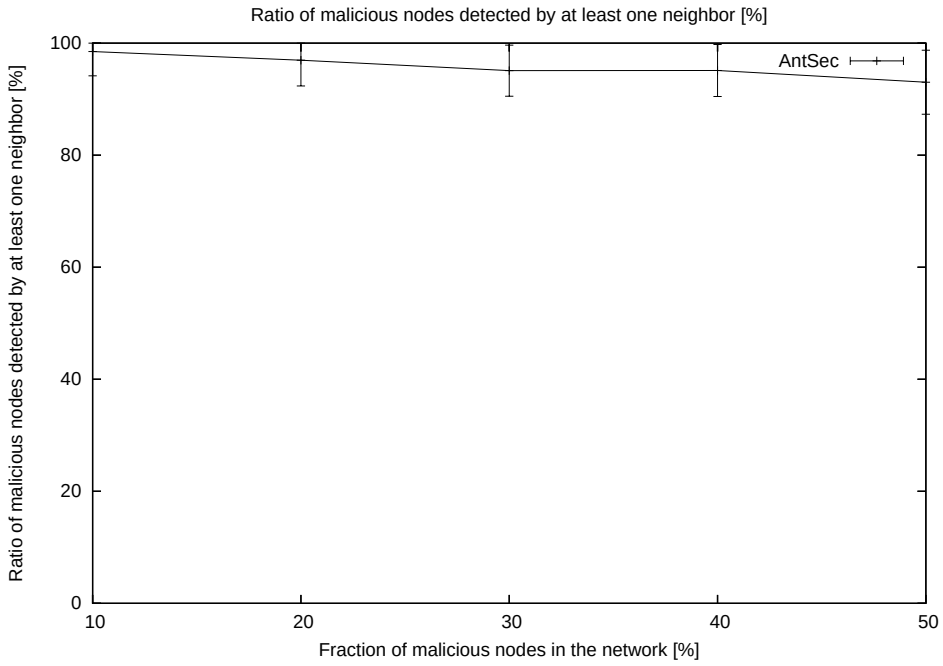


Figure 8.11: Ratio of malicious nodes detected by at least one neighboring node

The changes of the ratio of malicious nodes, detected by at least one neighboring node, are shown in Figure 8.12 and 8.13.

The results in Figure 8.13 are very interesting for our needs. More than 50% of all malicious nodes are permanent suspected of being malicious by one of their neighbors. This again in combination with the results of throughput in Figure 8.4 show that the usage of only direct observation is quite good, but improvements in throughput are still possible if the observations of neighboring nodes are combined. The detection ratio does not increase above 70% in average. Various reasons having effect on the detection quality are hindering. For example malicious nodes in the network which are not utilized for routing, are not detected as misbehaving. Furthermore, malicious nodes are not constantly suspected as malicious. Since reputation recovers, they are utilized again, at least until detected as misbehaving again.

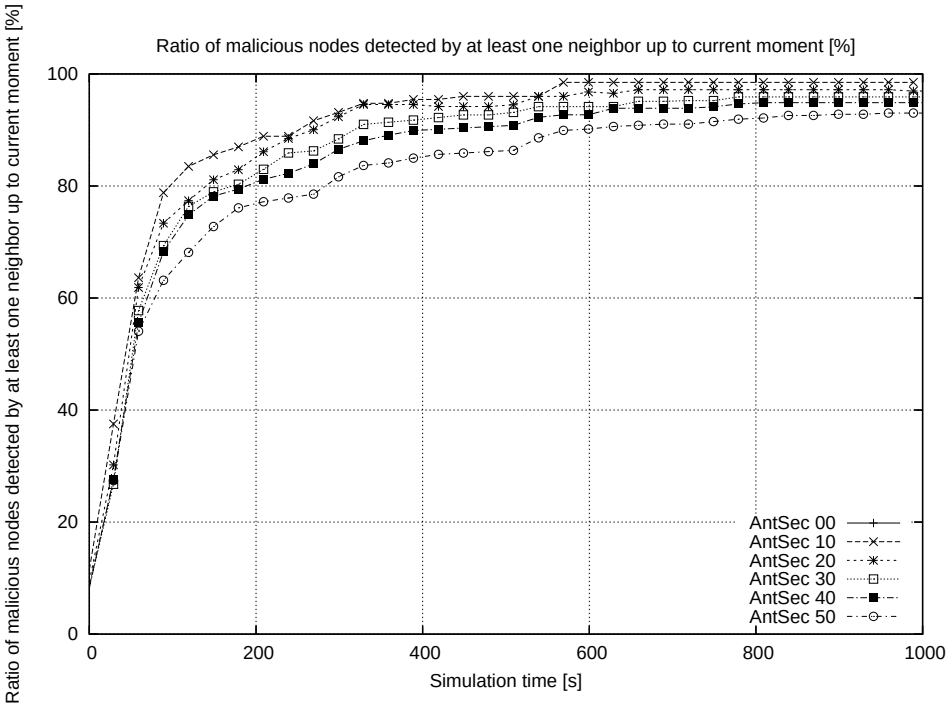


Figure 8.12: Ratio of malicious nodes detected by at least one neighbor until the simulation time listed in the x axis.

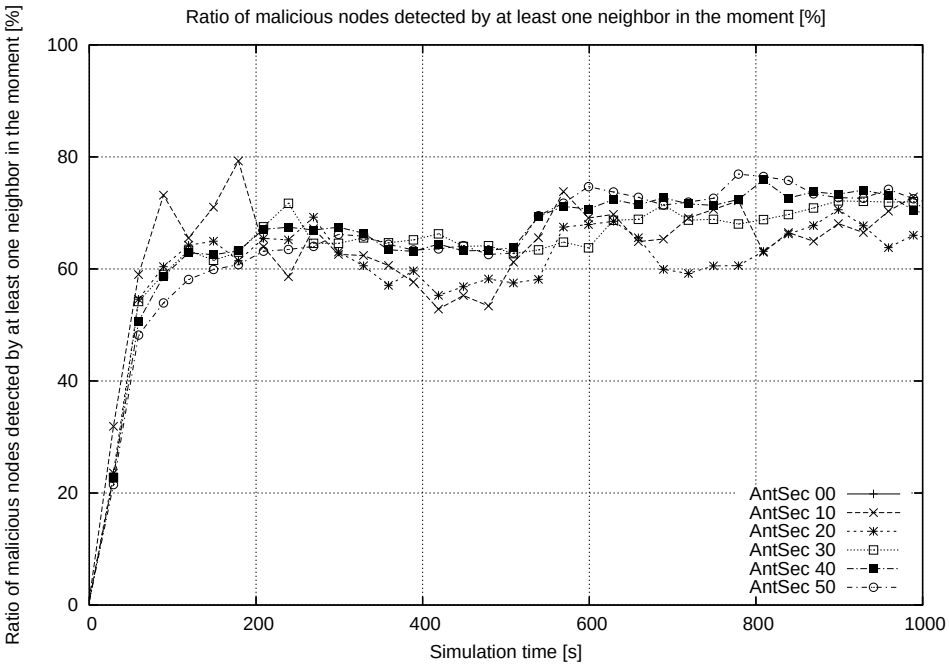


Figure 8.13: Ratio of malicious nodes detected by at least one neighbor in the denoted moment of the simulation.

Figure 8.14 shows the ratio of nodes detecting a malicious neighboring node. Two points have effect on this graph. First: nodes at the edge of the network or on places where no traffic is, are unable to detect misbehaving neighboring nodes. This is because malicious nodes drop only data packets, which would not pass in this case.

The second point is, that malicious nodes do not send WatchAnt Requests. With these they would be able to detect misbehaving neighbors. But as malicious nodes do not forward any data packets, it would make no sense to check neighboring nodes, since they were not utilized for forwarding data packets in the past.

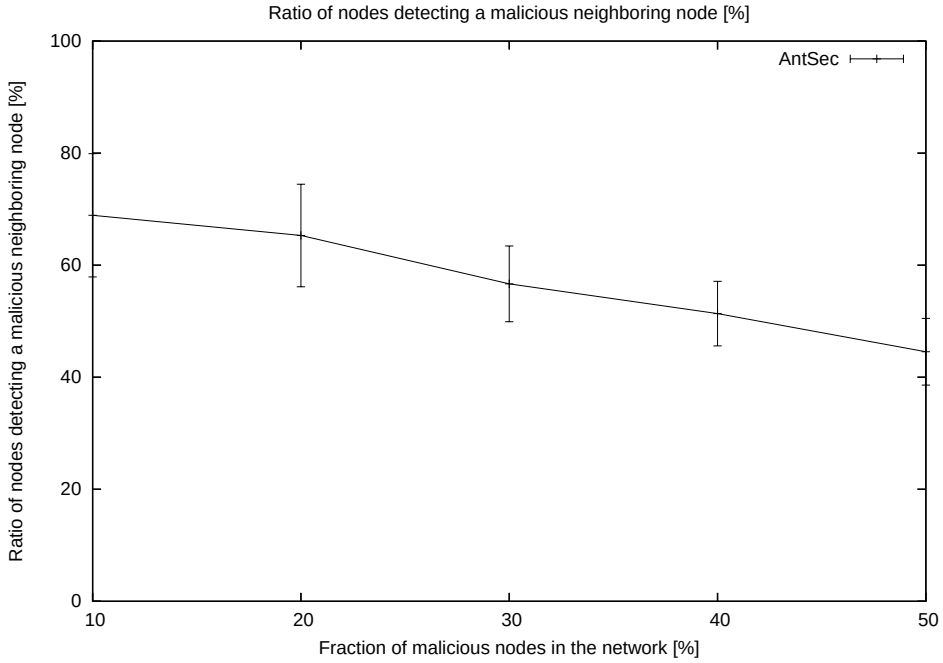


Figure 8.14: Ratio of nodes detecting a malicious neighboring node. Note: Malicious nodes do not try to find out whether neighboring nodes are misbehaving. The WatchAnt mechanism is ineffective when used by malicious nodes,as they do not forward any data packets. Therefore they cannot check forwarding behavior of their neighbors.

Figure 8.15 demonstrates the ratio of malicious nodes detected by their neighboring nodes. Here each node-node link is counted once. We see that in only 21.9% of all neighbor-links a malicious node is detected, and this is the best case. In the cases for malicious node ratio 0.3, 0.4 and 0.5 the ratio is only 6.8%, 4.12% and 2.21%. This is quite low, but for AntSec it is sufficient to detect the malicious nodes on the current route and avoid malicious nodes in future routing. Malicious nodes not utilized for routing or laying on the edge of the network are hardly to detect, as they do not have big impact on routing. With increasing ratio of malicious nodes in the network, throughput decreases

and some malicious nodes are not utilized anymore in routing, in this case they cannot be detected as malicious.

Comparing Figure 8.11 and Figure 8.15 it is obvious that a big amount of malicious nodes is detect by at least one neighbor, but all in all only a small fraction benign nodes detects every malicious neighbor. With the influence of second hand reputations the detection quality is expected to rise.

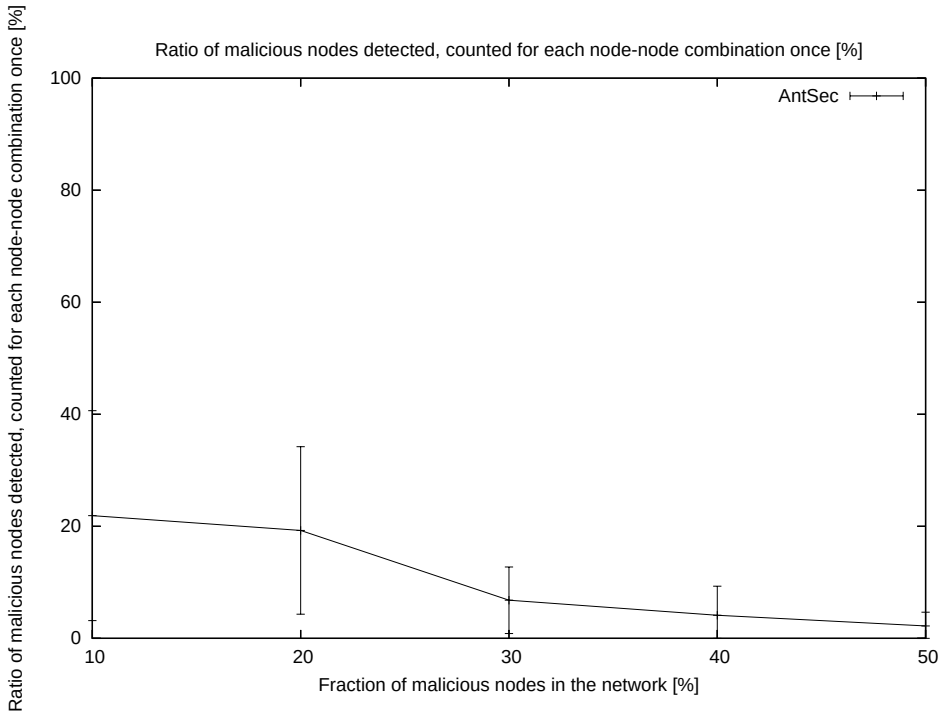


Figure 8.15: Ratio of malicious nodes detected by its neighboring nodes. This means that the number of neighbors of malicious nodes, which detect the misbehaving of the malicious node, is set in relation to the total number of neighbors of all malicious nodes. A neighbor-malicious node combination is counted once in this relation.

We present in Figures 8.16 and 8.17 the throughput of data packets over simulation time for the varying simulation parameters in Scenario A. In addition to this in the Figures 8.18 and 8.19 the loss ratio of data packets is shown.

In these figures we see that throughput rises over time. This comes from the fact, that malicious nodes in the network are detected and not longer utilized for routing by the detecting node. Standard deviation in this figure is still significant. Various parameters have effect on it. First the changes of the throughput over time is taken into account for each simulation run. Additionally topology, the placement of the nodes and flows are chosen randomly for each of the 20 simulation runs, so that various setups may exist

benefiting either the malicious nodes or the benign nodes. Still, the differences in the total throughput for all simulation runs related to a fraction of malicious nodes are not so big, as the standard deviations for the total throughput show in Table 8.5.

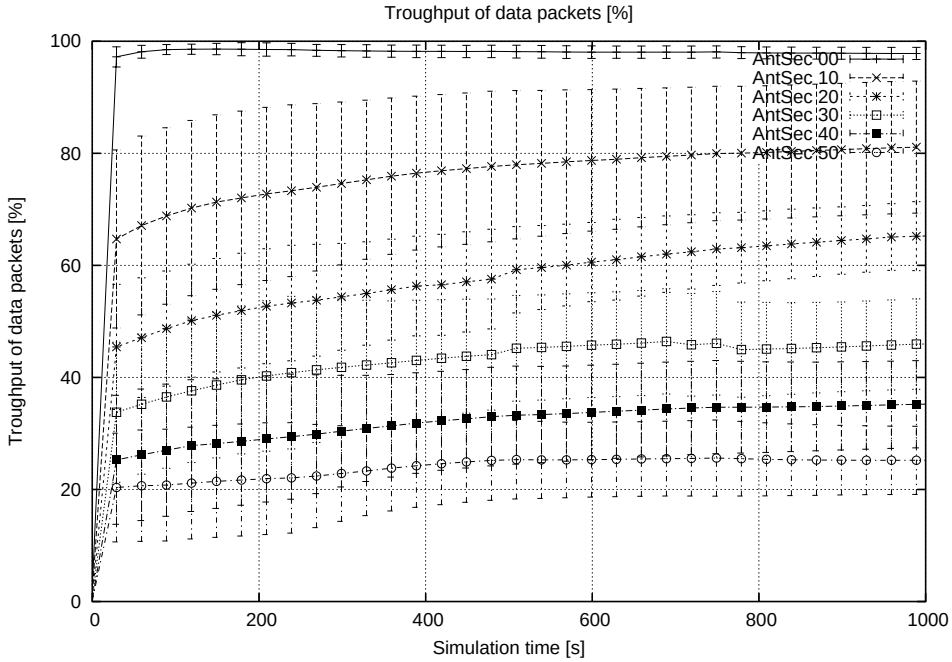


Figure 8.16: Throughput of data packets over time under varying fraction of malicious nodes in the network. With error bars.

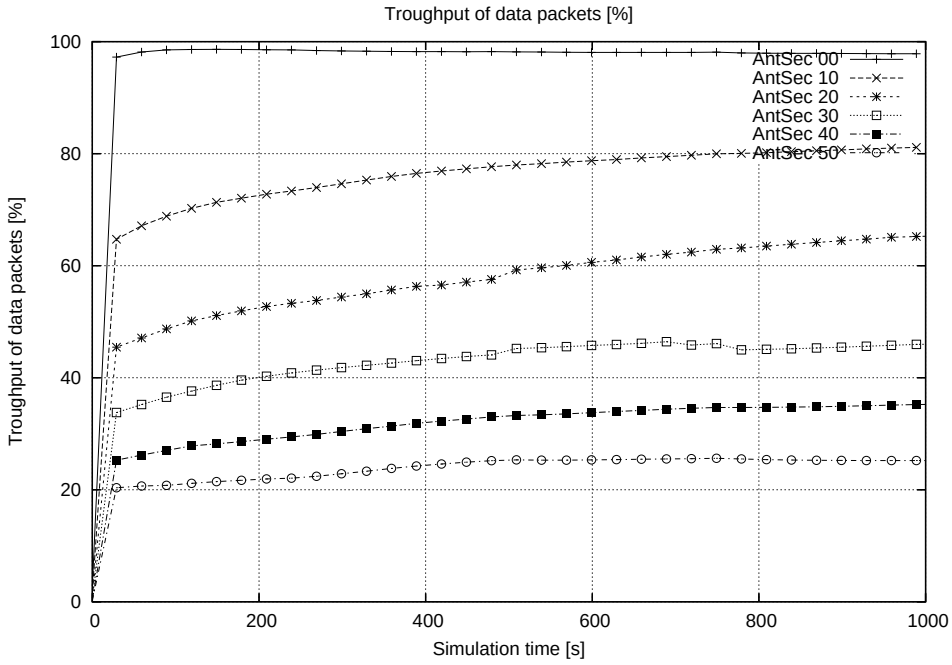


Figure 8.17: Throughput of data packets over time under varying fraction of malicious nodes in the network. Without error bars.

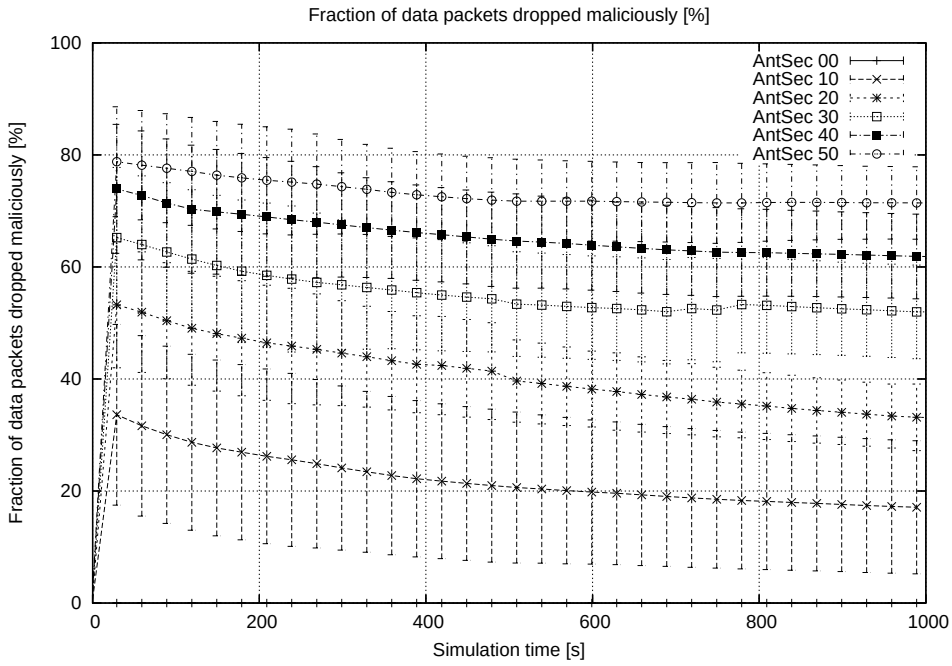


Figure 8.18: Fraction of data packets dropped maliciously over time. With error bars.

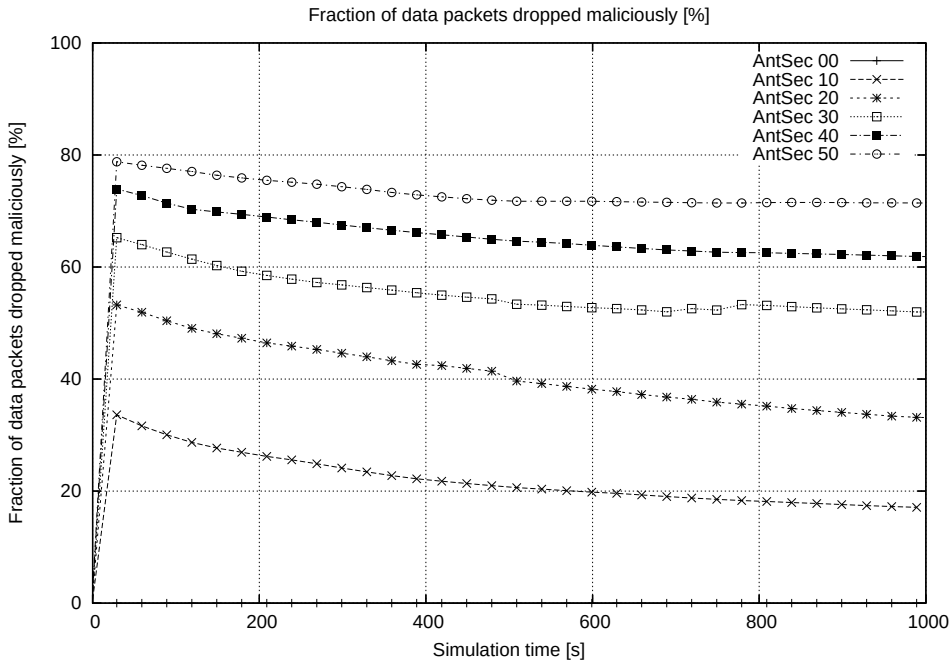


Figure 8.19: Fraction of data packets dropped maliciously over time. Without error bars.

8.5.5 Detailed Analysis of the Throughput of Data Packets over Time

In the following we discuss Figures 8.20 to Figure 8.25. They show the throughput in Scenario A over time. For each ratio of malicious nodes in the network, that we have chosen, one figure is showing the throughput for Ant, AntSec and DSR. Each point in the graph shows the number of received packets in relation to the number of sent packets up to the current simulation time.

We see that the throughput of Ant and DSR are quite constant, i.e. not changing over time, whereas the throughput of AntSec raises. AntSec and its security components detect misbehavior and avoid malicious nodes in routing. With time more and more malicious nodes are detected and excluded from routing. Therefore the throughput ratio raises over time when using AntSec. This is seen in comparison of Table 8.6 and Table 8.5, too. The first table shows the total throughput for Ant, AntSec and DSR. The second shows the throughput for the last 100 seconds of simulation time. In the last 100 seconds the throughput in the network if using AntSec is up to eight percentage points higher than the average throughput.

With DSR throughput ratio is constant. Routes are established once and remain established until the end of the simulation, as there is no mobility to break old routes and no detection mechanism exists to have effect on routing.

The throughput ratio of Ant is changing only little over time. As malicious nodes participate during route establishment and maintenance (i.e. forward routing messages), misbehavior is not detected. Still slight changes in the average throughput ratio exist, as Ant is probabilistic. By this probabilistic approach the routing entries vary over time and with them vary the paths chosen. But these changes are small-sized as in Ant the routing probabilities converge in a static network to specific values (0 or 1).

8.5.6 Conclusion of Scenario A

We have shown in Figure 8.4 that AntSec provides higher throughput than Ant and DSR. This is due to the abilities of our security framework to detect misbehavior. Malicious nodes are detected and excluded from routing. Because of this AntSec uses longer routes than Ant, as seen in Figure 8.7. AntSec is a proactive routing protocol constantly causing traffic for route discovery and maintenance. This is seen in Figures 8.5(b) and 8.6(b).

Looking at the ratio of packets transmitted by AntSec, 43.2% to 46.2% of the total number of packets are used for AntSec. Looking at the bytes transferred for routing, AntSec took 30.1% to 31.5% of the total number of bytes transmitted.

Figure 8.13 shows that permanently at least 50% of the malicious nodes are detected. After 600s more than 90% of the malicious nodes has been detected by at least one of its

neighbors.

Finally Figure 8.16 demonstrates that the throughput ratio of Ant and DSR are quite constant, as no detection mechanisms exist and topology does not change. In AntSec throughput raises as malicious nodes are detected and excluded from routing.

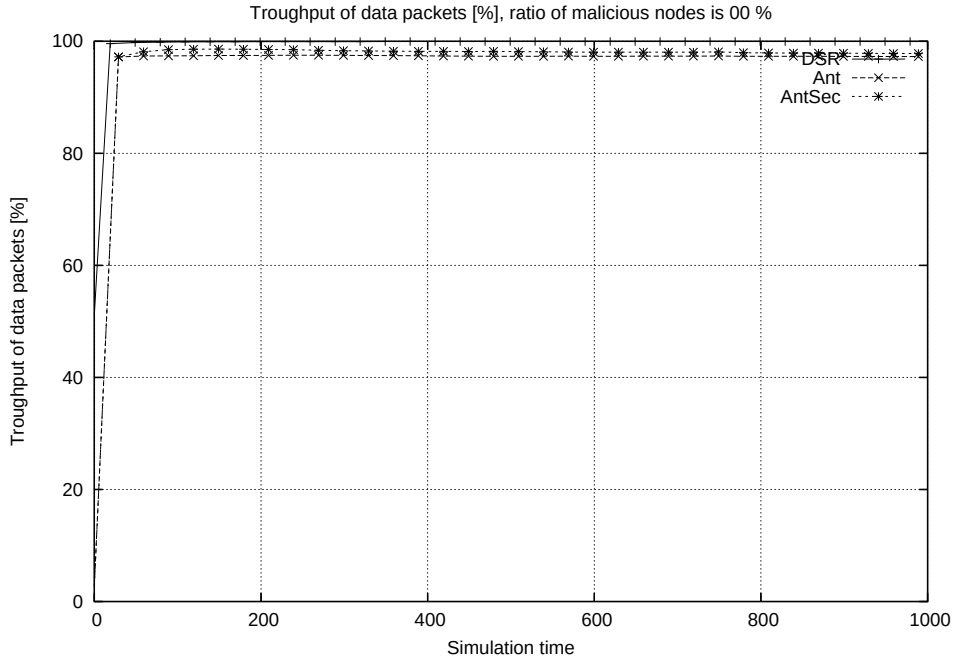


Figure 8.20: Throughput of data packets in a network with 0% of the nodes being malicious. Comparing the performance of DSR, Ant and AntSec

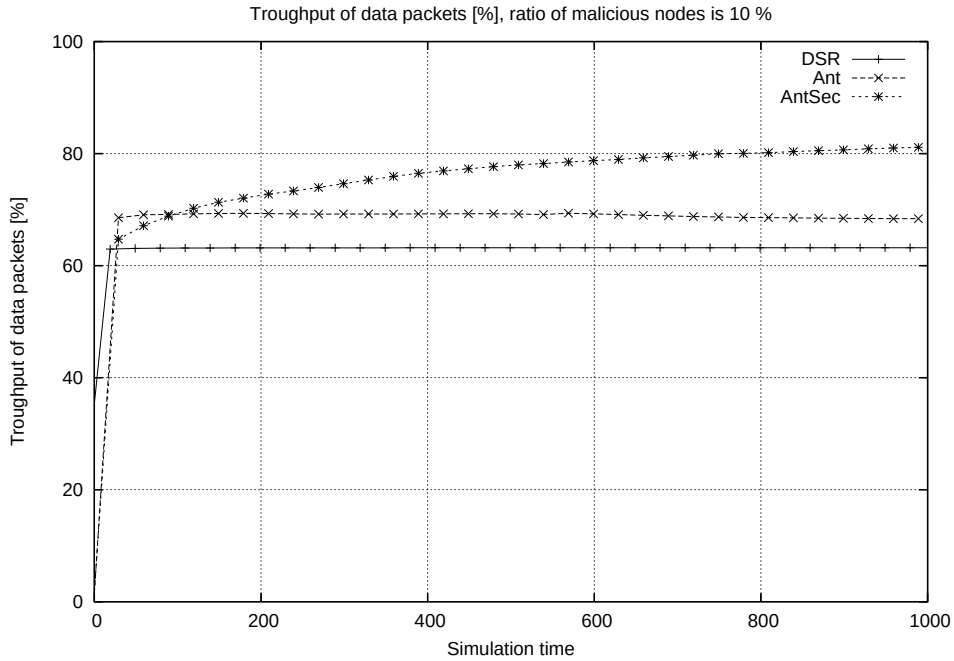


Figure 8.21: Throughput of data packets in a network with 10% of the nodes being malicious. Comparing the performance of DSR, Ant and AntSec

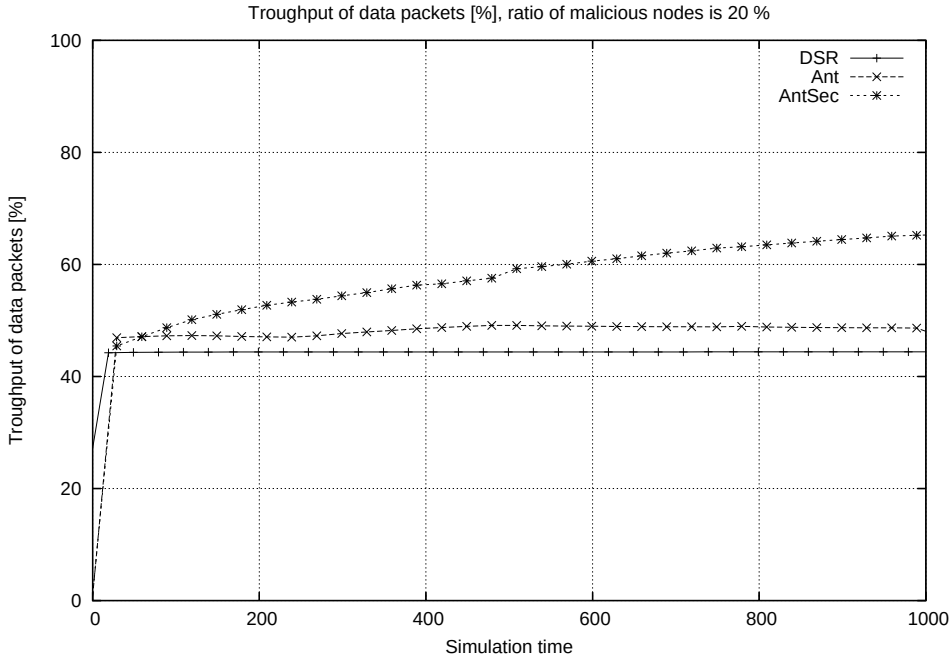


Figure 8.22: Throughput of data packets in a network with 20% of the nodes being malicious. Comparing the performance of DSR, Ant and AntSec

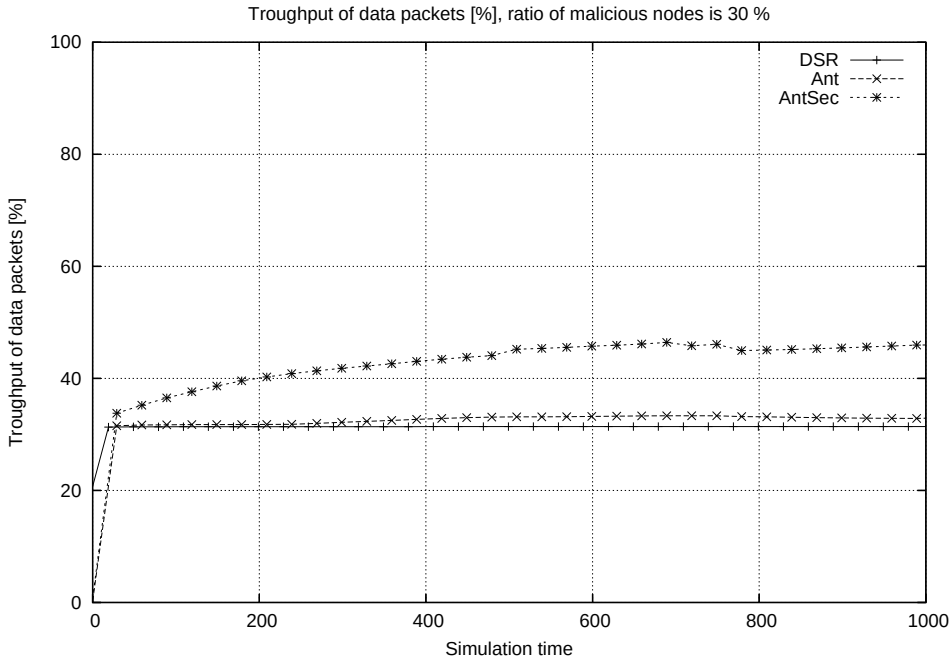


Figure 8.23: Throughput of data packets in a network with 30% of the nodes being malicious. Comparing the performance of DSR, Ant and AntSec

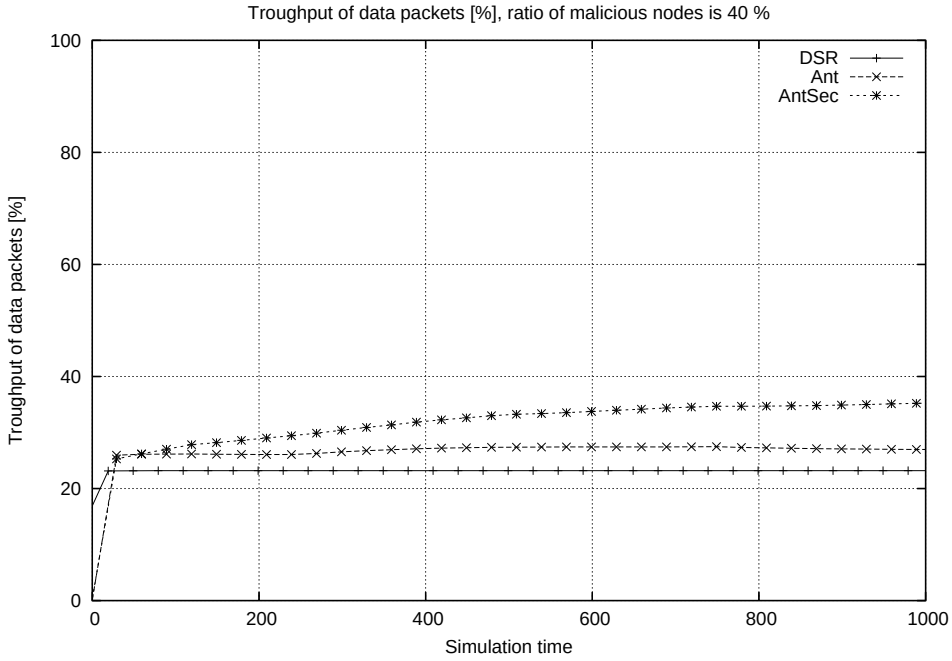


Figure 8.24: Throughput of data packets in a network with 40% of the nodes being malicious. Comparing the performance of DSR, Ant and AntSec

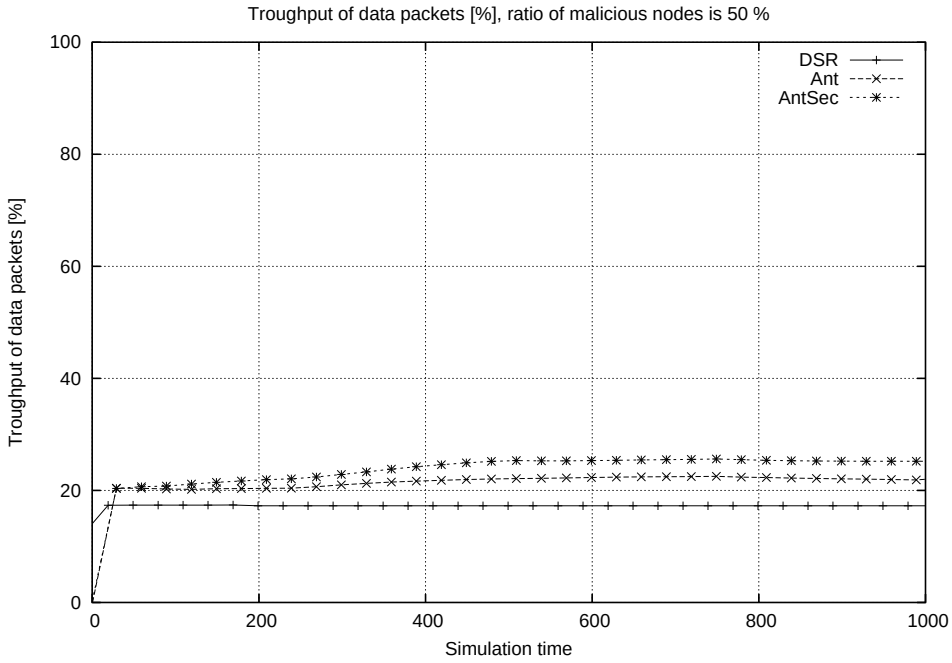


Figure 8.25: Throughput of data packets in a network with 50% of the nodes being malicious. Comparing the performance of DSR, Ant and AntSec

8.6 Results: Scenario B

8.6.1 Throughput in Scenario B under Varying Drop Rate

In Scenario B we used 100s for discovery and additional 100s for net entry, so the traffic flows start at 200s and last until 1200s. The malicious node in the central position drops packets in the simulation interval 300s to 1100s and behaves benign between 200s and 300s and 1100s and 1200s.

Figure 8.26 shows the throughput of the data packets simulated in the topology shown in Figure 8.2.

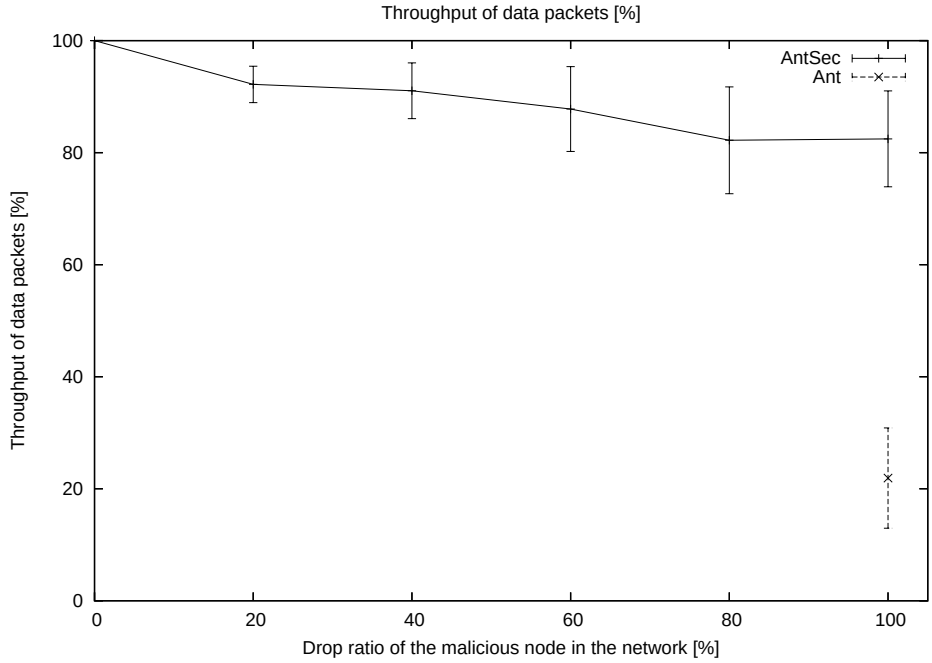


Figure 8.26: Throughput of data packets under varying drop ratio of the single malicious nodes in the network. We added for comparison the throughput results for using Ant (i.e. using no security features) in the this scenario. For Ant we used the drop ratio 1.0. The malicious node is dropping data packets in 800 of the 1000 seconds simulation time.

The single data set is showing throughput when using Ant and a drop ratio of 1.0. This result demonstrates that the topology is chosen well: (almost) both traffic flows, from A to B and from B to A, are passing the malicious node, when using no security mechanism. The malicious node drops (almost) 80% of the traffic as the node is malicious during 80% of the simulation time.

The connected data sets of Figure 8.26 show the throughput when using AntSec. The labels 00 to 100 show the drop ratio percentage of the malicious node. Throughput values are higher than 80% in average when using AntSec. With malicious behavior (drop ratio 0.2 to 1.0) the throughput of AntSec varies by 10 percentage points, as Table 8.7 shows. In the table we listed additionally the minimum throughput, which is guaranteed. Since the malicious node drops only in 80% of the simulation time and his drop ratio is varying, this defines a minimum ratio of throughput. This leads to the assumption that the drop ratio of malicious nodes does not have big effect on detection quality of our security framework.

Routing algorithm Drop ratio	AntSec 00%	AntSec 20%	AntSec 40%	AntSec 60%	AntSec 80%	AntSec 100%	Ant 100%
Throughput	100.00 %	92.18%	91.05%	87.79%	82.21%	82.46%	21.90%
Standard Deviation	0.00%	3.25%	4.97%	7.57%	9.53%	8.56%	8.95%
Guaranteed Throughput	100.00%	84.00%	68.00%	52.00%	36.00%	20.00%	20.00%

Table 8.7: Throughput in Scenario B

8.6.2 Detection Quality in Scenario B

Figure 8.27 shows the number of nodes in the network, which were detected as malicious. The figure shows that in average 4.3 to 4.55 of the nodes in the network detect the malicious node. This means that the drop ratio of the malicious node has no difference on the detection quality if the drop ratio ranges from 0.2 to 1.0. In all of these cases the number of neighboring nodes detecting the malicious node as misbehaving ranges between 4.3 and 4.55.

The effect of similar detection quality independent of the drop ratio of the malicious node is caused by the parameter *WA_Number_of_hashes_to_ask_for* which is presented in Table 8.1. This parameter is used during the creation of WatchAnt Requests. It describes the number of packets to which a report is requested. If one of these packets has not been forwarded, the whole WatchAnt Reply is false. For the simulations we have chosen *WA_Number_of_hashes_to_ask_for* = 12. And hereby it is not important whether 20% or 100% of these 12 packets has not been forwarded. Important is that the hashsum of the 12 packets sent in the reply is false. We have assumed that there is no loss of packets on MAC layer. Misbehavior of malicious nodes with a drop ratio of P_{drop} is detected with a probability of $1 - (1 - P_{drop})^{WA_Number_of_hashes_to_ask_for}$. This counts only if the neighboring malicious node is frequently utilized in routing. So with the parameter *WA_Number_of_hashes_to_ask_for* we can influence the detection quality of WatchAnt.

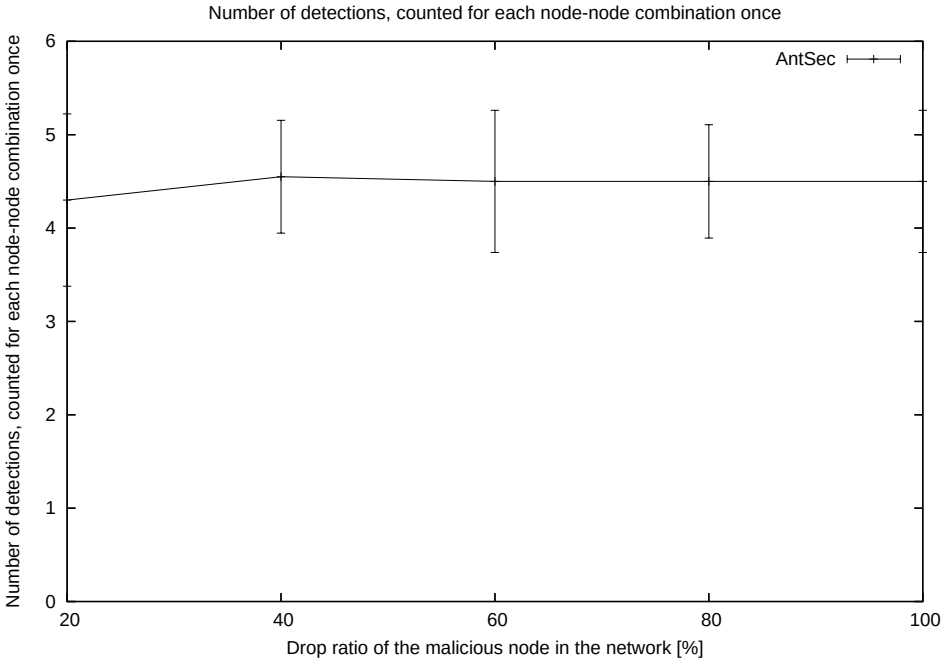


Figure 8.27: Number of detections in the network of Scenario B. Each neighbor of the malicious node, detecting the misbehavior of the malicious node, is counted.

Figure 8.28 shows the ratio of nodes in the network detecting a malicious neighbor. Here we can also see that the detection quality is very similar for all drop ratios. Still 100% is not reached. This comes from the fact that nodes are only rated if there are in use on a route. If one node is not utilized for forwarding by a neighboring node, this neighboring node cannot check whether forwarding is done compliant to the protocol. By this the reputation rises over time due to recovery and the malicious node is not detected as being malicious.

In Figure 8.29 it is shown that the ratio of malicious nodes detected by all of their neighboring node is varying between 50% and 60%. But complete detections are not needed to keep throughput at a high level. The detection of malicious nodes is only relevant for nodes which use these malicious neighbors in routing.

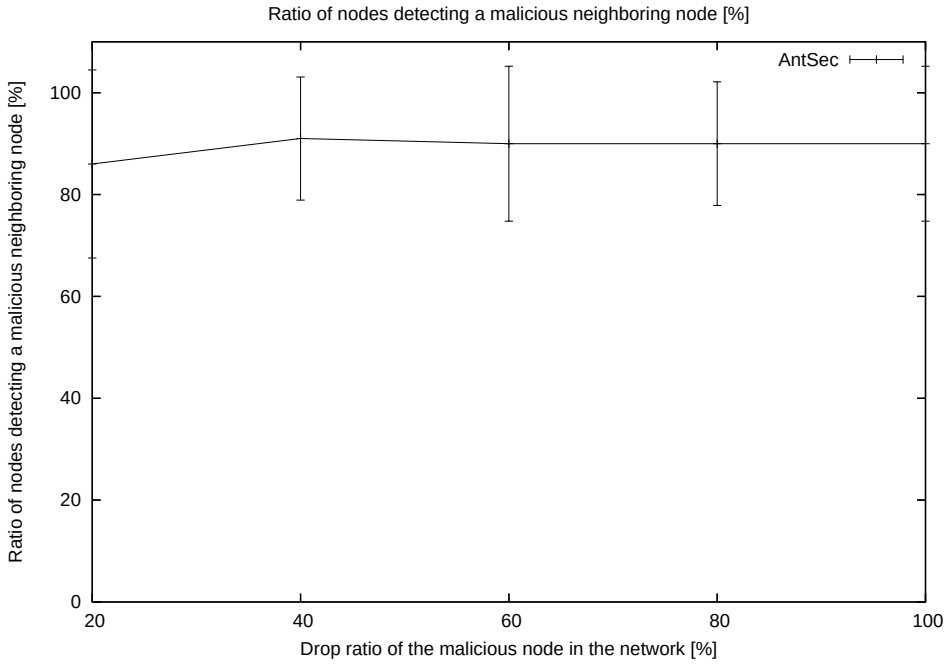


Figure 8.28: Ratio of nodes detecting a malicious neighboring node in Scenario B

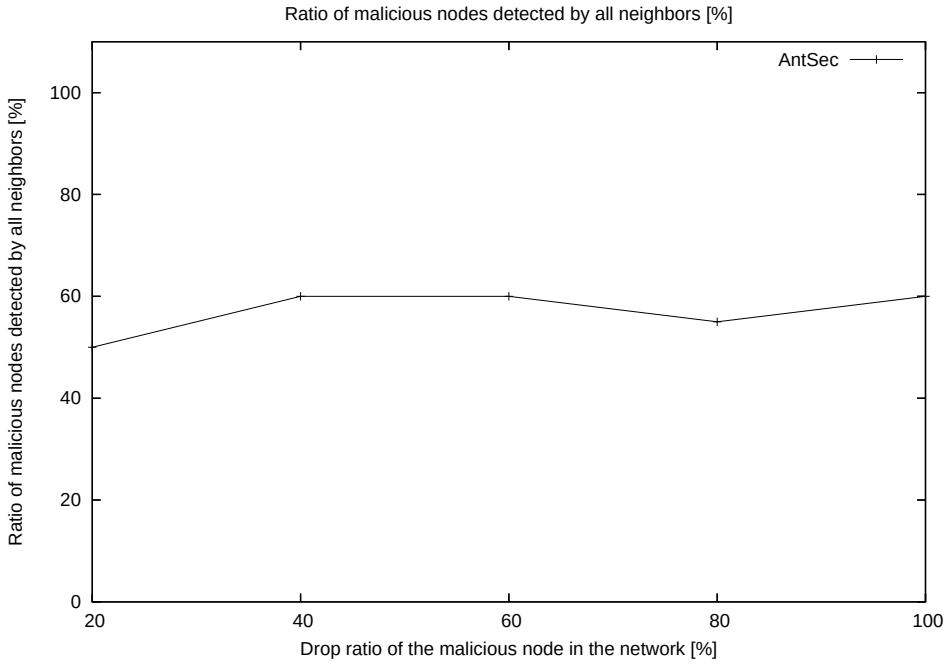


Figure 8.29: Ratio of malicious nodes detected by all neighboring nodes

8.6.3 Reputation Values of the Malicious Node in Scenario B

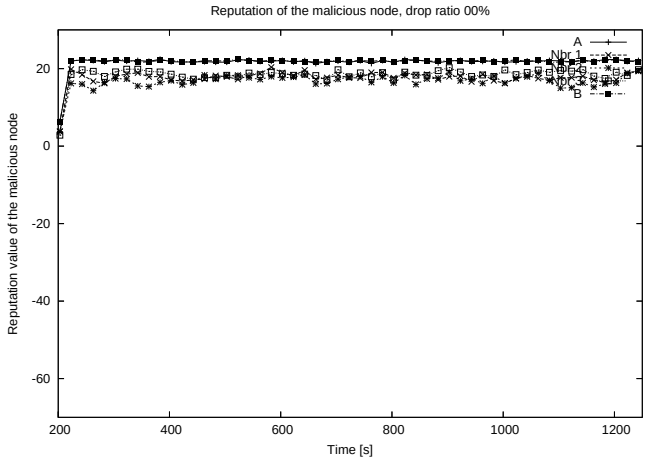
The reputation of the malicious node is shown in Figure-set 8.30 and Figure-set 8.31. These pictures show the detection quality of our security framework. As described in Subsection 8.6.2 detection quality for the drop ratios 0.2 to 1.0 are similar.

Figure-sets 8.30 and 8.30 show the detection speed of WatchAnt. The reputation value of the malicious node maintained by each one of its 5 neighbors decreases immediately after entering the time interval, in which the malicious node drops packets. Having a very bad reputation (i.e. -40 to -60) the detected nodes are not utilized in routing and no service of forwarding packet is provided for them by the detecting nodes. As they are not utilized they reputation can recover. After a short while they are detected as malicious again. So the state of being detected fluctuates. This can be seen very good in Figure 8.30(b) in the figure-set for neighbor 2 between simulation time 600s and 1000. When the malicious nodes stops dropping packets (from 1100s on) its reputation recovers. 100 seconds are enough to gain a very good (i.e. +20) reputation again.

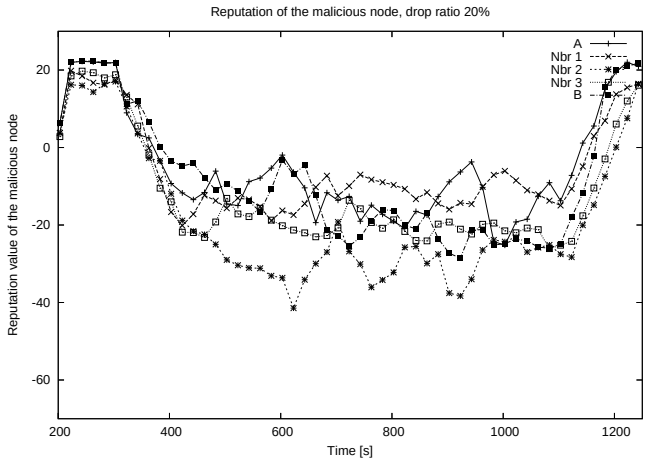
8.6.4 Conclusion of Scenario B

For Scenario B we have chosen a fixed topology with a malicious node in a central position (see Figure 8.2). Without any security mechanism both flows we have injected are passing this node (as shown in Figure 8.26 with the results for Ant). AntSec provides good detection quality. The drop ratio of the malicious node does not have a big effect on the quality of detection. WatchAnt detects misbehavior when the report contained in the WatchAnt Reply is not correct. It is not important how many packets were dropped by the malicious node, the hashsum of the packets reported to the requesting node is incorrect as long as one packet has not been forwarded.

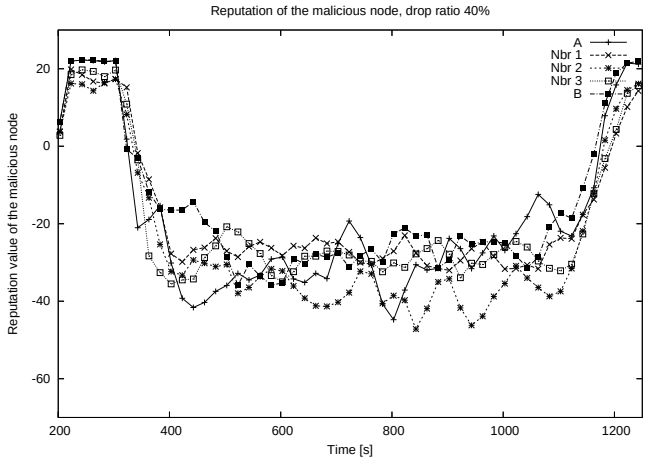
In Subsection 8.6.1 we stated that the detection quality is mainly influenced by the WatchAnt parameter *WA_Number_of_hashes_to_ask_for*. With this we can define the number of packets to which a neighboring node has to provide evidence of having forwarded these packets. With this parameter we can influence the probability to detect a malicious neighboring node as described in Subsection 8.6.1: Misbehavior of malicious nodes with a drop ratio of P_{drop} is detected with a probability of $1 - (1 - P_{drop})^{WA_Number_of_hashes_to_ask_for}$.



(a) Reputation of the malicious node with drop ratio 0.0. Interval of misbehavior: 300s to 1100s

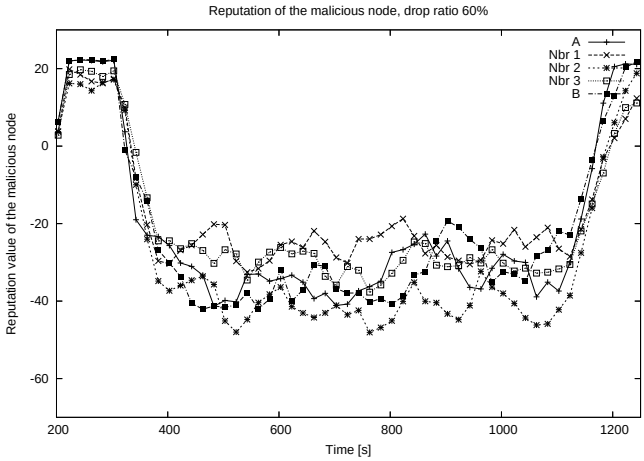


(b) Reputation of the malicious node with drop ratio 0.2. Interval of misbehavior: 300s to 1100s

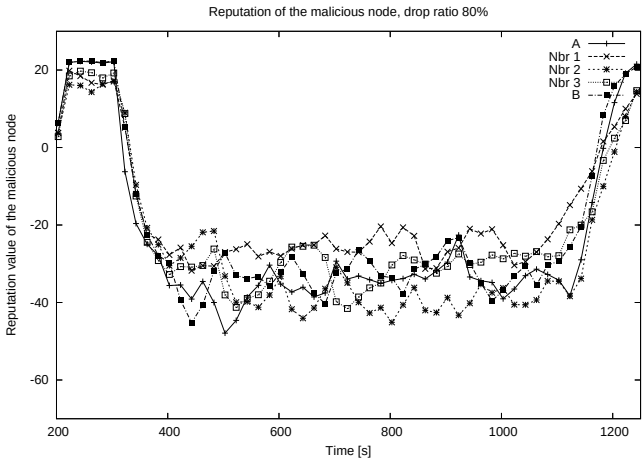


(c) Reputation of the malicious node with drop ratio 0.4. Interval of misbehavior: 300s to 1100s

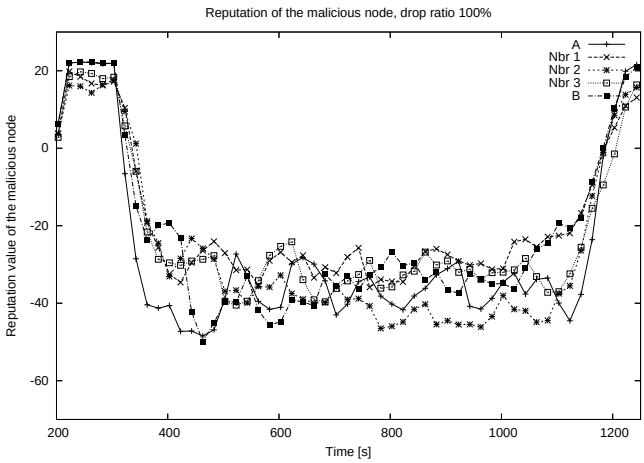
Figure 8.30: Reputation of malicious node in Scenario B, with varying drop ratio (A)



(a) Reputation of the malicious node with drop ratio 0.6. Interval of misbehavior: 300s to 1100s



(b) Reputation of the malicious node with drop ratio 0.8. Interval of misbehavior: 300s to 1100s



(c) Reputation of the malicious node with drop ratio 1.0. Interval of misbehavior: 300s to 1100s

Figure 8.31: Reputation of malicious node in Scenario B, with varying drop ratio (B)

8.7 Results: Scenario C

In Figure 8.32 we see the effect of colluding misbehaving nodes: the malicious nodes decrease the throughput in the network from 97,5% to 34,75% when colluding. By comparing the second and third bar we can see how effectively AntSec detects misbehavior and increases throughput. In both cases the malicious nodes are active and not colluding. Throughput rises from 7,49% when AntSec is disabled to 97,5% when AntSec is enabled. For the results of the fourth bar the malicious nodes were colluding, i.e. dropping data packets of benign nodes only when received from a malicious node. Throughput decreases from 97,5%, when AntSec is enabled and malicious nodes are not colluding, to 34,75% in case that AntSec is enabled and malicious nodes are colluding.

This shows that collusion is an effective strategy for malicious nodes to avoid detection by WatchAnt. Or more generally it is an effective strategy to avoid detection by mechanisms which verify correct forwarding behavior of neighboring nodes. Still WatchAnt fulfills its tasks: it detects misbehavior. But as only malicious nodes detect the misbehavior of each other, these observations lead to no penalty.

With this scenario we also want to demonstrate the quality of our Leak Detector mechanism. Based on the traffic information transmitted in the Maintenance Forward Ants the destination node of a route builds a virtual graph modeling the route. The destination node calculates for each node, which participates in the route, the inflow and outflow. Based on the inflow and outflow of each node we implemented rules to decide whether a node is to be considered malicious or not.

We implemented the following rules:

A node in the route is not considered malicious if:

- The node is source or destination of the route
- The difference of the inflow and the outflow of the node is smaller than 5% of the total traffic

If a node does not fit in the previous two categories, the node is considered malicious if:

- Inflow of the node is 10 time bigger than the outflow: $in_{node} > 10 \cdot out_{node}$
- Inflow of the node is bigger than 5% of total traffic and $in_{node} > constA \cdot out_{node}$. Whereas *constA* is a parameter we varied in our simulations.

If none of the categories mentioned above suits for a node, the node it is considered benign.

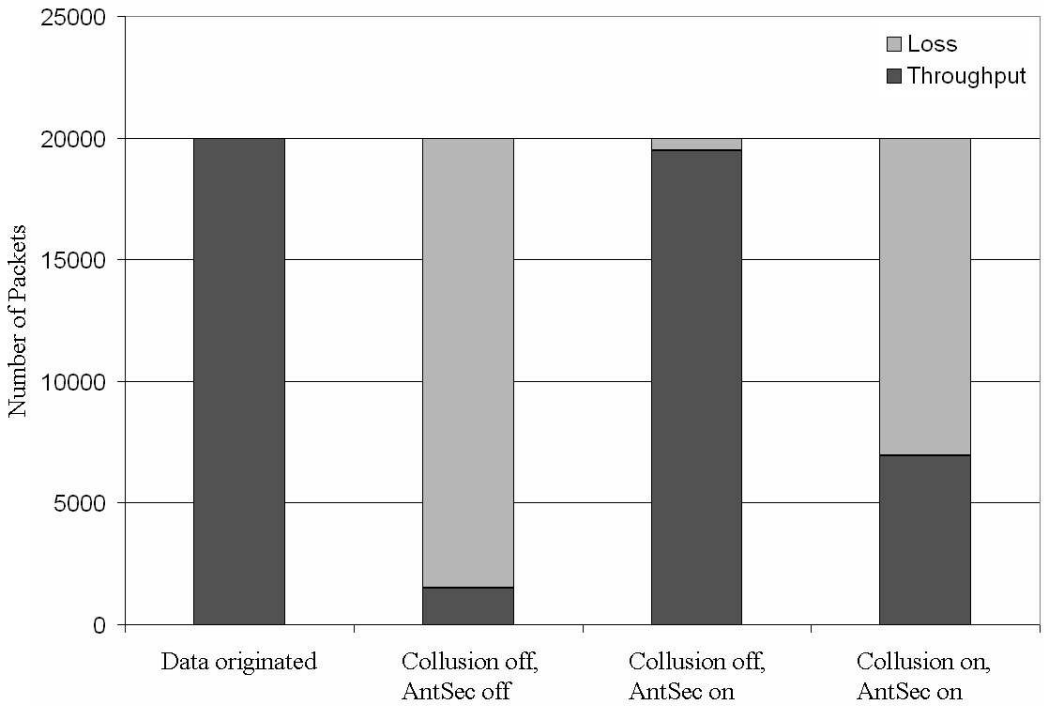


Figure 8.32: The effects of colluding malicious nodes are shown: The first bar shows the amount of originated data packets. The second, third and forth bar show the numbers of the lost and received data packets. The second bar shows the results for the case that the malicious nodes are active and AntSec is disabled. The third bar shows the results for the malicious nodes being active but not colluding and AntSec is enabled. The fourth bar shows the results for the malicious nodes colluding and AntSec being enabled.

ConstA	valid detections	standard deviation		invalid detections	standard deviation
2	3071,6	310,7		551,4	473,4
3	3204,4	357,8		34,4	153,8

Table 8.8: Number of valid and invalid detections of malicious colluding nodes on the route.

The Leak Detector is a passive element in this Scenario. It does not give any feedback to the routing system. In case of C.4 the Leak Detector is enabled. It analyzes the traffic and decides according to the aforementioned rules, whether a node is to be suspected of being malicious or not. In Table 8.8 we discuss the results for choosing 2 and 3 for the parameter *constA* of the Leak Detector. For choosing *constA* = 2 we have 3071 valid detectionsin 1000 seconds simulation time, i.e. a malicious node has been suspected of misbehaving 3071 times in cases upon receipt of a FANT. But there have also been 551

invalid detections, i.e. a benign node has been suspected for being malicious.

The results for choosing *constA* = 3 are significantly better. There have been 3204 valid detections and only 34 invalid detections in average. Only very few benign nodes have been suspected falsely as malicious, which shows two things: First the Leak Detector provides very good detection quality. Its precise observations can be used to improve routing. Second, the strategy used to detect misbehavior is not mature and needs further optimization. But still, using the current strategy provides are very good results.

Chapter 9

Conclusions and Future Work

Abstract

In this chapter we give summary of our work and contribution. We also present a brief overview of the questions and tasks that remained open.

In our work we have developed a security framework for Wireless Mesh Networks. We first have analyzed the characteristics of IEEE 802.16 Mesh Mode [ICS04]. Hereby the specifications and requirements defined by the MAC layer were emphasized in our investigations. These characteristics are relevant for any routing algorithm operating on IEEE 802.16 Mesh Mode. The main aspects of the MAC layer, which have influence on the design of a secure routing algorithm, are:

- IEEE 802.16 Mesh Mode does not provide broadcasting functionality for data messages.
- Links between two neighboring nodes are encrypted, each node-node combination maintains its own keys.
- Collisions may occur, due to the optimized utilization of the wireless medium. But collisions may only occur at nodes to which the messages were not destined. IEEE 802.16 guarantees only unidirectional unicast links.

It is important to derive the requirements defined by the IEEE 802.16 Mesh Mode on the routing algorithm, as these requirements have to be considered in general by every secure routing algorithm for the IEEE 802.16 Mesh Mode.

In the next step we looked at the terminology used to describe security related topics. Based on the investigations of the IEEE 802.16 MAC layer and security we looked at the categories of attackers and derived a detailed listing of attacks, that can be applied in Wireless Mesh Networks.

As Mobile Ad Hoc Networks are in many aspects equal to Wireless Mesh Networks, we investigated routing protocols in Mobile Ad Hoc Networks. We then presented mechanisms, discussed in the context of Mobile Ad Hoc Networks, to secure routing:

- Watchdog mechanism variants to detect misbehavior in Mobile Ad Hoc Networks. Up to now the Watchdog mechanism was the only way to locally detect misbehavior of neighboring nodes in forwarding data messages without using cryptography. But Watchdog does not cope with encrypted links and collisions.
- Cryptographic mechanisms which are essential to provide integrity and authenticity. These mechanisms are used to make specific misbehavior impossible. Attacks on integrity and authenticity are detected immediately.
- Incentive based concepts to motivate nodes participating in the network to forward data packets for other nodes. These have strict requirements, which are hard to fulfill.

Based on the investigation of related work, we have developed a security framework for routing in Wireless Mesh Networks, optimized for IEEE 802.16 Mesh Mode.

To accomplish authenticity and integrity for our routing algorithm we first developed a mechanism for secure network entry. During the secure network entry Registration Certificates are given to new nodes, which are authorized to participate in the network. Based on these Registration Certificates any node in the network can be authenticated and non-repudiation is provided. This is to our best knowledge one of the first solutions for bootstrapping security in the network entry of IEEE 802.16 Mesh Mode. Solutions to provide security for the MAC layer of IEEE 802.16 Mesh Mode are only discussed in [Sar06] in literature.

AntSec is a proactive, probabilistic, multipath, stigmergy based, distributed, no-broadcasting, secure routing algorithm. The basis routing functionality has been implemented and evaluated by Andreas Ziller (see [Zil06]). Based on this insecure routing algorithm, which provides only core routing functionality, we designed AntSec.

AntSec is one of the first contributions for securing stigmergy based routing algorithms. Stigmergy based routing algorithms as discussed in 4.3 are a quite young research area. Additionally AntSec is one of the few secure routing algorithms which does not require broadcasting of routing messages. AntSec guarantees that only authenticated nodes participate in the network. With this misbehaving nodes can be identified and excluded from the network. With AntSec forging and modification of routing messages have no effect, routing messages which have effect on routing guarantee integrity and are authenticated.

The main security issue in routing is that malicious or selfish nodes in the network do not forward messages. To detect this misbehavior we have developed WatchAnt. In contrast to the commonly used Watchdog mechanism, our solution copes with encrypted links in MAC layer. Besides our solution only Watchdog introduced by Marti et al. [MGLB00] and derivates from it, are discussed in literature. WatchAnt is the first mechanism to detect forwarding misbehavior in networks with encrypted links without using cryptography. Furthermore WatchAnt detects misbehavior of malicious nodes quite independent of their drop ratio, as it is shown in Subsection 8.6.2.

Observations made with WatchAnt are passed to the Reputation Management System, which rates the observations and modifies the appropriate reputations. Based on these reputations, strategies are determined how to handle neighboring nodes. We have implemented a system to pass reputation values efficiently and compliant to IEEE 802.16 Mesh Mode to neighboring nodes. By this nodes can cooperate to improve detection quality.

In addition to a secure routing algorithm and mechanism to detect misbehavior, we developed a mechanism to detect colluding malicious nodes. Colluding malicious nodes are

a severe risk to routing due to the fact that misbehavior detection mechanisms discussed in literature and also WatchAnt detect only misbehavior of direct neighbors. By colluding, malicious nodes seem benign to other nodes, misbehaving is revealed only to each other. But as they are colluding, misbehavior is not punished. In literature this problem is described, but only few solutions exist (e.g. iterative probing described in Subsection 5.2.6). The Leak Detector we introduced in Section 6.5 is coping with this problem and presenting good detection results, as the simulation results in Section 8.7 show. The Leak Detector is one of the first mechanism solving the problem of malicious colluding nodes in Wireless Mesh Networks.

With the simulations we have shown that AntSec provides significantly higher throughput than Ant and Dynamic Source Routing [JM96]. In Scenario B we have shown that the detection capabilities of WatchAnt do not depend on the drop ratio of the malicious nodes (at least to a specific value). Finally we have shown in Scenario C that colluding misbehaving nodes can significantly decrease throughput, even in the presence of misbehavior detection mechanisms. But the Leak Detector shows good results in detecting malicious colluding nodes on routes.

The overview of the work done leads us to the work that remained open and could be done in future.

Looking at AntSec there are two components that can be optimized: the routing algorithm itself and the security components. As shown in Table 8.1, AntSec has various parameters, especially the intervals in which Discovery Forward Ants and Maintenance Forward Ants are sent can be optimized. These two parameters have the greatest effect on the traffic overhead measured in Figure 8.5(b) and 8.6(b). With respect to the security components of AntSec packet sizes are already optimized. But still, all parameters listed in Table 8.1 are based on empirical observations and need evaluation and optimization.

The detection quality of WatchAnt depends on the parameters listed in 8.1. The strategy by which neighboring nodes are chosen to send WatchAnt requests to them, is choosing a neighboring node randomly. Strategies have to be researched in which intervals to ask which neighbor.

The Reputation Management System maintains the reputation values and shares them with neighboring nodes. Both tasks can be optimized. The modifications on the reputations in reply to observations are based on empirical results, but need further analysis. The second task is to share reputation values with neighboring nodes. This is done already, but second hand reputation values are not taken into account for routing. Various strategies, how to use second hand reputation values, can be determined and have to be analyzed.

We have implemented the Leak Detection mechanism. Based on the results seen in Section 8.7 our Leak Detector provides very good detection quality. Still the optimal rules

to determine colluding malicious nodes on routes needs to be determined. Additionally detection results provided by the Leak Detector have to influence routing. Strategies how these results effect AntSec has to be developed and evaluated.

Furthermore the WatchAnt, Reputation Management System and Leak Detector can be applied to other routing algorithms.

Nomenclature

ABC Ant Based Control

AES Advanced Encryption Standard

AK Authorization Key

AN Authorization Node

AntHocNet Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks

AntNet Adaptive Agent-based Routing

AODV Ad Hoc On-Demand Distance-Vector Routing

ARA Ant-Colony-Based Routing Algorithm

ARAN Authenticated Routing for Ad Hoc Networks

Ariadne A Secure On-Demand Routing Protocol for Ad Hoc Networks

BANT Backward Ant

BS Base Station

CBR Constant Bit Rate

CID Connection Identifier

CN Candidate Node

CONFIDANT Cooperation Of Nodes - Fairness in Dynamic Ad-hoc Networks

CORE Collaborative Reputation Mechanism

DES Data Encryption Standard

DFANT Discovery Forward Ants

DH Diffie-Hellman

DLP Discrete Logarithm Problem

DSDV Destination-Sequenced Distance-Vector Routing Protocol

DSR Dynamic Source Routing

DV Distance Vector

FDD Frequency Division Duplex

GSM Global System for Mobile Communications

HMAC Keyed-Hash Message Authentication Code

IEEE Std 802.16-2004 IEEE 802.16 Standard in the revision of the year 2004

IP Internet Protocol

KEK Key Encryption Key

LOS Line of sight

LS Link State

MAC Message Authentication Code

MAN Metropolitan Area Network

MANET Mobile Ad Hoc Network

MANETID Mobile Ad-Hoc Network Node Identifier

MFANT Maintenance Forward Ant

mobIDS mobile intrusion detection system

MSH Mesh Mode

MSH-DSCH Mesh Distributed Scheduler (control message)

Nonce Number used once

OCEAN Observation-based Cooperation Enforcement in Ad hoc Networks

PANT Proactive Forward Ant

PERA Probabilistic Emergent Routing Algorithm

PFS Perfect Forward Secrecy

PKI Public Key Infrastructure

PKM Privacy Key Management

PKM Public Key Management

PMP Point-to-MultiPoint Mode

RANT Reactive Forward Ant

RERR Route Error Message

RREP Route Reply Message

RREQ Route Request Message

RSA A common asymmetric cryptography mechanism named after its inventors Ronald L. Rivest, Adi Shamir and Leonard Adleman

SA Security Association

SAM Security Architecture for Mobile Ad Hoc Networks

SAODV Secure Ad-Hoc On-Demand Distance Vector Routing

SDSR Secure Dynamic Source Routing

SEAD Secure Efficient Ad Hoc Distance Vector Routing Protocol

SRP Secure Routing Protocol

SS Subscriber Station

TDD Time Division Duplex

TEK Traffic Encryption Key

TESLA Timed Efficient Stream Loss-tolerant Authentication

TIK TESLA with Instant Key Disclosure

TTL Time To Live

WMANIFMIB Wireless MAN Interface Management Information Base

WMN Wireless Mesh Network

Bibliography

[Abe01] Karl Aberer. P-Grid: a Self-Organizing Access Structure for P2P Information Systems. In *COOPLS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*, pages 179–194, London, UK, 2001. Springer-Verlag.

[Abr99] Henrik Abrahamsson. Traffic Measurement and Analysis. 1100 3154, SICS, 1999.

[AGK+01] Payman Arabshahi, Andrew Gray, I. Kassabalidis, M.A. El-Sharkawi, R.J. Marks II, A. Das, and S. Narayanan. Adaptive Routing in Wireless Communication Networks using Swarm Intelligence. In *9th AIAA Int. Communications Satellite Systems Conference, Toulouse, France*, 17 April 2001.

[AHNRR02] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *ACM Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, September 2002.

[AK96] R. Anderson and M. Kuhn. Tamper Resistance - A Cautionary Note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[AK97] Ross Anderson and Markus Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *IWSP: International Workshop on Security Protocols, LNCS*, 1997.

[Aka02] Idris Akar. Modeling and Simulation of Mobile Packet Switched Networks. Master's thesis, Technical University of Darmstadt, February 2002.

[Ame81] American National Standards Institute. American National Standard for Data Encryption Algorithm (DEA). Technical report, 1981.

[Bar01] Benjamin Baran. Improved AntNet Routing. *SIGCOMM Comput. Commun. Rev.*, 31(2 supplement):42–48, 2001.

[BB02a] S. Buchegger and J. Le Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In IEEE Computer Society, editor, *Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, January 2002.

[BB02b] Sonja Buchegger and Jean-Yves Le Boudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, January 2002. IEEE Computer Society.

[BB02c] Sonja Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation of Nodes - Fairness in Dynamic Ad Hoc Networks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, CH, June 2002. IEEE.

[BB03a] S. Bansal and M. Baker. Observation-based Cooperation Enforcement in Ad Hoc Networks. Technical report, Stanford University, 2003.

[BB03b] S. Buchegger and J. Le Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad Hoc Networks. In Proc. WiOpt'03 (Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks), 2003., 2003.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Keyed Hash Functions and Message Authentication. In *Proceedings of Crypto'96, LNCS 1109*, pages 1–15, 1996.

[BDT99] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[BH00] Levente Buttyan and Jean-Pierre Hubaux. Enforcing Service Availability in Mobile Ad Hoc WANs. In *Proceedings of the First IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Boston, MA, USA, 2000.

[BH02] Levente Buttyan and Jean-Pierre Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Selforganized Ad Hoc Networks. Technical Report DSC/2001, Swiss Federal Institute of Technology – Lausanne, Department of Communication Systems, March 2002.

[BHvR05a] Rimon Barr, Zygmunt Haas, and Robbert van Renesse. *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad hoc Wireless, and Peer-to-Peer Networks*, chapter 19, pages 297–311. CRC Press, 2005. Scalable Wireless Ad hoc Network Simulation.

[BHvR05b] Rimon Barr, Zygmunt Haas, and Robbert van Renesse. JiST: an Efficient Approach to Simulation using Virtual Machines: Research Articles. *Softw. Pract. Exper.*, 35(6):539–576, 2005.

[BL73] D.E. Bell and L. J. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, The MITRE Corporation, Bedford MA, May 1973.

[Bli06] Boris Bliwier. Medium Access Control Layer Quality of Service Mechanisms for IEEE 802.16 Mesh Mode. Master's thesis, Technical University of Darmstadt, Germany, 2006.

[BM03] J. Baras and H. Mehta. A Probabilistic Emergent Routing Algorithm for Mobile Ad hoc Networks. In , March 3-5, 2003. *WiOpt'03: Modeling and Optimization in Mobile Ad Hoc and Wireless Networks*, 3 March 2003.

[BS01] B. Baran and R. Sosa. AntNet: Routing Algorithm for Data Networks based on Mobile Agents. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artifical. No.12*, (12):75–84, 2001.

[BSNP95] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Cryptographic Hash Functions: a Survey. Technical Report 95-09, Department of Computer Science, Universit of Wollongong, July 1995.

[Buc04] Sonja Buchegger. *Coping With Misbehavior in Mobile Ad-hoc Networks*. PhD thesis, Swiss Federal Institute of Technology (EPFL), April 2004.

[CMZ+05] Min Cao, Wenchao Ma, Qian Zhang, Xiaodong Wang, and Wenwu Zhu. Modelling and Performance Analysis of the Distributed Scheduler in IEEE 802.16 Mesh Mode. In *MobiHoc*, pages 78–89, 2005.

[DCD97] Gianni Di Caro and Marco Dorigo. Antnet: a Mobile Agents Approach to Adaptive Routing. Technical Report 12, Universite Libre de Bruxelles, IRIDIA, 1997.

[DCD98a] Gianni Di Caro and Marco Dorigo. Ant Colonies for Adaptive Routing in Packet-Switched Communications Networks. In *Lecture Notes in Computer Science*, volume 1498, pages 673–??, 1998.

[DCD98b] Gianni Di Caro and Marco Dorigo. Antnet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[DD99] Marco Dorigo and Gianni Di Caro. *The Ant Colony Optimization Meta-Heuristic*, pages 11–32. McGraw-Hill, London, 1999.

[DDCG98] M. Dorigo, G. Di Caro, and L. Gambardella. Ant Algorithms for Discrete Optimization. IRIDIA 98-10, Universite Libre de Bruxelles, Belgium, 1998.

[DDG04] Gianni Di Caro, Frederick Ducatelle, and Luca Maria Gambardella. *AntHocNet: an Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks*. Number 3242 in Lecture Notes in Computer Science. Springer-Verlag, Birmingham, UK, September 2004. Best paper award.

[DH76] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - the Advanced Encryption Standard*. Springer-Verlag, Berlin Germany/ Heidelberg, Germany / London, UK / etc., 2002.

[Eck04] Claudia Eckert. *IT-Sicherheit - Konzepte - Verfahren - Protokolle, 3. Auflage*. Oldenbourg-Verlag, 2004.

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In *Proceedings of Crypto 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[FN01] Laura Marie Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *INFOCOM*, pages 1548–1557, 2001.

[Gra59] P. P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. La théorie de la stigmerie: essai d'interprétation du comportment des termites constucteurs. *Insects Sociaux, 6, 41-81*, 1959.

[GS03] Mesut Günes and Otto Spaniol. Routing Algorithms for Mobile Multi-hop Ad Hoc Networks. In *Net-Con 2003*, pages 120–138, 2003.

[GZ02] Manel Guerrero Zapata. Secure Ad hoc On-Demand Distance Vector Routing. *ACM Mobile Computing and Communications Review (MC2R)*, (3):106–107, 6 July 2002.

[H. 02] H. Mehta. Dynamic Adaptive Routing in Mobile Ad Hoc Networks. Master's thesis, ECE Department, University of Maryland, December 2002.

[HJP02] Y. Hu, D. Johnson, and A. Perrig. SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks. In *4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002), IEEE, Calicoon, NY*, June 2002.

[HOW94] Bert Hoelldobler and Edward O. Wilson. *Journey to the Ants : a Story of Scientific Exploration*. Cambridge, Mass.: Belknap Press of Harvard Univ. Press, c1994, 3 edition, 1994.

[HPJ01] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. Technical report, Department of Computer Science, Rice University, December 2001.

[HPJ02a] Y. Hu, A. Perrig, and D. Johnson. Ariadne: a Secure On-Demand Routing Protocol for Ad Hoc Networks. In *8th ACM International Conference on Mobile Computing and Networking*, September 2002.

[HPJ02b] Y. Hu, A. Perrig, and D. Johnson. Wormhole Detection in Wireless Ad Hoc Networks. Technical report, Rice University, Department of Computer Science, June 2002.

[ICS04] IEEE Microwave Theory and Techniques Society IEEE Computer Society. 802.16 Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems, 1 October 2004.

[ICS05a] IEEE Microwave Theory and Techniques Society IEEE Computer Society. 802.16 Standard for Local and metropolitan area networks, Part 16: Air

Interface for Fixed Broadband Wireless Access Systems, Amendment 1-Management Information Base, 2005.

[ICS05b] IEEE Microwave Theory and Techniques Society IEEE Computer Society. 802.16 Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems. Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands, 2005.

[IETF81] Network Working Group Internet Engineering Task Force. RFC 791 Internet Protocol. http://www.ietf.org/rfc/rfc791.txt, 1981.

[IETF83] Network Working Group Internet Engineering Task Force. RFC 868 Time Protocol. http://www.ietf.org/rfc/rfc868.txt, 1983.

[IETF97] Network Working Group Internet Engineering Task Force. RFC 2104 HMAC: Keyed-Hashing for Message Authentication. http://www.ietf.org/rfc/rfc2104.txt, February 1997.

[IETF98] Network Working Group Internet Engineering Task Force. RFC 2460 Internet Protocol, Version 6 (IPv6) Specification. http://www.ietf.org/rfc/rfc2460.txt, December 1998.

[IETF00] Network Working Group Internet Engineering Task Force. RFC 2828 Internet Security Glossary. http://www.ietf.org/rfc/rfc2828.txt, May 2000.

[IETF03] Network Working Group Internet Engineering Task Force. RFC 3561 Ad Hoc On-Demand Distance Vector (AODV) Routing. http://www.ietf.org/rfc/rfc3561.txt, 2003.

[Ink04] Kai Inkinen. New secure Routing in Ad Hoc Networks: Study and Evaluation of Proposed Schemes. Helsinki University of Technology, Laboratory of Multimedia etc, 24 April 2004.

[JK05] Paul Barom Jeon and George Kesidis. Pheromone-aided Robust Multipath and Multipriority Routing in Wireless MANETs. In *PE-WASUN '05: Proceedings of the 2nd ASM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, pages 106–113, New York, NY, USA, 2005. ACM Press.

[JM96] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. in Mobile Computing (ed. T. Imielinski and H. Korth), Kluwer Academic Publishers, Dordrecht, The Netherlands., 1996.

[JW04] David Johnston and Jesse Walker. Overview of IEEE 802.16 security. *IEEE Security and Privacy*, 2(3):40–48, 2004.

[Kar03] Frank Kargl. *Sicherheit in Mobilen Ad-Hoc Netzwerken*. PhD thesis, University of Ulm, Germany, September 2003.

[KG05] Frank Kargl and Andreas Geiss. Secure Dynamic Source Routing. *Hawaiian International Conference on System Sciences 38, Hawaii, USA*, January 2005.

[KK04] Frank Kargl and Andreas Klenk. Sensors for Detection of Misbehaving Nodes in MANETs, Detection of Intrusion and Malware & Vulnerability assessment (dimva 2004). Lecture Notes in Informatics, Vol. 46, Dortmund, July 2004.

[KS04] Frank Kargl and Stefan Schlott. Securing Ad Hoc Routing Protocols. 30th Euromicro Conference, Rennes, September 2004.

[KSK+04] Frank Kargl, Stefan Schlott, Andreas Klenk, Alfred Geiss, and Michael Weber. Advanced Detection of Selfish or Malicious Nodes in Ad Hoc Networks. Proceedings of 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004), Springer Lecture Notes in Computer Science, Heidelberg, September 2004.

[LX01] Wenke Lee and Dong Xiang. Information-Theoretic Measures for Anomaly Detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 130–143, May 2001.

[Mer78] Ralph C. Merkle. Secure Communications over Insecure Channels. *Communications of the. ACM*, 21(4):294–299, 1978.

[MGLB00] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Sixth Annual International Conference on Mobile Computing and Networking*, pages 255–265, 2000.

[MM01] P. Michiardi and R. Molva. Core: A COllaborative REputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks. Technical Report RR-02-062, EurecomResearch, December 2001.

[MM02a] P. Michiardi and R. Molva. Game Theoretic Analysis of Security in Mobile Ad Hoc Networks. Technical Report RR-02-070, Eurecom Research, April 2002.

[MM02b] P. Michiardi and R. Molva. Prevention of Denial of Service Attacks and Selfishness in Mobile Ad Hoc Networks. Technical Report RR-02-063, Eurecom Research, January 2002.

[NS87] R M Needham and M D Schroeder. Authentication Revisited. *SIGOPS Oper. Syst. Rev.*, 21(1):7–7, 1987.

[PB94] Charles Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[PCST01] Adrian Perrig, Ran Canetti, Dawn Song, and Doug Tygar. Efficient and Secure Source Authentication for Multicast. In *Network and Distributed System Security Symposium, NDSS '01*, pages 35–46, February 2001.

[PCTS02] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. In *Cryptobytes*, volume 5, pages 2–13. RSA Laboratories, 2002.

[Per88] Radia Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Massachussetts Institute of Technology, 1988.

[Per97] C. Perkins. Ad-Hoc On-Demand Distance Vector Routing. *MILCOM '97 Panel on Ad Hoc Networks*, November 1997.

[PH02] P. Papadimitratos and Z. Haas. Secure Routing for Mobile Ad Hoc Networks. *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference, San Antonio, TX,*, 27 January 2002.

[Rab79] M.O. Rabin. Digitalized Signatures as Intractable as Factorization. Technical Report MIT/LCS/TR-212, MIT, 1979.

[RSA77] R. L. Rivest, A. Shamir, and L. M. Adelman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Technical Report MIT/LCS/TM-82, 1977.

[RW03] Martin Roth and Stephen Wicker. Termite: Emergent Ad-Hoc Networking. In *Second Mediterranean Workshop on Ad-Hoc Networks*, 2003.

[Sar06] Damir Sarac. Security Mechanisms for IEEE 802.16 based Mesh Networks. Master's thesis, Technical University of Darmstadt, 2006.

[SBHR96] Ruud Schoonderwoerd, Janet L. Bruten, Owen E. Holland, and Leon J. M. Rothkrantz. Ant-Based Load Balancing in Telecommunications Networks. *Adapt. Behav.*, 5(2):169–207, 1996.

[SDLBR02] K. Sanzgiri, B. Dahill, B. Levine, and E. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. *International Conference on Network Protocols (ICNP), Paris, France*, November 2002.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sho99] P. W. Shor. Polynomial-time Algorithm for Prime Factorization and Discrete Logarithms on a Quantum Computer. In *SIAM Review*, volume 41, pages 303–332, 1999.

[Sim00] Simon Singh. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. New York NY: Anchor Books, 2000.

[TMS02] C. Tham, S. Marwaha, and D. Srinivasan. Mobile Agents Based Routing Protocol for Mobile Ad Hoc Networks. In *Global Telecommunications Conference - GLOBECOM'02, Taipei, Taiwan*, 2002.

[ZE02] W. Zhong and D. Evans. When Ants attack: Security Issues for Stigmergic Systems. Technical Report CS-2002-23, University of Virgina, Department of Computer Science, April 2002.

[Zil06] Andreas Ziller. Dienstgüteorientiertes Routing in IEEE 802.16 Mesh Netzen mit Cross-Layer-Optimierung. Master's thesis, Technical University of Darmstadt, 2006.

[ZL00] Yongguang Zhang and Wenke Lee. Intrusion Detection in Wireless Ad-Hoc Networks. In *ACM MOBICOM'2000*, pages 275–283, 2000.

[ZLH02] Y. Zhang, W. Lee, and Y. Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 2002.

[ZYC02] S. Zhong, Y. Yang, and J. Chen. Sprite: A Simple, Cheat-proof, Credit-based System for Mobile Ad Hoc Networks, 2002.