A Distributed Policy Enforcement Architecture for Mobile Ad-Hoc Networks

Mansoor Alicherry

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2011

ABSTRACT

A Distributed Policy Enforcement Architecture for Mobile Ad-Hoc Networks

Mansoor Alicherry

Mobile Ad-hoc Networks (MANETs) are increasingly employed in tactical military and civil rapid-deployment networks, including emergency rescue operations and *ad hoc* disaster-relief networks. However, this flexibility of MANETs comes at a price when MANETs are compared to wired and base station-based wireless networks: MANETs are susceptible to both insider and outsider attacks. This is mainly because of the lack of a well-defined defense perimeter preventing the effective use of wired defenses, including firewalls and intrusion detection systems.

In this thesis, we present DIPLOMA, a novel distributed security policy enforcement architecture that is designed specifically for MANETs. Our approach harnesses and extends the concept of *network capabilities* and is especially suited for mobile and heterogeneous communication environments. Our model imposes communication restrictions between MANET nodes by enforcing hop-by-hop policies in a distributed manner. We use a *deny-by-default* principle, allowing compromised nodes to access only authorized services. This significantly limits their ability to disrupt or even interfere with end-to-end connectivity and nodes beyond their local communication radius.

In this thesis, we present the feasibility of the architecture using a multitude of techniques. We define a threat model and a security analysis based on that threat model. We conduct a preliminary evaluation of the system using the GloMoSim simulator. Through simulations, we show that our solution incurs minimal overhead in terms of network bandwidth and latency even in the presence of cryptographic operations. Furthermore, we show that the protection remains effective even in the presence of misbehaving nodes and routing changes due to mobility.

We also present an implementation of DIPLOMA on real systems running the Linux operating system. Our implementation works at the network layer and does not require any changes to existing applications. We identify the bottlenecks and make improvements to the system so that it works well in practice. We evaluate our system in a realistic MANET testbed Orbit. To that end, we identify ways of creating multi-hop topologies in indoor environments so that a bad node cannot interfere with every other node. Our evaluations show that the system incurs minimal overhead and protects network bandwidth and the end-hosts in the presence of attackers.

We also extend the DIPLOMA architecture to secure multicast traffic. We use capabilities to provide a unified solution for sender and receiver access control to multicast groups, as well as to limit the bandwidth usage of the multicast groups. We have extended common multicast protocols, including ODMRP and PIM-SM, to incorporate DIPLOMA. We have implemented multicast DIPLOMA in Linux without requiring any changes to existing applications and the routing substrate. We evaluate the system in the Orbit Lab testbed as well as the GloMoSim simulator to show that the system incurs minimal overhead and protects multicast traffic in the presence of attackers.

In this thesis, we also identify how senders and receivers can misuse capabilities by using them in multiple paths, and we provide distributed solutions for detecting those misuses. To that end, we modify the capabilities to aid in misuse detection and provide protocols for exchanging information for distributed detection. We also provide efficient algorithms for misuse detection and protocols for providing proof of misuse. Our solutions can handle privacy issues associated with the exchange of information for misuse detection. We have implemented misuse detection and recovery in DIPLOMA systems running on Linux and have conducted extensive experimental evaluation of the system in the Orbit MANET testbed. The results show that our algorithms require minimal processing and memory and that distributed detection requires a very small amount of bandwidth. We also show that our system is effective in detecting and containing multi-path misuses.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to offer my sincerest gratitude to my advisor Angelos Keromytis for the guidance, encouragement and support he provided for the past four years. I appreciate all his contribution of time and ideas that made this thesis possible. I am also grateful for the freedom he has given me to work on my own schedule that made my journey much easier.

I would like to thank my thesis committee members Sal Stofo, Steve Bellovin, Angelos Stavrou and Scott Alexander for their time and comments. Insightful comments provided by Scott helped me improve the thesis considerably. Angelos Stavrou also contributed to the initial discussion of the architecture and co-authored the architecture papers.

I am grateful to my employer, Alcatel-Lucent Bell labs, for supporting my PhD. I was fortunate to have managers who always encouraged me to do a PhD. I want to thank Vijay Kumar, Vishy Poosala, Pramod Koppol, Dimitri Stiliadis, Dor Skuler, and Ram Ayyakad for the support they have given me during the course of my PhD. I am also thankful to Alcatel-Lucent colleagues who encouraged me, including Randeep, Hari, Furquan, Ken, Doug, Goutham, Satish and all the NLG team members.

I want acknowledge WINLAB for providing the testbed for running the experiments. I want to thank Ivan Sesker for resolving all the issues related to the testbed even on night and weekends.

I am thankful to my NSL lab mates Sambuddho, Kang-Kook, and Vasilis for resolving issues related to lab machines and handling many paper work required that helped me avoid many commutes to the university campus. I also thank Evelyn, Lily and other administrative and IT staff of computer science department that made my journey easier.

Doing a part-time PhD requires lot of sacrifices from the family. I cannot thank enough my wife Hafsath and daughter Sheza for the sacrifices they made, and care, love and support they gave me to during the course of my PhD. I also thank our family friends Krishnan,

Renu, and little Dev for the company they gave us during this course. Finally, I thank my parents, in-laws, and siblings for all the encouragements they provided.

Chapter 1

Introduction

Recent advances in low-power computing and communications have led to the proliferation of hand-held and portable devices equipped with wireless connectivity [iph10; nex10; dro10]. These mobile wireless devices appear to be ideal for situations where fixed infrastructure is too costly or dangerous to deploy or has been rendered inoperable. However, because of radio power consumption, physical obstacles, and channel capacity, a mobile node may not be able to reach all other nodes within a single broadcast. Therefore, to achieve end-to-end connectivity, nodes have to form mobile *ad hoc* wireless networks (MANETs), which allow data to be routed through intermediate nodes. MANETs are fundamentally different from the Internet because all peers act as both sources and routers using the other participants to relay packets to their final destination. Due to their flexibility, MANETs are currently employed in both military and commercial applications.

Unfortunately, not all MANET nodes are equally capable, nor can all users be equally trusted. Worse yet, mobile nodes in tactical environments run the danger of being captured or malfunctioning. Even a small number of misbehaving nodes can successfully render the entire MANET inoperable: malicious peers can abuse the network, exhausting all network and power resources.

In traditional networks, malicious nodes and traffic are kept away from a set of nodes belonging to an organization or a group using *firewalls* [CB94; CZ95]. Traditional firewalls rely on the existence of a well-defined network topology and a perimeter. Every node on one side of the perimeter ("inside") can be trusted, and the nodes in the other side ("outside")

are potential enemies. Specialized nodes called firewalls are placed at the perimeter nodes. All incoming and outgoing traffic needs to transit through these firewall nodes, which enforce the *policies* at the perimeter. The administrators set the policies for controlling external network access. Within the perimeter, smaller sub-groups can have policies that are more stringent by deploying their own firewalls.

Unfortunately, the concept of a network perimeter does not exist in MANETs. The mobile nodes can join or leave a MANET at any time. The mobility of the nodes makes the topology of a MANET change in an unpredictable manner. As there is no concept of "inside" and "outside" in a MANET, traditional firewalls cannot be deployed. None of the nodes in a MANET can be trusted; hence, access control needs to be enforced for all traffic.

Traditional firewalls also have several shortcomings [Bel99; Li00; IKBS00; Ste01]. As firewalls must analyze every packet in network communications, they often decrease network performance. They are also the single point of failure in the management of network security. Hence, if intruders break through the firewall, they may have unlimited access to the network the firewall is protecting.

Distributed firewalls [Bel99; IKBS00] were proposed in the context of the wire line Internet to overcome the assumptions made by traditional firewalls. Distributed firewalls attempt to solve two limitations of traditional firewalls. First, the assumption that everyone inside the firewall can be trusted and that everyone outside cannot be trusted is no longer true. Extranets can allow outsiders to reach the inside of the firewall; telecommuters' machines that are outside need protection when encrypted tunnels are not in place. Second, the need for external access for machines inside the firewall is not uniform. Some machines need more access to the outside than others do.

In distributed firewalls, as in traditional firewalls, policy is centrally defined. Enforcement, however, takes place on each endpoint. Thus, they retain the advantages of traditional firewalls while avoiding the problems described above due to dependence on topology.

Even though a distributed firewall solves the access control problem without dependence on topology, it has two shortcomings. First, access control is enforced only at the end hosts, not at the intermediate hosts. This causes bandwidth to be wasted, which can be a scarce resource in networks like bandwidth-limited MANETs. Second, the access control policies

state *who* can access the end hosts but do not say *how much* they can access. Nodes that have access to the end hosts can send as much traffic as they want. Because of these two limitations, the end hosts are susceptible to Denial of Service (DoS) attacks, even in the presence of a distributed firewall. It is important to allocate bandwidth for the services accessed by the hosts. It is also important that the hosts providing the service have a say in that bandwidth allocation, as these hosts are in the best position to make that decision based on the dynamic load.

MANETs are susceptible to another type of attack that is not common in the wired Internet. Due to the lack of structure and hierarchical organization (i.e., subnets) in the topology, unlike the wired Internet, MANET routing protocols may generate lot of traffic. As the topology may be changing due to the nodes joining and leaving the MANETs, and due to node mobility, the number of route requests initiated can be large. Hence, it is possible for a rogue node to launch attacks using the routing protocols alone, without fear of detection. Distributed firewalls that perform access control at the end hosts cannot prevent these kinds of attacks.

In this thesis, we overcome the above limitations by introducing a **DIstributed PoLicy enfOrceMent Architecture (DIPLOMA)** for MANETS. The architecture enforces the security policy at both the end hosts and the intermediate nodes. The policy contains the access control information as well as the allocated bandwidth for the access. Both the administrator and the end nodes can allocate the policy.

The security enforcement models in operating systems motivate our solution. Operating systems have to deal with many resources (or objects) such as memory and files that are shared and accessed by multiple processes or users. Each user has different access rights to different objects, which are represented by an access matrix. The conventional method of maintenance of this protection information is to use an *access control list*, in which an access list is associated with each object. Each object's list contains the names of users permitted to access the object and the privileges they may exercise. When a user attempts to access an object, the operating system checks the access list associated with that object to see if the operation is authorized.

The capability system offers an alternative structure in which the operating system

arranges the protection information by user instead of by object. A capability list is associated with each user in the system. Each capability contains the name of an object in the system and the user's permitted privileges for accessing the object. To access an object, the user specifies a capability from the local capability list. The operating system verifies the capability and allows access to the object.

The capability system offers advantages over the access control list-based system. In the access control list system, a malicious user can try to access any object by just naming the object. The system gives access to the object if the access list turns out to be giving the access. In the capability system, however, a user can try to access only the objects for which capability is held. Hence, the capability system has a better chance of limiting damage on a poorly configured system.

The traditional firewall and the distributed firewall follow the access list model. The firewall or the end host has an access list in the form of firewall policies. Any host can attempt to access any service. It gets the access if the policy allows for it. We propose to use the capability-system model for the firewall in DIPLOMA. The hosts need to hold a capability to access a service. Only hosts that have the capability to access a service can try to access it.

This model of firewalls is useful for a highly dynamic environment like MANETs. There are three factors against using the traditional firewall model in MANETs. First, the nodes participating in a MANET are not known in advance. New nodes may join or existing nodes may leave the MANET at any time. Therefore, it is not practical to have the access list populated in advance for MANETs. Second, traditional firewalls use IP addresses or subnets to identify the hosts in the access list. The IP address of a node may change as it leaves one and joins another MANET. Multiple nodes may get the same IP address on a given MANET at different times. Hence, it is not possible to have the access list based on IP addresses or subnets. It has to be based on a unique identifier like the public key. Finally, multiple nodes (in fact, all nodes) need to act as a firewall in a MANET. When the access list needs to be updated, it has to be updated in all of those nodes, many of which may be offline. A capability-based firewall model does not have these shortcomings. The firewall rule that allows access to a node is "carried" by the node itself. Whenever a new

node is introduced into the system, it is provided with all the access rules. These rules do not depend on the IP address it receives. In addition, there is no issue of synchronizing firewall rule updates, as there is only one copy of the rule; it is with the node that needs the access.

Another advantage of a capability-based system is the prevention of snooping attacks. In the current IP architecture, there is no validation of the source IP address field. A host can try to send a packet to a destination for which it does not have access by changing the source IP address field in the packet. This is typically used for launching an attack without disclosing the identity of the attacker, as the return packets do not reach the attacker. In wired networks, source IP filtering (ingress filtering [FS98]) and IP trace back are used against these attacks [SWKA00; DFS02; MMAK06]. It is much harder to solve this problem in MANETs due to lack of structure and frequent route changes. In the capability-based access control system, a node requires cryptographically verifiable tokens to send packets. Hence, it is not possible to launch these kinds of attacks.

In this thesis, we also extend the DIPLOMA architecture to secure multicast traffic. Multicast traffic, such as live audio/video streaming, is an important application for MANETs, including those used by military and disaster recovery teams. The open nature of multicast, where any receiver can join a multicast group and any sender can send to a multicast group, makes it an easy vehicle for launching Denial of Service (DoS) attacks in resource-constrained MANETs. Multicast security protocols for wired networks have treated receiver access control and sender access control as two separate problems [JA02]. We use capabilities to provide a unified solution for sender and receiver access control to the multicast groups, as well as to limit the bandwidth usage of the multicast group.

Hence, the proposed architecture of DIPLOMA encompasses the following benefits.

1. The advantages of distributed firewalls over the traditional firewalls for the support of lack of perimeter.

2. Protection of network bandwidth, which is not addressed by distributed firewalls.

3. The advantage of capability-based security compared to access control list-based security.

Our architecture falls into the category of consent-based network architectures [SNW+09; NSW+10] where the nodes require permissions to send traffic. In consent-based architectures, the permission to send the traffic to a destination may be on a specific path [SNW+09], or on any path [EMT89]. If the consent is on a fixed path, we call the system *path-based* consent networking. If the consent is on any path to the destination, we call the system *destination-based* consent networking. Only destination-based consent networking is suitable for highly dynamic environments like MANETs, as the path to a destination can change frequently. DIPLOMA falls under this category. Furthermore, DIPLOMA allows consent to send traffic to be given to a group of destinations, rather than only one destination.

In MANETs, it is possible for a source node to reach the same destination on multiple disjoint paths. In those cases, it is possible for rogue nodes to overcome the bandwidth enforcement of DIPLOMA by using the same capability in multiple node disjoint paths. In this thesis, we overcome these types of attacks on destination-based consent systems by detecting those attacks and taking actions against the attackers. We provide an efficient distributed algorithm for detecting misuses of capabilities in DIPLOMA.

DIPLOMA has following components:

- **A policy language that encodes access control and bandwidth allocations:** The policies represent permission to access a particular network service by a particular host. These policies also contain limits on bandwidth usage. These policies are cryptographically verifiable and are generated by the nodes that have authority to do so. Nodes can allocate access to services using these policies. An administrator/group controller distributes the policy strings to all nodes. These nodes generate the policy strings containing smaller allocations and distribute those policies to other nodes in the network.

- **Protocols for communicating the policy:** Hosts that need to access the network and the end host services need to carry proof of accessibility. These proofs are policy strings that are issued by the administrator or by the nodes that have the capability to do so.

- **Enforcement of policies at all the nodes:** The behavior at the nodes is to deny

access to the network and the services, unless a policy allowing the access is presented. This enforcement is done in an efficient manner without incurring large overhead on the processing required at the nodes as well as on the network bandwidth.

1.1 Thesis Contributions

The contributions of this thesis are:

1. The first comprehensive security architecture for multi-hop wireless networks that can protect bandwidth resources and end host services [AKS09]. This deny-by-default architecture, called DIPLOMA, enforces trust relationships and traffic accountability between the wireless nodes through a distributed policy enforcement scheme. We achieve this by extending the concept of network capabilities and tailoring it to the broadcast and resource-constrained MANET environment. This architecture allows compromised nodes to access only authorized services, limiting their ability to disrupt or even interfere with the end-to-end connectivity and the nodes beyond their local communication radius.

2. A preliminary implementation and evaluation of the architecture in GloMoSim simulator [ASK09]. Our evaluation shows that the system has minimal overhead in terms of latency and bandwidth. It also shows that the system can work even in the presence of route changes due to mobile nodes, and it can protect against misbehaving nodes.

3. An implementation of the architecture in real systems running the Linux operating system [AK10a]. The implementation works at the network layer and does not require any changes to the existing applications. We identify the bottlenecks in the original architecture and provide improvements, including a signature optimization, so that it works well in practice. We evaluate the architecture in a realistic MANET testbed Orbit. The results show that the architecture incurs minimal overhead in throughput, latency, and jitter. We also show that the system protects the network bandwidth and the end-hosts in the presence of attackers. For the evaluation, we identify ways of creating multi-hop topologies in indoor environments so that a bad node cannot interfere with every other node.

4. An extension of the architecture to secure multicast traffic [AK10c]. This is the first unified solution to achieve sender and receiver access control to the multicast groups, as

well as to limit the bandwidth usage of multicast groups. We have extended the common multicast protocols, including ODMRP and PIM-SM, to incorporate DIPLOMA. We have implemented multicast DIPLOMA in Linux without requiring any changes to the existing applications and the routing substrate. We conducted an experimental evaluation of the system in the Orbit MANET testbed and simulations using GloMoSim simulator. The results show that the architecture incurs limited overhead in throughput, packet loss, and packet inter-arrival times. We also show that the system protects multicast traffic in the presence of attackers.

5. We identify the misuses in destination-based consent networking like DIPLOMA and provide solution for detecting and recovering from these misuses [AK10b]. We identify how senders and receivers can misuse capabilities by using them over multiple paths, and we provide distributed solutions for detecting those misuses. To that end, we modify the capabilities to aid in misuse detection and provide protocols for exchanging information for distributed detection. We also provide efficient algorithms for misuse detection and protocols for providing proofs of misuse. Our solutions can handle privacy issues associated with the exchange of information for misuse detection. We have implemented misuse detection and recovery in DIPLOMA systems running on Linux operating systems and have conducted extensive experimental evaluation of the system in the Orbit MANET testbed. The results show that our algorithms require minimal processing and memory and that distributed detection requires a very small amount of bandwidth. We also show that our system is effective in detecting and containing multi-path misuses.

1.1.1 What is not addressed in the thesis

This thesis mainly addresses the architectural and protocol aspects of distributed policy enforcement in MANETs. This thesis does not address how these policies are defined. The definition of the policy in DIPLOMA involves two aspects: access control and bandwidth allocation. In MANET networks where the roles of the nodes are well defined, like military networks, policy allocation may be based on the roles and the tasks assigned to the nodes.

This thesis also does not address how the nodes split the bandwidth among the individual connections using the policies allocated to them. This allocation may depend on the services

that are running on the node, the current load on the system, and the priority of the clients connecting to it.

1.2 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 gives an overview of the background and related work. Chapter 3 introduces the DIPLOMA architecture, the threat model, and the security analysis and gives the details of the protocols. Chapter 4 provides the results of an early evaluation of the system using simulations with GloMoSim simulator. Chapter 5 provides the details of implementing DIPLOMA on real systems running Linux and an evaluation of the system using the Orbit lab testbed. Chapter 6 gives the extensions of DIPLOMA for multicast traffic and their evaluations. Chapter 7 identifies possible misuses of DIPLOMA architecture and provides solutions for detecting and recovering from these misuses. Chapter 8 discusses future work.

Chapter 2

Background and Related Work

In this chapter, we first introduce multi-hop wireless networks, including MANETs, and describe some of the security challenges facing those networks. Then we describe work related to this thesis in detail.

2.1 Multi-hop wireless networks

A multi-hop wireless network consists of nodes communicating over multi-hop wireless links to provide end-to-end connectivity. In multi-hop wireless networks, one or more nodes in the path receive and send packets over wireless links. There are several benefits of multi-hop networks. They are easy to deploy and improve connectivity. Due to transmission over multiple short links, they require less power and offer improved spectral efficiency and throughput compared to networks with single wireless links (e.g., cellular networks). The nodes in the multi-hop networks have one or more wireless antennas. With multiple antennas, the nodes can use non-overlapping channels to improve the capacity of the network [KV05; ABL05]. A dense multi-hop wireless network can improve robustness due to the availability of multiple paths. A recent study [Mun09] has shown that more than 300 cities and counties in the U.S. have deployed either multi-hop wireless networks or Wi-Fi access for their residents.

Mobile ad-hoc networks (MANETs) [Per01; PH02; AGI05] (Figure 2.1) are multi-hop wireless networks where some or all of the nodes may be mobile. Mobility poses additional

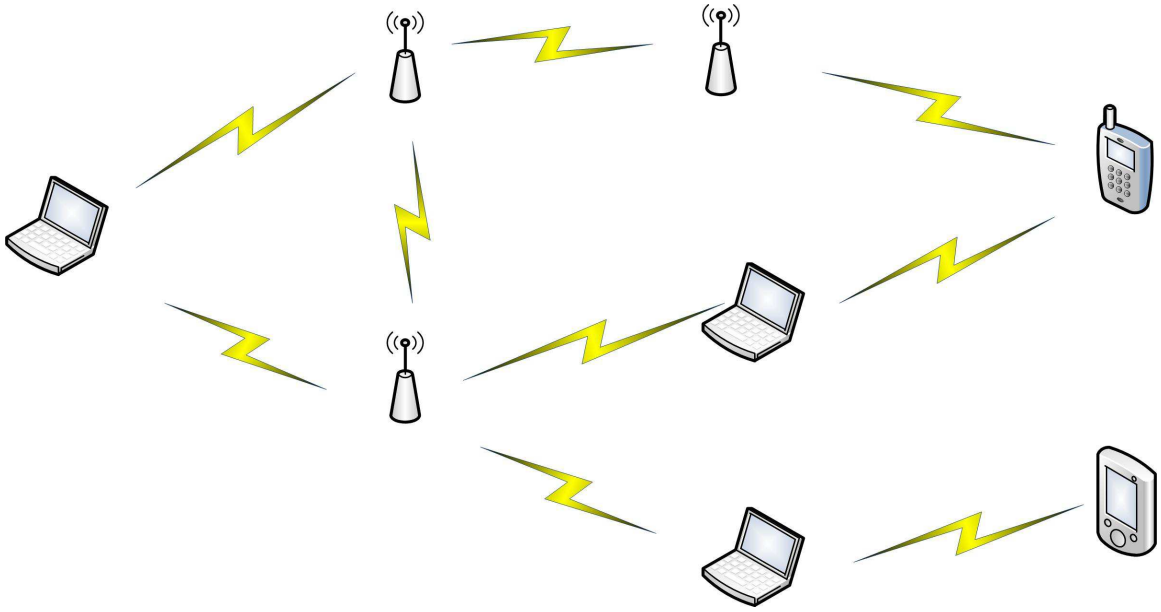

Figure 2.1: A mobile Ad hoc Network

challenges in multi-hop wireless networks. MANETs need to maintain connectivity during user mobility. They typically have limited resources, in terms of computing resources, memory, and battery life. In a MANET, the nodes have dual roles as end hosts and routers. They act as end hosts for the communication sessions where they are the end points of the traffic. They act as routers for the communication sessions of other nodes by receiving and relaying the traffic over wireless links to achieve end-to-end connectivity.

Because of the dual role of the nodes and the broadcast nature of the medium, many of the security solutions used in wired networks are not applicable to MANETs. In wired networks, routers are specialized nodes controlled by the administrators of the domain and are well trusted. In MANETs, however, all the nodes participating in them perform routing functionality. Not all of these nodes can be trusted. Even in wired networks, relying on the security of routers to provide security to hosts leads to a single point of failure. In wired networks, the packets going on a link cannot be heard by the nodes that are not connected to that link. In MANETs, however, all the nodes in the transmission range of a sender can hear the packets transmitted by it. Many security solutions in wired networks take advantage of the trusted nature of the routers and the privacy of the communication links [ARW03;

YPS04; YWA05].

2.2 Related Work

2.2.1 Network Capabilities

The concept of capabilities was implemented in early computer systems [Lev84]. A chronology of the evolution of the capability model is given in [Tul06]. A capability is a communicable, unforgeable token of authority. It can be used for system security in the place of forgeable references. Many operating systems use it for access control of resources[WABL94].

Estrin *et al.* [EMT89] proposed "visas for packets for allowing controlled exposure of resources at the network layer, which are similar to network capabilities. Visa protocol deals with controlling the flow of information across organizations by authenticating and authorizing the flow of datagrams at the source and the destination gateways. Routers other than the source and the destination gateways do not participate in the protocol. They also do not enforce any bandwidth constraints. More recently, network capabilities were proposed to prevent DoS in wired networks [ARW03; YWA05]. In that architecture, the nodes must obtain permission to send from the destination in the form of tokens or capabilities. The sender includes these tokens in the packets. The intermediate routers (verification points) verify that packets contain those tokens.

We extend the concept to MANET and use it for both access-control rules and traffic-shaping parameters. In the previous approach, the capabilities are assigned only by the receivers, and there is no limit on the amount of capability that a receiver can assign. Though it achieves the goal of preventing a DoS attack at the receiver, it does not prevent two nodes from taking up all the available network resources. Previous solution also assumes that the links in the path between a sender and a receiver cannot be snooped and that the path is fixed. They also assume that the routers can be trusted. These assumptions are reasonable for the wire line systems that their solution is designed for but do not work for MANETs.

2.2.2 Distributed Firewalls

Distributed firewalls [Bel99; IKBS00; Li00; GAA01] overcome some of the shortcomings of traditional firewalls. In distributed firewalls, a firewall is placed at each host in the network. However, the distributed firewalls are managed centrally. As the policies are defined centrally and enforced in a distributed way, distributed firewalls do not have dependence on the topology. They also eliminate the network bottleneck by removing the single point of policy enforcement.

Previous work on distributed firewalls focused on wired fixed-network environments and attempted to protect only the end hosts using a host-based solution. These firewalls did not protect network bandwidth. Our solution is for a mobile network using a combination of network (*i.e.,* forwarding nodes) and host-based solutions that attempt to protect both the network and end-host resources.

Routing As a Firewall Layer (ROFL) [ZCB08] is a firewall architecture that treats port numbers as part of the IP address. Hosts permit connectivity to a service by advertising the IPaddr:port/48 address; they block connectivity by ensuring that there is no route to it. This architecture provides greater protection against insider attacks than do conventional firewalls but drops unwanted traffic far earlier than distributed firewalls do. Subsequent work on ROFL [ZJCB09] includes source prefix constraints also in route announcements, accomplishing the complete set of filtering functionality provided by traditional firewalls.

Ethane [CFP+07] is a network architecture for enforcing a single-network-wide fine-grain policy in enterprises. It has a centralized controller that manages the admittance and routing of flows in Ethernet switches. The switches are dumb elements that forward packets under the direction of the controller. Ethane uses a centralized control plane that requires connectivity to the controller for admittance of new flows. Our architecture is distributed in nature, where the individual nodes make decisions based on the capability presented to them.

2.2.3 Packet Authentication

Signing and verification of packets between a sender and a receiver were commercially available in early 1990s. Novell's Netware 3.11 and 4.x supported *NCP Packet Signature*

Option, where a unique signature was appended to each packet sent between the client and the server [Lee93]. The keys for the signatures were negotiated at login time. Intermediate nodes were not involved in packet verification.

Mitigating denial of service attacks by including a message authentication code and the certificate of the sender for each packet has been previously proposed in [WY07]. They do not study the high overhead associated with sending a large signature or a large certificate on each packet. The authors use game theory to study the problem of dealing with selfish nodes that do not verify the packet signatures, using incentives and punishments. This mechanism or any other reputation-based mechanism [JS07] can also be used in our scheme to deal with selfish nodes.

HEAP [AKR07] mitigates various MANET attacks from outsider nodes by doing a hop-by-hop packet authentication using HMAC. MACs (end-to-end or hop-by-hop) cannot deal with insider attacks. They also cannot provide access control unless different MAC keys are used for different policies. Even with different keys, MACs allow rogue nodes to "hide." MACs are repudiable, as all the intermediate nodes in the path between a sender and a receiver need to know the key. Only asymmetric key mechanisms can allow validation by all intermediate nodes that the packets were indeed sent by the source node of the packet.

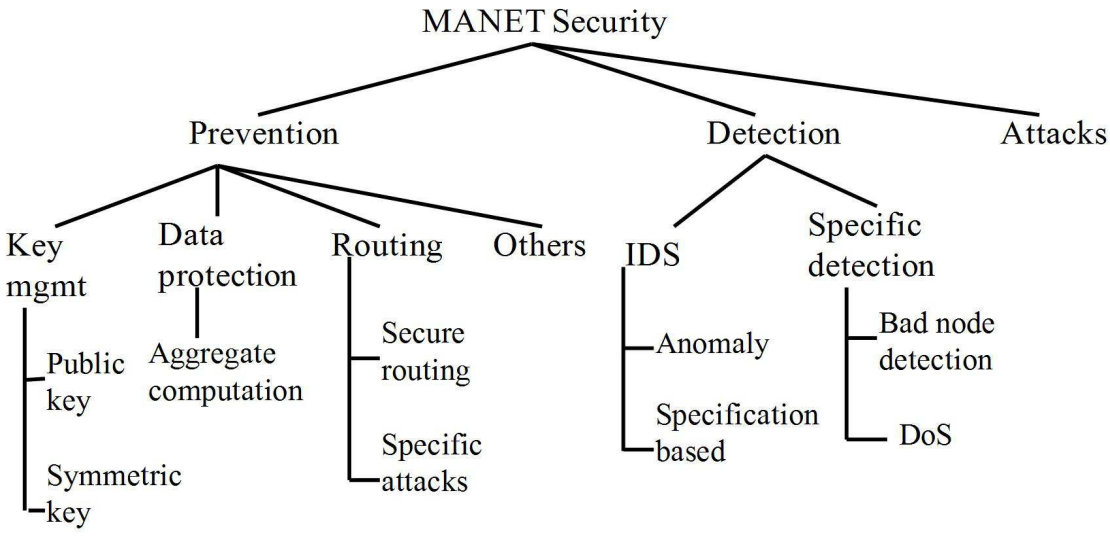

Figure 2.2: A classification of security research in MANETs

2.2.4 MANET Security

Security for mobile *ad hoc* networks is an active area of research. Figure 2.2 gives a high-level taxonomy of security research on MANETs. Most of the prior research on MANET security focused on solving specific problems or retrofitting security into an existing IP-based network architecture; we are trying to introduce a new architecture where security is built into the network. In this section, we give a sample of research in the MANET security area. Surveys of research on MANETs can be found elsewhere [WCWC06; YLY+04; SP04; DKB05; LdSP09].

The majority of the security research in MANETs has focused on three areas: prevention, detection, and identification of possible attacks. Prevention efforts have focused on modifying the protocols to prevent various attacks identified in the protocol. Many solutions in prevention use cryptography for protecting the protocol packets. Hence, key management in MANETs for these protocols has received a lot of attention. Detection work has focused on defining intrusion detection architecture for resource-constrained MANETs, as well as detecting specific attacks like denial of service attacks. There has been lot of focus on analyzing and identifying possible attacks on popular MANET protocols like routing protocols.

Key management is an important building block of any security system. There are various schemes proposed in the literature for establishing symmetric keys between a pair of nodes in the presence of a central authority by the random pre-allocation of key shares. The central station generates a large set of random keys and allocates a subset of the random keys to each node. There is a shared key between two nodes if they receive at least one common key from the central station [EG02]. If two nodes do not share a common key, a shared secret can be introduced by a third node that shares keys with both nodes. Extensions of this scheme, where two nodes require more than one common key to establish a secret key, are proposed in [CPS03]. Use of a pool of random bivariate polynomials, where the central server distributes the projection of the polynomial to one of the variables, is proposed in [LN03]. Extensions are proposed to Blom's [Blo82] key distribution scheme that uses multiple space Blom's scheme [DDHV03]. In this scheme, each node participates in a subset of the set of Blom's scheme. All these schemes are particularly useful for sensor

networks, which have limited memory and computing. LEAP [ZSJ03] identifies various key types used in a sensor network and proposes solutions for each of those keys.

Many papers have addressed public key distribution under the assumption of the unavailability or the compromise of certificate authority. Threshold cryptography [DF89] is proposed for use in MANETs, where the private key is divided into multiple shares and distributed to the nodes [ZH99]. At least a threshold number of shares is required for the signing operation. They also propose to refresh the share proactively without disclosing the secret keys. Another proposed approach is for users to issue their own public keys and to trust a limited amount of other certificates [HBC01]. When two users want to communicate, they find a common certificate chain by merging the repository of trusted certificates. This is similar to PGP [Zim93]. PICO [KC09] is a distributed protocol for managing group membership and keying MANETs. It uses threshold cryptography to implement its services. PICO tolerates a limited number of Byzantine nodes and an additional limited number of crashed nodes.

In our proposed architecture, we make use of public keys to sign the policy tokens. Our system does not require the central server to be available all the time. A node does not need to contact the central server once it has policies that allow it to access the services provided by the servers in MANET. Furthermore, when the servers derive smaller policy tokens (capabilities) to distribute to the other nodes, a central server is not required.

Research has also focused on secure data and global time in the presence of malicious nodes [CPS06; ZSJN04; SNW+06]. Many protocols in MANETs depend on global time synchronization. For example, μTESLA uses delayed disclosure of keys to achieve authenticated broadcast, which requires loose time synchronization [PSW+01]. In time synchronization protocols, due to the shared nature of the wireless medium, the nodes need to record the time only when the packets are guaranteed to leave. TinySeRSync [SNW+06] uses a MAC layer time stamping for pair-wise time synchronization and uses authenticated broadcast μTESLA for global time synchronization. For securing aggregated data from false injection of values, hierarchical solutions are used, where the data are collected in clusters and sent to the upstream nodes for aggregation [CPS06; ZSJN04].

Securing the routing protocols in MANETs is an active area of research [MMDM04].

A major differentiator of MANETs from wire line networks is the need for specialized routing protocols, as the topology is not static and not structured. Routing protocols in MANETs fall into three categories: on-demand routing, link-state routing, and hybrid approaches. In on-demand routing, routes are discovered when there is a need to send data. Popular on-demand routing algorithms are Ad hoc On Demand distance Vector routing (AODV) [PBRD03] and Dynamic Source Routing (DSR) [JMH03]. Link state algorithms maintain a topology of the network. Link state routing algorithms exploit the periodic exchange of control messages between the routers to ensure that the route to every host is always known. Popular link state algorithms are Optimized Link State Routing (OLSR) [CJ03] and Topology Broadcast based on Reverse-Path Forwarding (TBRPF) [OTL03]. Another popular proactive protocol that is distance vector based is Destination Sequenced Distance Vector (DSDV) [PB94]. In this proposal, we will focus on on-demand routing protocols.

Routing attacks on MANETs mostly fall into three categories: forging the initiated routing packets, forging the forwarded routing packets or the node identity, and dropping the forwarded packets. For detecting forged initiated routing packets, an IDS like DEMEM [TWKL06] can be used. Cryptographic techniques are used to prevent the forging of forwarded packets or the node identity. Dropping of the forwarded packets can be detected using sender-based acknowledgements. Those problems can be prevented using reputation-based mechanisms.

Various attacks are identified on the routing protocols, many of which are very specific to MANETs. In rushing attacks [HPJ03a], the attacker forwards the route requests more quickly than others, forcing the route to go through the attacker. This is avoided first by nodes detecting the neighbors securely using strict timing of the messages, then delegating the routing messages only to these secure neighbors, and finally randomizing the route requests in which the nodes collect a lot of route requests and forward just one. In wormhole attacks [HPJ03b], an adversary has control over two nodes that it connects through a low-latency link. The nodes use this link to forward the route requests, forcing the route to take that path. This attack depends on a node misrepresenting its location. Hence, directional antennas can be used to prevent this problem [HE03]. Location-based routing protocols can

also potentially prevent these attacks [KW02]. In black hole attacks [DLA02], a malicious node advertises itself as having the shortest path to the node whose packets it wants to intercept. Solutions proposed for these attacks are based on identifying and avoiding the malicious nodes [RFS+03]. In ad-hoc flooding attacks [YDZZ05], the attacker floods the MANET with route request packets. These attacks can be prevented by limiting routing-request traffic. The policy framework presented in this thesis can be applied to both data and protocol traffic.

Various routing protocols have been proposed to secure existing MANET routing protocols. Secure Efficient Ad-hoc Distance Vector (SEAD) [HJP02] is based on the proactive routing protocol DSDV. In SEAD, one-way hash chains are used along with routing metrics to prevent malicious nodes from advertising non-existent shorter paths. ARIADNE[HPJ02] is a secure routing protocol based on DSR. It cryptographically signs the route discovery and route maintenance messages to avoid forging. Security Aware Routing (SAR) [KYN01] is a routing protocol based on AODV. It integrates the trust level of a node and the security attributes of a route to provide a security metric for the requested route. A Secure Routing Protocol for Ad Hoc Networks (ARAN) [SDL+02] is an on-demand protocol for managed environments. Nodes use certificates provided by a trusted third party to authenticate to other nodes during the exchange of routing messages. The architecture proposed in this solution can work with any of the above protocols. Furthermore, we can protect the above protocols from DoS by limiting the number of protocol packets using capabilities.

Solutions are also proposed to specifically detect the attacks on routing protocols based on the protocol specification. An extended finite state automation (EFSA) of AODV is used to detect violations in the protocol specification in [HL04]. The Distributed Evidence-Driven Message Exchange Intrusion Detection Model (DEMEM) [TWKL06] detects inconsistency among the routing messages in OLSR. Modeling of attacks on AODV protocol using an attack tree to identify the damage is presented in [EB06]. Proposals are also made regarding when to isolate a misbehaving node based on the criticality of that node in maintaining the connectivity [WTLB07]. There is also a rich literature on detecting denial of service and node replication attacks on sensor networks [MSPR05; PPG05].

BARTER [FMSK09] is a behavior-based access and admission control system for MANETs.

In that system, the nodes initially exchange their behavior profiles and compute individual local definitions of normal network behavior. During admission or access control, each node issues an individual decision based on its definition of normalcy. Individual decisions are then aggregated via a threshold cryptographic infrastructure that requires agreement among a fixed amount of MANET nodes to change the status of the network. BARTER works at the application layer and hence cannot protect the network bandwidth. It can co-exist with DIPLOMA to provide additional security based on user behavior.

2.2.5 MANET Intrusion Detection

Intrusion detection systems (IDS) for MANETs are an active area of research [MNP04]. MANET nodes are expected to be co-operative in nature. Hence, it is easy to launch attacks by malicious nodes. MANETs are distributed in nature, without a centralized point to control the network. MANETS have limited bandwidth and processing power. The links are much more unreliable due to the mobile nature of the nodes. An IDS architecture needs to consider these factors. In the Local Intrusion Detection Architecture (LIDS) [ACJ+02], communities of nodes are formed that exchange various security data and intrusion alerts. The nodes can also place mobile agents in the other nodes to do a specific mission in an autonomous and asynchronous manner. Distributed IDS architecture [ZL00] is another proposed IDS architecture for MANETs. It uses a local detection engine with input from the local data collection and a co-operative detection engine with input from neighboring nodes to detect intrusions.

In trust-based security systems for ad-hoc networks [YZV03; BR08], the security decisions on data protection, secure routing, and other network activities are based on the evaluation of trust by each node. In those systems, each node evaluates other nodes based on trust factors such as experience statistics, data value, intrusion detection results, references from other nodes, etc.

In this thesis, we provide solutions for detecting and recovering from misuses of capabilities. Misuses may result from the use of the same capability in multiple paths to a destination or to different destinations. Our model is based on the distributed IDS architecture [ZL00], where we use a combination of local and distributed detection that uses

minimal network bandwidth and CPU processing.

2.2.6 Multicast Security

A survey of security issues and solutions for multicast in wired networks is presented in [Amm03]. Multicast groups have the following three characteristics. First, all the members receive all the packets sent to a multicast group. Second, any node can join a multicast group. Third, any node can send packets to a multicast group. The authors classify the issues and solutions based on these three properties. These solutions are specific to wired networks and not directly applicable to MANETs, which have no specialized router nodes.

A number of solutions have been proposed for multicast receiver access control [JA02; HC00; BC95]. These solutions have trusted routers or query centralized servers; thus, none is suitable for MANETs. These protocols do not also have limitations on the amount of service accessed. DIPLOMA provides a unified solution to both receiver and sender access control and supports bandwidth constraints.

The ZODIAC [ABC+09] system provides an application-to-application model of security using the concept of a Dynamic Community of Interest (DCoI). A DCoI is a dynamic group of networked nodes whose membership, application, and resources are regulated by its members as constrained by policy. The nodes enforce the DCoI policies hop-by-hop, dropping the traffic that is not cryptographically authorized and protected or that violates pre-negotiated constraints. Both DIPLOMA and ZODIAC systems are based on a deny-by-default principle and enforce policies at each hop between the sender and the receiver. In DIPLOMA, the policies are defined by the central controller, whereas in ZODIAC, nodes explicitly join the DCoIs. A node can join multiple DCoIs, one per application. In addition, ZODIAC requires that all the nodes in the path between a source and a destination be part of the DCoI. In DIPLOMA, the capabilities are decided by the sender and the receiver based on the policy allocated by the central controller; the intermediate nodes do not have any influence on it.

There are a number of multicast routing protocols proposed for MANETs. A survey of these protocols is present in [CGA03]. There has been research dealing with the security issues of these protocols. A discussion of possible attacks on MAODV (Multicast-

extended AODV) routing can be found in [RASJ05]. The authors also propose an authentication framework to protect an MAODV network against these attacks. Tactical MAODV [SKC+09] extends MAODV through the integration of the security services necessary for the tactical deployment of MANETs, such as forward and backward secrecy and data confidentiality. Securing multicast MANET protocol MMARP with digital signatures is presented in [GRGSK05]. They use Cryptographically Generated Addresses (CGA) to keep attackers from impersonating other nodes. A protocol to secure communication in multicast groups with a pair of multicast trees for each multicast group is presented in [KLNY03]. They use one multicast tree for security information and a different tree for data traffic.

Chapter 3

DIPLOMA Architecture

Mobile Ad-hoc Networks (MANETs) are increasingly employed in both military and commercial network situations where fixed infrastructure is too costly or dangerous to deploy, has been rendered inoperable, or nodes are mobile. MANETs are fundamentally different from the Internet because all peers act as both sources and routers using other participants to relay packets to their final destination. MANETs are susceptible to both insider and outsider attacks. Even a small number of misbehaving nodes can successfully render the entire MANET inoperable: malicious peers can abuse the network exhausting all network and power resources.

Traditionally, security policies of a network are enforced by *firewalls*, which are placed at the perimeter of the network. Unfortunately, the concept of a network perimeter does not exist in MANETs, and policies need to be enforced in a distributed manner while taking into consideration node mobility. To address this, we propose an architecture that enforces trust relationships and traffic accountability between mobile nodes through a novel policy enforcement scheme designed specifically for MANETs. We extend the network capability framework [PWS+07; ARW03] and we tailor it to the resource-constrained MANET environment. A capability is a token of authority that has associated rights. In our model, capabilities propagate both access control rules and traffic-shaping parameters that should govern a node's traffic. To that end, we define a protocol for communicating capabilities, which are treated as soft state, across the MANET. We name our architecture DIPLOMA, which stands for **DI***stributed* **P***o***L***icy enf***O***rce***M***ent* **A***rchitecture*.

Our architecture enables the enforcement of adaptive bandwidth constraints inside the network, denying by default any unauthorized traffic. Nodes can only access the services and hosts they are authorized for by the capabilities given to them. Compromised or malicious nodes cannot exceed their authority and expose the whole network to an adversary. Upon detection, we can prevent a compromised node from further attacking the network simply by revoking its capabilities. Moreover, our architecture helps mitigate the impact of denial of service (DoS) attacks because excess or unauthorized packets are dropped closer to the attack source. Thus, we avoid unnecessary data processing and forwarding at the target node and the network itself.

Even though we focus on MANETs, our system can also be used in wired networks. However, MANETs provide our architecture both advantages and challenges. Specifically, the ratio of CPU cycles to available bandwidths (Hz/kbit) is normally higher in MANET nodes compared to their wired counterparts. This enables us to do more intelligent processing (and use cryptography) on most or all of the packets transiting through a MANET node. The number of traffic flows handled by a MANET node is also small due to the small network size. However, frequent route changes between a source and a destination node due to node mobility represents a difficult challenge in a distributed enforcement environment such as ours.

The rest of the chapter is organized as follows. We begin by describing the threat model in Section 3.1. We then present the system architecture and a high-level overview of our scheme in Section 3.2. Then we present the security analysis of DIPLOMA in Section 3.3. We present the processing that is required at the nodes in the new architecture in Section 3.4, and the protocol details in Section 3.5. We present a packet signature optimization scheme for achieving high send rate in Section 3.6. Finally, we discuss some of the design choices of DIPLOMA in Section 3.7.

3.1 Threat Model

Our goal is to protect network resources and end-node services from denial of service attacks, and to enforce access control rules in the absence of a fixed topology. Thus, we want a node

to be able to access only the services it is entitled to, and to limit the amount of traffic that can be sent to any such service. To preserve bandwidth and power, we need to filter any unauthorized traffic early on.

We assume MANET environments where an adversary may be an existing node that has been compromised (insider) or a malicious external node that might want to participate in the MANET. In addition, there may be multiple cooperating adversaries; and compromised nodes may not be detected as such immediately, or ever (depending on their actions).

The resources needed to access a service are allocated by the *group controller(s)* (GCs) of the MANET. Group controllers are nodes responsible for maintaining the group membership for a set of MANET nodes, and *a priori* authorize communications within the group. This means that GCs do not participate in the actual communications, nor do they need to be consulted by nodes in real time; in fact, if they distribute the appropriate policies ahead of time, they need not even be members of the MANET. In some cases, the GC may be reachable through a high-energy-consumption, high-latency, low-bandwidth long-range link (*e.g.,* a satellite connection); interactions in such an environment should be kept to a minimum, and only for exceptional circumstances (*e.g.,* for revoking access for compromised nodes).

Without compromising a GC, an external node can participate in a MANET only by stealing the authorization credentials that are bound to the identity of a legitimate node. Because we envision GCs as being primarily offline or, at best, intermittently reachable (with respect to the MANET), we are not addressing the issue of compromised controllers in this thesis.

If a node is compromised, an adversary can only access the services and bandwidth that node is authorized to access. If other MANET nodes are adhering to our architecture, a compromised node does not have the ability to disrupt or interfere with end-to-end service connectivity and other nodes beyond its local radio communication radius. The nodes providing services will receive only the traffic that the compromised node is authorized to transmit, unless the adversary is in the local communication radius.

3.2 System Architecture

In our architecture, there is one or more pre-defined nodes that act as a *group controller* (GC). These nodes are trusted by all the group nodes. For simplicity and without loss of generality, we will assume that all the MANET nodes are part of a single group. A group controller has authority to assign resources to the nodes in the MANET. These resources are expressed in terms of limits on the number of packets or on bandwidth rates that a MANET participant is permitted to transmit towards another node. The resource allocation by the GC to a node is represented using a credential called *policy token* that all the nodes can verify. The policy tokens are typically provisioned ahead of time, and represent the projections of the centralized policy, even though an on-demand allocation from the GC is possible. The GC may be offline after it distributes the policy tokens, and may be reachable sporadically at best after that (as external connectivity permits). The presence of the GC is not required, after the initial policy token distribution, for the normal working of the protocol.

When a node (initiator) requests a service from another MANET node (responder) using the policy token assigned to the initiator, the responder can provide a capability back to the initiator. This is called a *network capability*, and it is generated based on the resource policy assigned to the responder and its dynamic conditions (*e.g.,* level of utilization).

All the nodes in the path from an initiator to a responder (*i.e.,* nodes relaying the packets) are required to enforce and abide by the resource allocation encoded by the GC in the policy token and the responder in the network capability. The enforcement involves both accessibility and bandwidth allocation. A responder accepts packets (except for the first one) from an initiator only if the initiator has authorization to send, in the form of a valid network capability. An intermediate node will forward the packets from a node only if the packets have an associated policy token or network capability, and if they do not violate the conditions contained therein. Note that the possession of a network capability does not imply resource reservation; they are the maximum limit a node can use. Available resources are allocated by the intermediate nodes in a fair manner, in proportion to the allocations defined in the policy token and network capability. Intermediate nodes cache policy tokens and network capabilities in a *capability database*, treating them soft state.

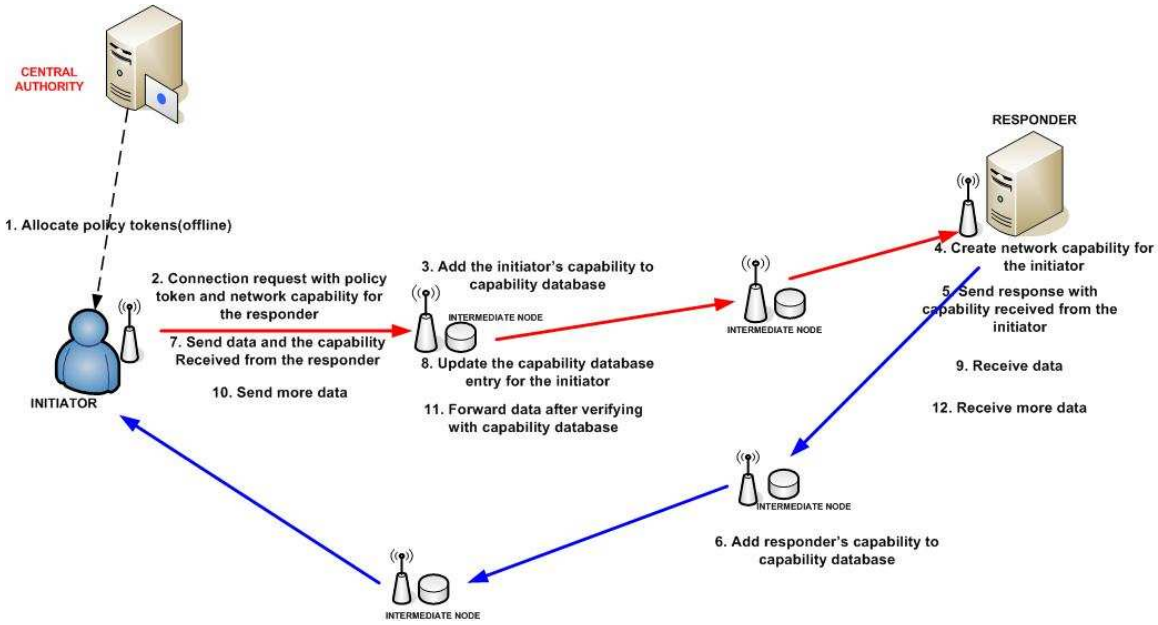

Figure 3.1: System overview

Figure 3.1 gives an overview of the protocol exchanges when an initiator wants to communicate with a responder. The initiator has a policy token previously issued by the GC that authorizes the communication with the responder (step 1). The initiator sends a communication request (and, optionally, initial data), along with its policy token toward the responder (step 2). This packet also contains a *transaction id* that the initiator will use in subsequent packets to the same responder. The packet may also contain a network capability that the initiator generates; this can be used by the responder to communicate back to the initiator. Here, we assume that the initiator has a routing table entry for the responder. Otherwise the underlying routing protocol will be invoked to get the route. An intermediate node will forward the packet only after validating it. The validation involves cryptographic verification of the capability, and verification of the constraints (*e.g.,* bandwidth usage, service and destination address) specified in the policy token. If the validation is successful, the intermediate node also records the policy token in its capability database, along with other attributes of the packet, such as source and destination node address and the transaction id (step 3). This step is performed at each intermediate nodes.

The responder, on receiving the packet verifies the policy token and creates a network capability for the initiator (step 4). The responder sends the response to the request as well as the newly created network capability for the initiator (step 5). The responder also creates a transaction id for the communication, and includes it in the response. The responder also needs to include the network capability it received from the initiator in the first message, which authorizes it to communicate back; alternatively (or in addition), it may use a policy token issued by the GC to responder that is authorizing the communication with the initiator. Intermediate nodes, on receiving this packet from the responder, validate the packet and add the responder's policy token and network capability to its capability database (step 6). In the diagram, the reverse path is shown to be different from the forward path; the paths can also be the same. The initiator will then have to include the responder-issued network capability in subsequent packets it transmits (step 7); intermediate nodes will add this credential to their capability database (steps 8, 9).

Any further data traffic between the initiator and the responder does not contain the policy token and network capability; instead, it contains only the transaction id that was included in the initial handshake (steps 10-12). The packets are signed by the sender, and can be verified by the intermediate nodes. If the cost of the cryptographic operations is too high in terms of latency or power consumption, then the cryptographic validation may be done probabilistically. The probability is set as a system policy dependent on the security level and availability of resources. The intermediate nodes can validate the packets by looking at the policy token and network capability contained in the capability database corresponding to the transaction id in the packet. This process ensures that the packet does not exceed the resource limit allowed in the policy token and the network capability, and that it is authorized to reach the destination by both the GC and the destination itself. For this validation, the intermediate node also maintains the resource usage against each capability in its capability database. The only time the initiator or responder need to re-send the capability is when the path between them changes due to node mobility, or when the network capability expires and is reissued by the peer.

3.2.1 Feasibility

We argue that the proposed solution is feasible for MANETs, even though the memory and processing power are lower in MANET nodes compared to routers in wired networks. Our scheme requires memory to store the information about the traffic sessions, and CPU cycles for the cryptographic operations. The feasibility comes from the fact that the bandwidth in MANETs is significantly lower than that of wired networks, while the nodes are relatively powerful (*e.g.,* normal laptops, or high-end cell phone devices). As a result, the available memory and processing power per packet is higher in MANETs than in wired networks. The processing power per packet for MANET nodes are increasing everyday with the advent of faster but less power-hungry processors for portable devices.

Furthermore, the per-packet cryptographic operations, which involve a public key signature verification, can be achieved with very small key sizes. This is because, unlike traditional uses of public keys, these keys are useful only for the short duration of the session. For longer sessions, new keys can be generated and old ones discarded.

3.2.2 Capability definition

Each node has authority to send traffic to its peers at certain rates. This authority is encoded in the policy token and network capability. Both of these are represented by KeyNote-style credentials [BIK01]. Each credential contains

1. Identity of the node (principal)
2. (Optional) Identity of the destination node; if left unspecified, it applies to all destinations
3. Type of service and amount of data the principal is allowed to send
4. An expiration time
5. Signature of the GC (for policy tokens) or peer (for network capabilities)

All nodes in the MANET know the public key of the GCs, so that they can verify policy tokens issued by them. Identities are expressed in term of the long-term public key of

the node to which a credential is assigned. The destination node can be a host, a subnet, or a public key. Type of service refers to the transport protocol identifiers a credential authorizes.

Typically, the bandwidth available to a node on a network capability is higher than that of its policy token. Policy tokens are assigned by the GC, which has no knowledge of network load at the time the communication takes place. Hence, the central authority will consider the worst case scenario while assigning the policy token and permit only enough communication to take place for a handshake to occur. It is up to the responder to provide a network capability with enough bandwidth allocation to enable the communication to proceed. This allocation has to be within the constraints of the policy allocated to the responder. Note, also, that it is in the interest of a node to issue short-lived network capabilities to its communicating peers, so that it can quickly respond to changing network dynamics or (more importantly) to peer misbehavior (*e.g.,* a flood-based DoS).

Policy tokens and network capabilities have the same syntactic representation. Following is an example:

```
serial: 130745
owner: unit01.nj.army.mil (public key)
destination: *.nj.army.mil
service: https
bandwidth: 50kbps
expiration: 2012-12-31 23:59:59
issuer: captain.nj.army.mil
signature: sig-rsa 23455656767543566678
```

The above represents a policy token assigned by node captain.nj.army.mil to unit01. The unit can use this policy token to send the traffic to any node in the domain nj.army.mil. The peak data rate using this credential cannot exceed 50kbps.

If unit01 wants to communicate with unit02, it will send a message to unit02 using this policy token. Unit02 will issue a network capability for unit01, if the communication needs more bandwidth than available in the policy token.

```
serial: 1567
owner: unit01.nj.army.mil (public key)
destination: unit02.nj.army.mil
bandwidth: 150kbps
expiration: 2010:12:21 13:05:35
issuer: unit02.nj.army.mil
comment: Policy allowing the receiver
    to issue this capability.
signature: sig-rsa 238769789789898
```

This capability is restricted to be used only by unit01 for communication with unit02. It specifies a higher bandwidth, but a shorter expiration date. The issuer of the capability is the same as the destination of the capability.

After receiving this capability, unit01 will use this capability for communication with unit02. The more general policy token can be used by unit01 for communicating with other nodes.

If the communication from unit01 to unit02 was short and required low bandwidth, unit01 could have used its policy token for the entire duration of the communication, without requesting for a network capability from unit02. This will be faster for short communication as there is no capability request/reply, and unit02 does not have to issue any capabilities. If unit01 expects some messages from unit02 that require more resources than the one that is available to unit02 in the form of its corresponding policy token, then unit01 could issue a network capability to unit02.

3.3 Security Analysis

We now discuss how our architecture relates to the threat model described in Section 3.1. Figure 3.2 illustrates various threats that can be prevented by DIPLOMA.

As the policy tokens are signed by a GC and are verifiable by all nodes, adversaries cannot generate their own valid policy tokens. Similarly, as the network capabilities contain the policy signed by the GC that authorizes a receiver node to issue those capabilities, an

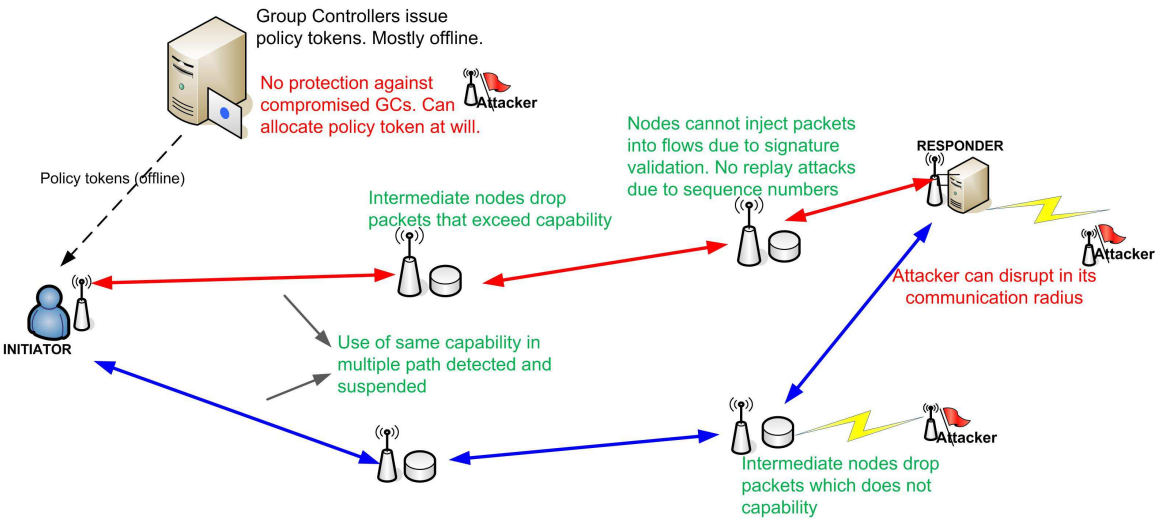

Figure 3.2: Security analysis of DIPLOMA

adversary cannot issue a network capability without compromising the identity of that receiver node. Adversaries can create valid capabilities only if the GC is compromised. As the individual packets are signed, an adversary cannot use a transaction id that does not belong to it for transmitting packets.

A compromised or malicious node that does not enforce the capability protocol can only have impact within its communication radius. Packets generated without a capability or with a snooped transaction id by a malicious node will be dropped by the neighboring nodes due to invalid signatures. A node can only access the services it is authorized to. Hence, damages due to compromising a node are limited to its capabilities. Packets of nodes trying to use more bandwidth than is allocated to them will be rejected. A malicious node frequently doing this can be detected and isolated.

A receiver can protect against DoS attacks by controlling the issuance of network capabilities to its peers. A malicious node may use its policy tokens or network capabilities to send duplicate packets in multiple disjoint paths. This allows a node to transmit more traffic than it is authorized to. The local nodes in the radio perimeter of the misbehaving node can detect this scenario. In Chapter 7, we describe distributed algorithms for detecting and recovering from this type of multiple-path misuses. As the network capability can be

created only based on the policy allowed by the GC, it is not possible for two compromised nodes to collaborate and create arbitrarily large network capabilities.

As the packets are signed, any injection of packets into a data stream is easily detectable by the nodes in the path. With probabilistic verification of the signature at each node, the probability of a malicious packet escaping the checks before it reaches the destination is very low.

3.4 Node Operations

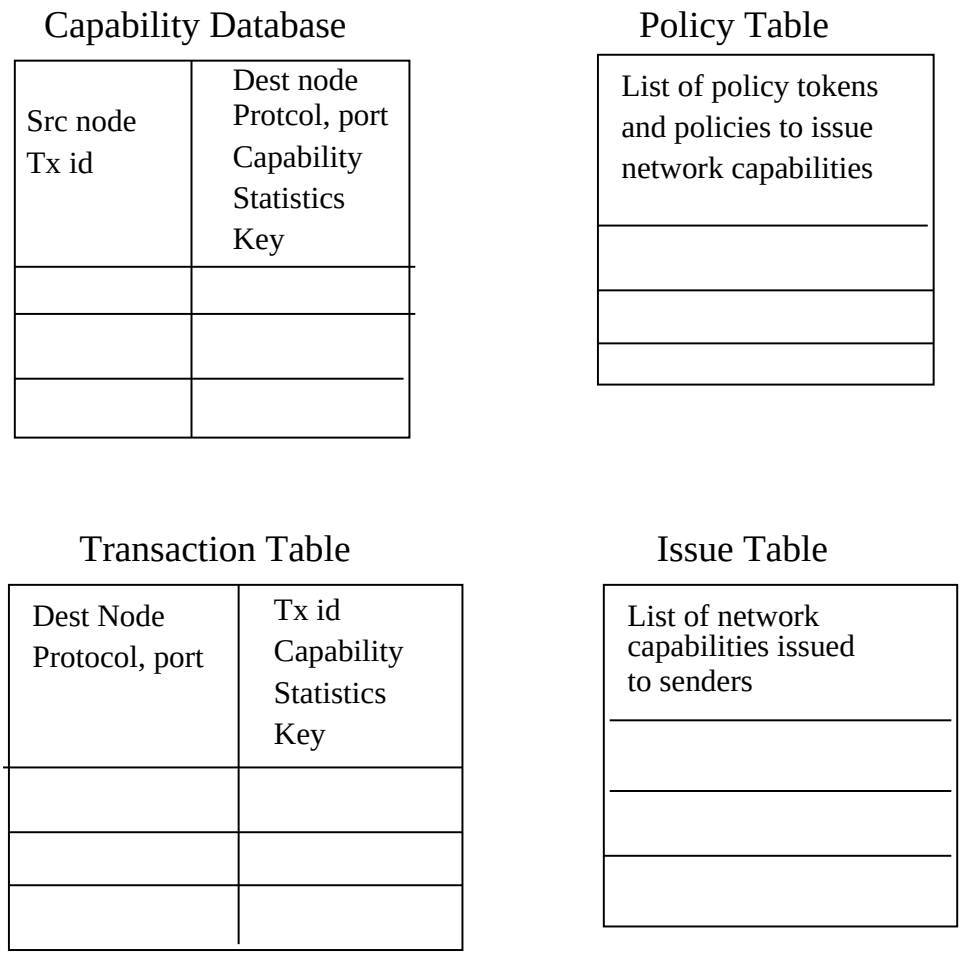

Figure 3.3: Various tables maintained at the nodes

Figure 3.3 shows various tables maintained by the nodes in DIPLOMA. The nodes along the path from a sender to a receiver maintain information about the communication session

in a table called **capability database**. The entries in this table are indexed with the source node address and the transaction identifier. The capability database contains the following information.

- *Source node:* The address of the owner of the capability.
- *Transaction id:* A unique id generated by the source node for this communication.
- *Destination node:* The address of the destination node.
- *Protocol and Port:* The service at the destination node.
- *Capability:* A copy of the credentials associated with this communication.
- *Statistics:* A structure maintained to enforce the bandwidth limitations of the capability.
- *Key:* Public key to verify the signature of the packets.

Each source node also maintains a **transaction table** that maps a traffic session (flow) to the appropriate transaction identifier, its credentials and its usage statistics. Nodes consult this table before sending the packets of a traffic session. Transaction table contains the following information.

- *Destination node:* The address of the destination node.
- *Protocol and port:* The destination protocol and port.
- *Transaction id:* A unique id for using with the packets in the communication session.
- *Capability:* Capability allowing this communication.
- *Statistics:* A structure maintained to enforce the bandwidth limitations of the capability.
- *Key:* Key used for signing the packets.

In addition, a node maintains a list of all its capabilities in its **policy table**. This table contains information on all the services the node has permission to access. When the node initiates a new connection, it retrieves appropriate capability from this table. A node that issues network capabilities to other nodes maintains those issued capabilities in its **issue table**.

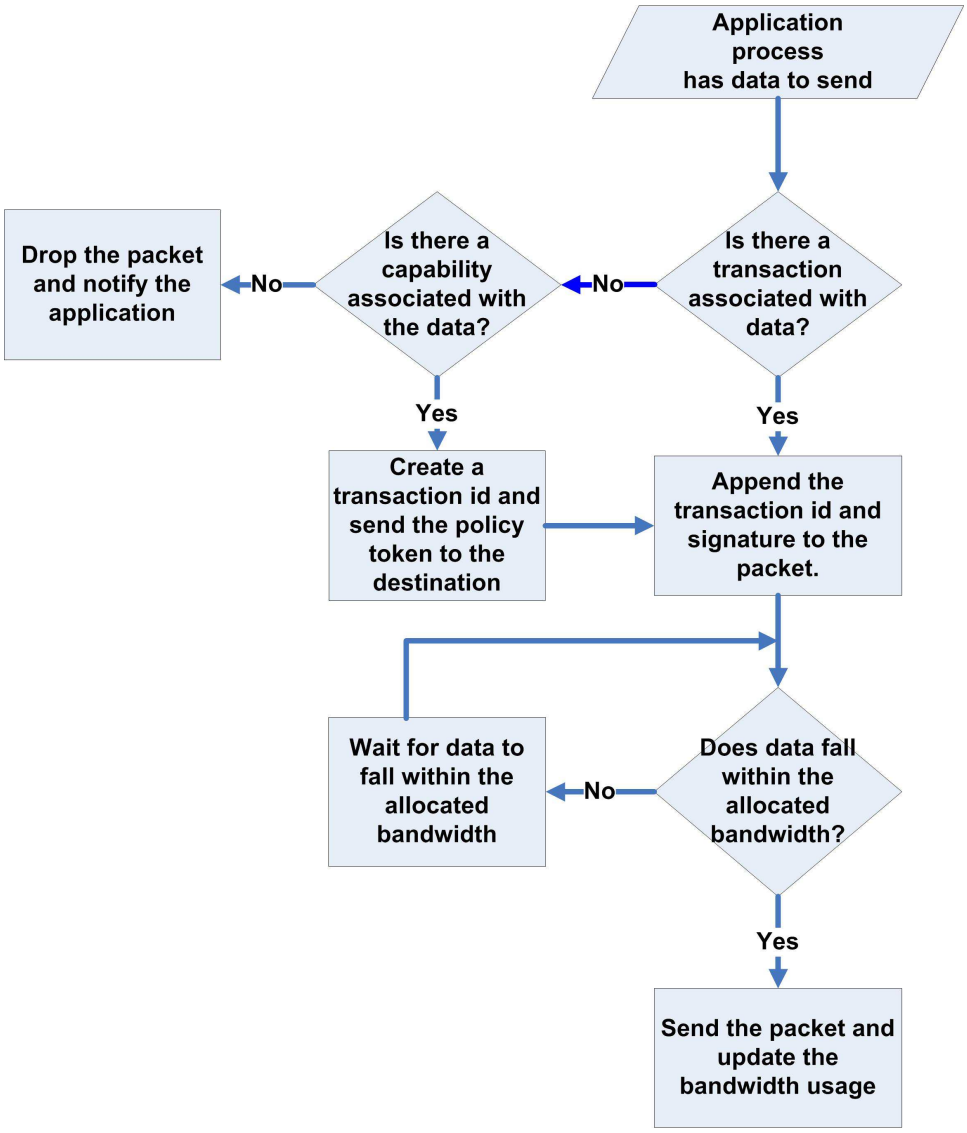

Figure 3.4: Packet processing steps at the sender

Figure 3.4 gives an overview of the processing done at a source node when an application sends data. When an application has data to send, the kernel checks if there is a transaction

id associated with the packet in the transaction table. This will be true if the packet belongs to an existing flow. If the transaction id exists, then the kernel computes the signature using the unique key associated with that transaction id. The kernel inserts the transaction id, a sequence number, and the signature in the packet. Then the kernel verifies that sending the packet does not violate the bandwidth limitations of the flow. If the sending violates the bandwidth constraints, then the kernel queues the packet until the packet can be send without violating the bandwidth limitation. Then it sends the packet and updates the bandwidth usage. Note that if the sender goes ahead by sending a packet that violates the bandwidth usage limitation, then some intermediate node will drop the packet.

If there is no transaction table entry associated with the packet, then the kernel checks whether the node has permission to access the service specified in the packet. This is done by searching for a matching capability in the policy table. If there is no such capability, then the kernel drops the packet. Otherwise, it creates a new transaction identifier for the communication session. It also creates an entry in the transaction table, and a key for signing the packets. Then the node initiates the capability establishment protocol, which will enable it to send the packets. The capability establishment protocol is described in Section 3.5.2.

Figure 3.5 gives an overview of the processing done at an intermediate node when it receives packet for forwarding. If the packet is a capability establishment packet, the node checks the validity of the capability. The validity check includes checks to see whether the capability has a proper signature, whether the sender owns it, whether the capability has a valid time stamp, and whether the sender can use the capability for accessing the service requested at the destination. If the validity check passes, the intermediate node creates an entry in its capability database and forwards the packet. If the validity check fails, the node drops the packet.

If the packet is a data packet, the intermediate node performs a lookup in the capability database for the corresponding entry. This lookup is based on the source address and the transaction identifier of the packet. If there is no corresponding entry, the node drops the packet. Otherwise, the packet is verified against the capability entry. This verification includes whether the packet has a valid signature and sequence number, whether the packet

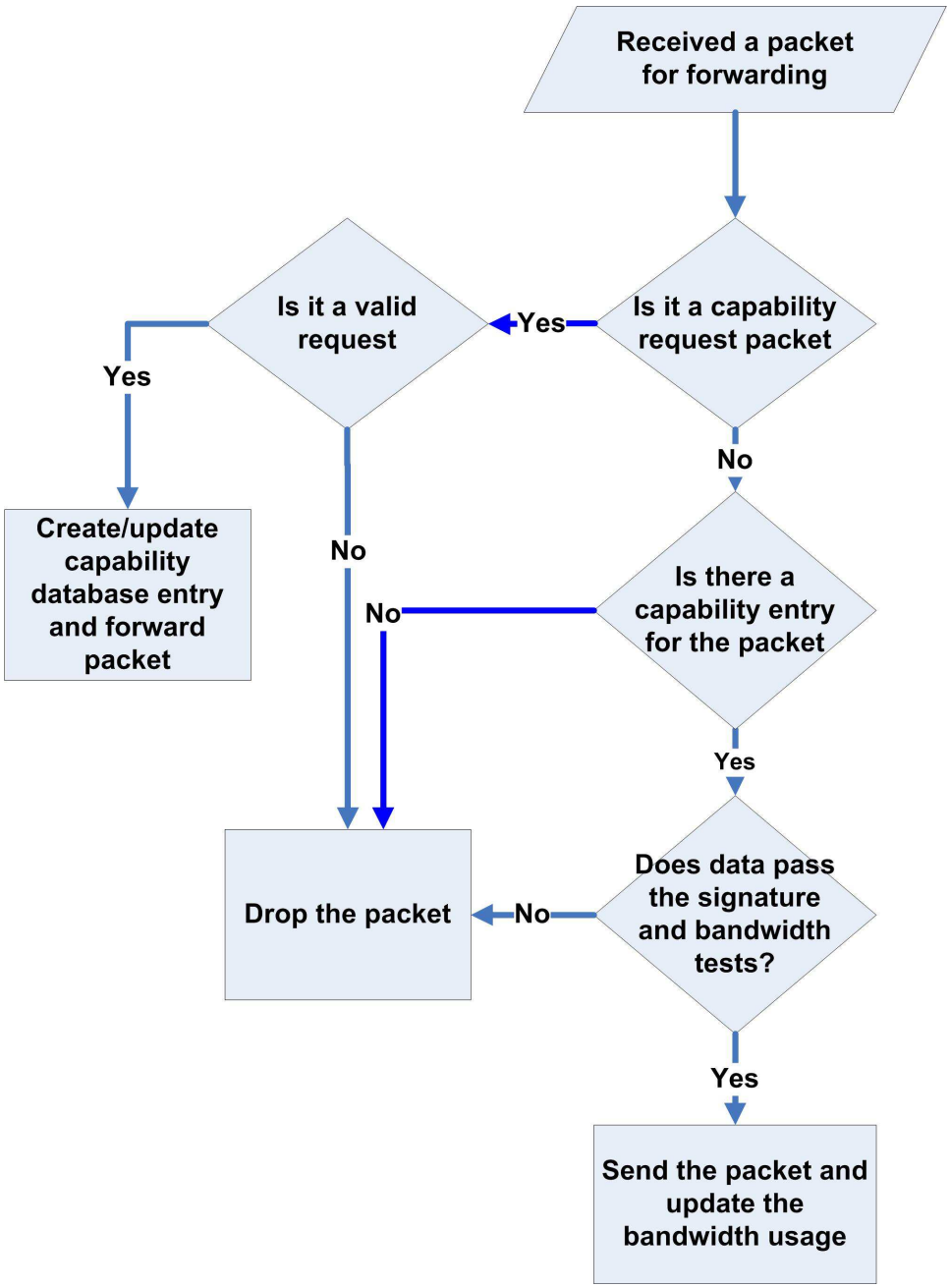

Figure 3.5: Packet processing steps at the intermediate node

is destined for the right destination, and whether the bandwidth constraints of the capability are satisfied. If packet passes all these tests, the node forwards it to the next hop. Otherwise, the node drops the packet.

Figure 3.6 gives an overview of the processing done at the receiver node. If the packet is a capability establishment packet, it checks the validity of the capability. This check is similar to one performed by the intermediate nodes. If the validity check passes, the receiver node creates an entry in its capability database and accepts the packet. The node also initiates a capability establishment in the reverse path for communicating back to the sender, as described in Section 3.5.2. If the validity fails, the node drops the packet.

If the packet is a data packet, the receiver node looks up in the capability database for the corresponding entry, and validates the packet. This check is similar to the one performed at the intermediate node. Then it removes the additional header that the sender had inserted in the packet, and delivers the packet to the right application process.

3.5 Protocol Details

Message Type	Name	Purpose
CAP-REQ	Capability request	Establish the capability along the path from a sender to a receiver
RECV-CAP	Capability response	Send a network capability by a responder to initiator
CAP-ERROR	Error	Send error messages
PUB-KEY	Public key	Send node's public key
CAP-INFO	Capability information	Send capability after a route change
DATA	Data	Send data packets

Table 3.1: Message types in DIPLOMA

In this section, we describe the high-level control protocols used to maintain the state at the nodes in DIPLOMA.

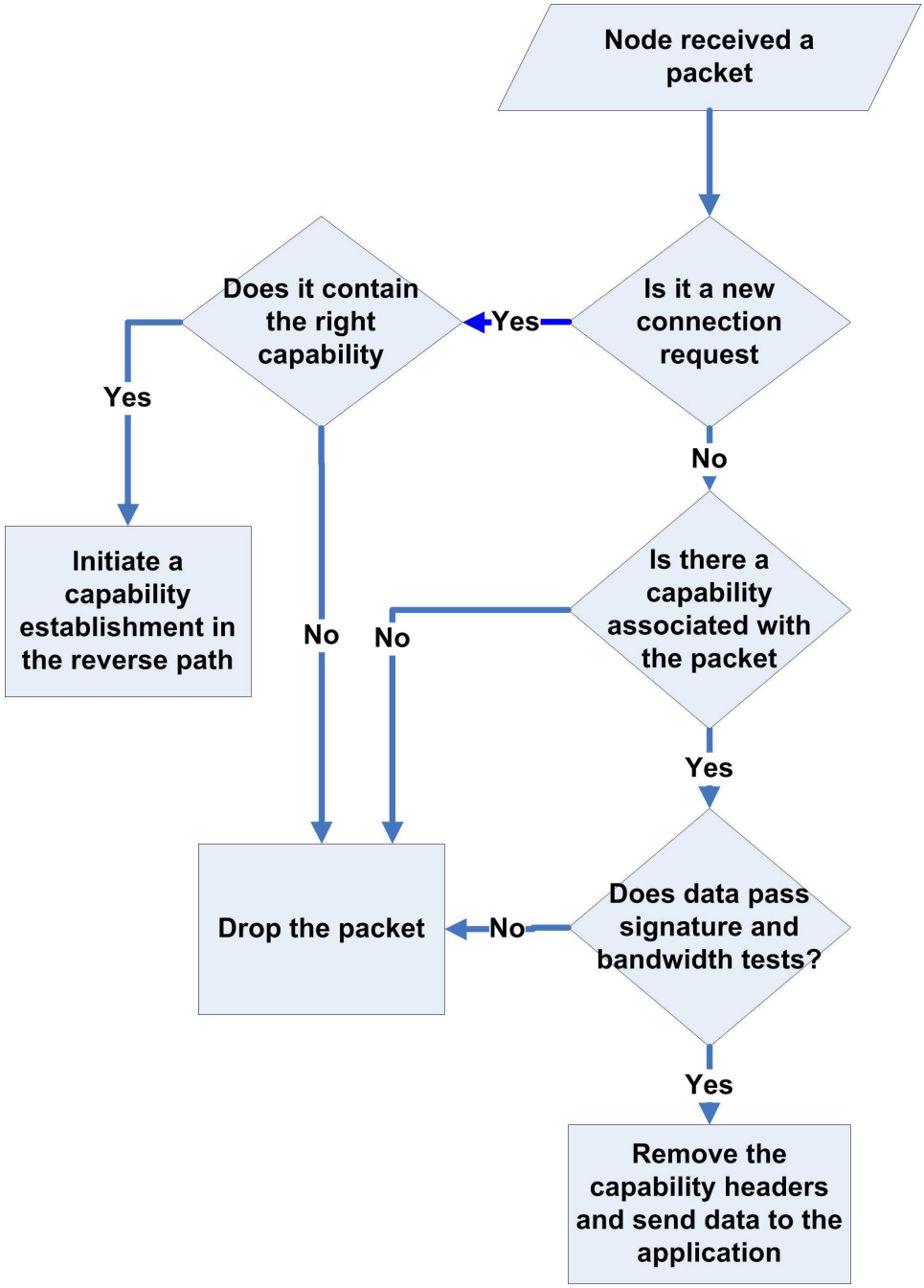

Figure 3.6: Packet processing steps at the receiver

3.5.1 Control Packets

In DIPLOMA, control packets are used to transmit the capability and flow informations, and for error reporting. Control packets may also contain application data. A control packet can have multiple messages embedded in it using type, length, value (TLV) field encoding. Table 3.1 lists all the message types used in DIPLOMA.

The CAP-REQ message is used for establishing an entry in the capability database of the nodes along the path from a source to a destination. The message contains the source node address (ID_i), the transaction id (TX_i), the destination node address (ID_r), and one or more credentials (C). For the first packet in a communication, C will contain the appropriate policy token. This message is signed with the private key of the sender. The message also contains a smaller public key that will be used to sign the subsequent packets. Smaller keys are used to reduce the signature size and the processing time. Smaller keys are sufficient for signing the individual data packets, As the purpose of the packet signatures are to prevent rogue nodes from denying the bandwidth allocated to legitimate nodes by sending packets with spoofed source address. This key is valid for a very short duration (a few minutes), and is changed frequently in long traffic sessions.

The RECV-CAP message is used by a receiver (responder) for sending a network capability back to the sender (initiator). There is no special processing of this kind of messages at the intermediate nodes. When the initiator receives this message, it sends that network capability in another CAP-REQ message toward the responder in the reverse path. On receiving the new CAP-REQ message, the intermediate node will add the network capability in their capability database.

The CAP-ERROR message is used for sending various error messages. The value in the message contains the error type. A node updates its long-term public key to the intermediate nodes using PUB-KEY messages, which may be piggybacked on a CAP-REQ or RECV-CAP message. A DATA message is used for piggybacking the data packets in the control messages. CAP-INFO messages are used for retrieving credential information after a route change.

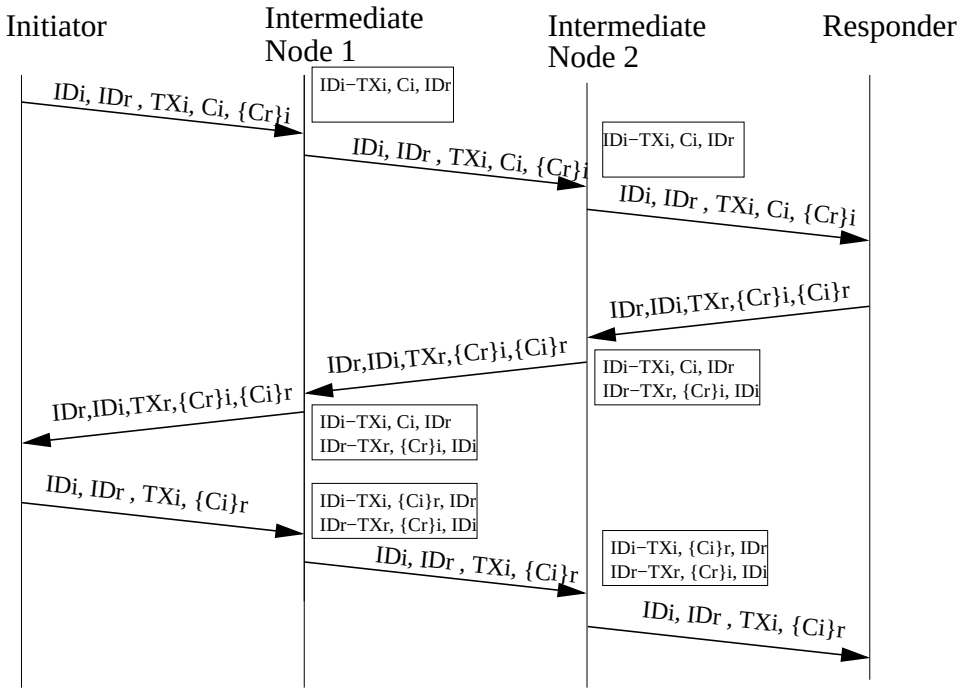

Figure 3.7: Connection establishment

3.5.2 Connection Establishment

When an initiator node wants to enter a communication session with a responder node, the initiator creates a transaction identifier, associates a capability with the transaction identifier, and adds it to the transaction table. Then the initiator sends the following control packet to the responder:

CAP-REQ[ID_i, ID_r, TX_i, C_i], RECV-CAP[$\{C_r\}_i$]

Here ID_i is the address of the initiator, ID_r is the address of the responder, TX_i is the transaction identifier, C_i is the policy token of the initiator, $\{C_r\}_i$ is the network capability assigned by the initiator to the responder.

This packet consists of two message types. One is a CAP-REQ message for establishing the initial credential entry at the intermediate nodes. The optional second message type is the network capability (RECV-CAP) for the responder to communicate back with the initiator. The control packet may also contain a PUB-KEY message if the initiator public key is not known to the nodes, and may carry some application data using the DATA message type.

The packet is send to the next neighbor (intermediate node) using the underlying MANET routing protocol. Each intermediate node verifies the capability C_i and adds it to the node's capability database. This entry in the capability database indicates that packets received from the initiator with transaction id TX_i are to be validated with credentials C_i and they are destined for ID_r.

The responder node, on receiving the connection request, creates a transaction identifier for the communication and associates it with the received network capability from ID_i, or with a capability the node may already have (from the GC). The node also creates a network capability $\{C_i\}_r$ for the initiator, if the initiator has requested one. Consider the general case where the initiator has sent a network capability for the responder, and also has requested a network capability from the responder. The responder sends the following control packet to the initiator:

CAP-REQ[ID_r, ID_i, TX_r, $\{C_r\}_i$], RECV-CAP[$\{C_i\}_r$].

Like the first packet, this packet may also contain the DATA message type and a PUB-KEY message. Each intermediate node (not necessarily the same as in the forward path) creates an entry in its capability database based on this CAP-REQ message.

On receiving the response packet, the initiator forwards the following packet to the responder:

CAP-REQ[ID_i, ID_r, TX_i, $\{C_i\}_r$]

The intermediate nodes update the capability database with the new credential $\{C_i\}_r$. Note that this scheme is unaffected by path changes between the two CAP-REQ messages, as the new intermediate nodes will simply create a new entry in their capability database; the intermediate nodes with an entry from the first CAP-REQ message that have moved will simply expire the said entry after a short period of inactivity on that transaction id.

The connection initiation sequence is shown in Figure 3.7. For ease of representation, message types are omitted from the figure. The boxes in the figure show the contents of the capability database.

3.5.3 Data Transfer

Figure 3.8 illustrates the data transfer. The data packets are of following format:

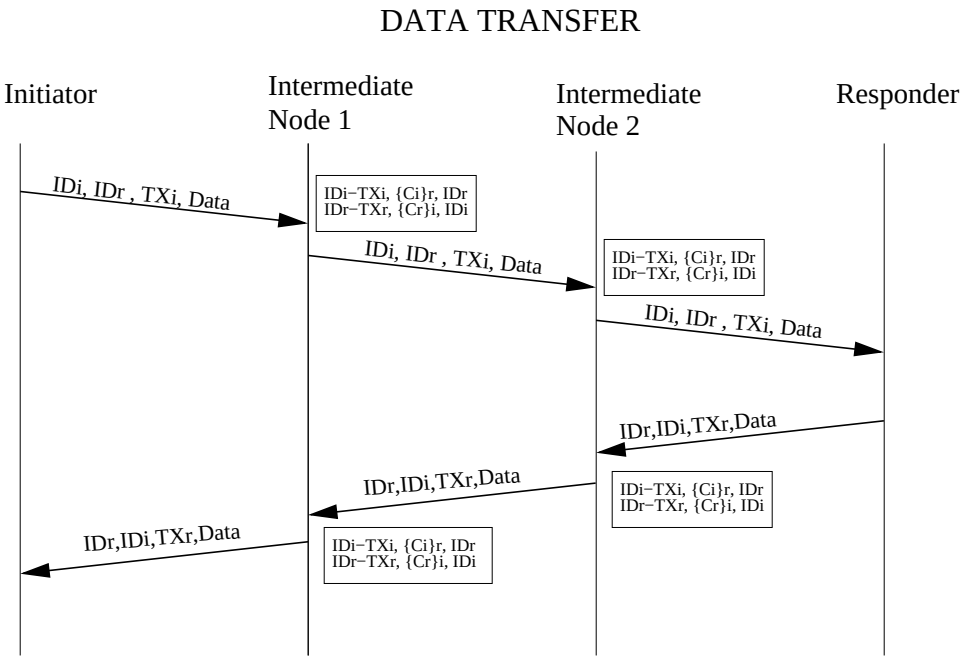

Figure 3.8: Data transfer

$$ID_i, ID_r, TX_i, < data >, < signature >$$

An intermediate node verifies a packet against the associated credentials before forwarding it. It also verifies the signature and sequence number to prevent any spoofing and replay attacks.

3.5.4 Capability Refresh

Periodically, or when the capability is about to expire, the end nodes (both the initiator and the responder) create new network capabilities and send them to each other. The nodes forward the new credentials along the forward path, so that the intermediate nodes update their capability database. Capability refresh packets are also used for updating the public keys used for signing the data packets if the communication session lasts for long duration.

Capability refresh is illustrated in Figure 3.9. Here the initiator generates a new network capability $\{C_r'\}_i$ and sends it to the responder. The responder forwards the new capability along the path from the responder to the initiator. The intermediate nodes update the entry for ID_r, TX_r in the capability database to

$ID_r, TX_r \rightarrow ID_i, \{C_r'\}_i$

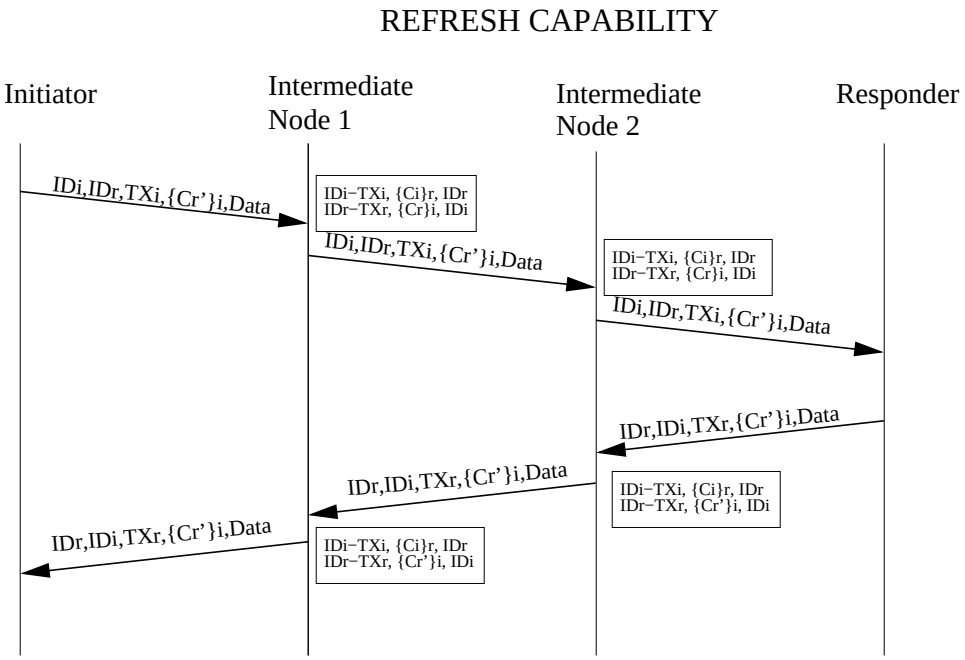

Figure 3.9: Capability refresh

If the end node does not receive any renewed network capabilities, they can still communicate using the policy tokens they possess.

3.5.5 Dealing with Route Changes

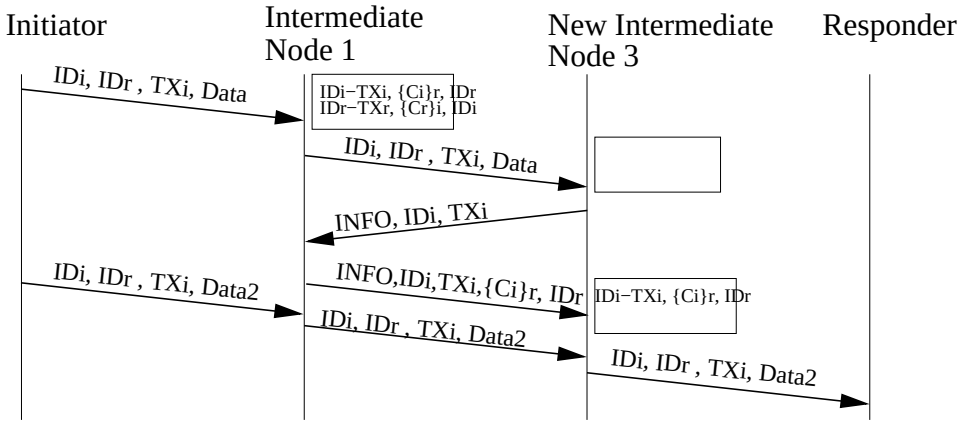

Figure 3.10: Route change

The route between two nodes in a MANET can change frequently because of node mobility. The new route may contain new intermediate nodes that were not part of the

initial connection setup. After a route change, a new intermediate node may receive data packets for which there is no capability database entry. We need a mechanism to re-create this soft state at the new node. In this section, we describe how DIPLOMA deals with this issue.

When a node receives a packet for which it does not have a capability entry, then it checks for the possible causes for the discrepancy. The packet could have come from a malicious node, or could have come due to a route change. In the later case, the intermediate node requests the capability entry from its upstream node. The entry needs to be (cryptographically) verifiable by the intermediate node to avoid spoofing attacks.

The protocol is illustrated in Figure 3.10. A new intermediate node 3 is added to the path from the initiator to the responder. When the data packet $ID_i, ID_r, TX_i, < data >$ is received by node 3, it searches for ID_i, TX_i in its capability database. As that entry is not present in its database, node 3 drops the packet and sends a CAP-INFO message requesting the information about the capability associated with that transaction id to the upstream node, from which it received the packet. If the upstream node has the information in its capability database, it forwards the corresponding entry to node 3 using a CAP-INFO response message. Node 3 verifies and installs the credential(s) in its capability database. Hence, the effect of a routing change is localized only to the neighborhood of the change and does not affect the whole route. Future communication using that transaction id can continue uninterrupted.

Note that the upstream node must have had that information in its capability database, otherwise it would not (should not) have forwarded the packet. If multiple intermediate nodes change in the path, then a packet transmission will cause a cascade of localized CAP-INFO exchanges between successive intermediate nodes. As an optimization, an intermediate node that created a new capability database entry through a CAP-INFO message may send a CAP-INFO message to the next node in the path (to which the data packet will also be forwarded). The recipient of such an unsolicited CAP-INFO message will hold onto it for a short period of time, and use it instead of a CAP-INFO exchange if it receives a packet that would have triggered such an exchange; if no such packet is received (*e.g.,* because the routing has changed again), the CAP-INFO message is discarded. Only a small

number of such messages may be kept at any given time, to minimize the possibility of a memory-exhaustion DoS attack.

3.6 Packet signature optimization

As MANETs do not have dedicated nodes for routing, it is easy for rogue nodes to inject packets into an existing flow. Hence, we require an integrity check for the packets to save bandwidth resources and to avoid unauthorized packets reaching destination hosts. To integrity protect the packets, we require an asymmetric scheme where the sender is able to perform the signature operation that the intermediate node can only verify, but not sign.

We use RSA signatures with small keys (256 or 512 bits) to sign the packets. Although the RSA key sizes used are small, signing individual packets is still a bottleneck if the sender transmits many packets. We implement a combination of the packet hash and the signature to overcome this bottleneck.

When a sender has to send large number of packets, say P packets, then it computes the RSA signatures only for the first packet. All the remaining $P-1$ packets only contain their hashes. The first packet's header also contains these $P-1$ packet hashes. The RSA signature for the first packet is computed on its data and these hashes. In this scheme, P is called the **block size (P)**. The first packet is marked as of type *DATA-FIRST* and the remaining $P-1$ packets are marked of type *DATA-NEXT*. The sender always sends the DATA-FIRST packet before any of the DATA-NEXT packets in the block.

When an intermediate node receives a DATA-FIRST packet, it verifies the packet against its capability for bandwidth usage and signature. It also retrieves the hashes of the subsequent DATA-NEXT packets from this packet, and saves in its memory. The packet is forwarded as any other packet, without waiting for the subsequent DATA-NEXT packets. When the intermediate node receives a DATA-NEXT packet, its hash is compared with the hashes stored for that flow. The packet is accepted for forwarding only if there is a match for the hash, and the packet satisfies the bandwidth constraints.

If the number of packets the source wants to send is less than P, then the DIPLOMA engine should not wait indefinitely for the packet block to fill before sending the first packet.

To handle this case, the engine waits only for a certain time period called **block timeout (T)**, before it sends out the first packet. T is typically a few tens of milliseconds. When the timer expires, the engine uses the available packets to form a block.

The parameter T is useful for dynamic content and for the last few packets of static content. If the dynamic content generated is small, then all the packets that are generated in T time are send as a block. When the dynamic content is large, the block may get filled before the block timeout. In that case, the packets are send as soon as the block is filled.

3.6.1 Priority for DATA-FIRST packets

As each node implements the token-bucket algorithm independently, in a distributed manner, it is possible that the available tokens in the nodes are not synchronized. Different packet processing and the propagation delays can also cause this issue. When a sender sends at the rate of its allocated capacity, a few packets may not have enough available tokens at some intermediate nodes. If the packet to be dropped is a DATA-FIRST packet, then all the subsequent DATA-NEXT packets in the block will also get dropped due to hash verification failure. To overcome this problem, our implementation accepts a DATA-FIRST packet even when there is not enough available tokens. This will make the available token negative. To prevent any misuse, all the subsequent packets, including DATA-FIRST packets, are dropped until the available token becomes positive again.

3.6.2 Block size and block timeout tradeoffs

There are tradeoffs between choosing a large block size vs. a small block size. The extra capability header used for the DATA-FIRST packet increases by 20 bytes for every increase in the block size. If one wants to remove the fragmentation processing for TCP, by setting the maximum segment size (MSS), then the MSS reduces by 20 bytes for every increase in the block size. This reduces the amount of data bytes per packet, increasing the percentage overhead of the headers in the packet. Another issue with the large block size is that the loss of a DATA-FIRST packet will render the system to drop all the DATA-NEXT packets in that block. Thus, a larger block size may cause higher packet loss. The processing cost of sending a packet, on a block size P, can be divided into three components:

p_s - signature computation cost

p_h - hash computation cost

p_o - other processing costs including protocol processing, scheduling of engine *etc.*

Per packet processing cost $= p_o + p_h + \frac{p_s}{P}$.

The advantage of using larger block size is lower processing at the sender. This is because there is only one RSA signature operation per block, which is the predominant transmission cost. If the sender has many packets to send, then a larger block size helps it achieve higher send bandwidth. Larger block size also reduces the packet forwarding cost, as the RSA signature verification needs to be done only one per block. This reduction in processing, however, is not significant, as signature verification requires orders of magnitude less processing than signature generation for RSA.

There is a similar tradeoff for the block timeout. Recall that the DIPLOMA engine waits the minimum of block timeout and the time to fill the packet block, before it sends the packet block. If the timeout is small, then the engine will end up sending partially filled blocks. If the timeout is large, then the packet block likely will get filled before the timeout, reducing the processing time per packet at the sender. The disadvantage of large block timeout is larger packet latency. If there are not enough packets to send, then the latency of the DATA-FIRST packets will be at least the block timeout. Large block size and block timeout tend to send the packets as bursts; this may reduce the wireless link utilization. We study these tradeoffs in experimental evaluation in Section 5.3.

3.7 Cryptographic Algorithms in DIPLOMA

One of the design choices we had to make was the cryptographic algorithm for packet signatures and the key size for the algorithm. Recall that DIPLOMA includes the signatures for the packets. For these signatures, we use RSA keys with key sizes smaller than the full public key. In this section, we discuss the rationale for this choice.

The per-packet signature is needed to guarantee that a packet being forwarded has originated from the node in its source address field, in order to perform the bandwidth accounting. The packet signatures help to prevent any attack in which a rogue node steals

Duration	Symmetric	RSA	ECC
Days/hours	50	512	100
5 years	73	1024	146
10-20 years	103	2048	206
30-50 years	141	4096	282

Table 3.2: Key lengths for confidentiality. (Source: *http://www.ecrypt.eu.org* and *http://homes.esat.kuleuven.be/preneel/preneel_securecom09.pdf*)

the bandwidth allocated to other nodes' communication sessions. Without the packet signatures, an attacker in the proximity of any node in the path from a source to a destination can inject packets into a communication session. The packets also contain sequence numbers to prevent replay attacks.

The signature algorithm used has to be an asymmetric algorithm, so that the intermediate nodes can verify the signature, but cannot alter the packet signed by the sender. If we were to use a symmetric algorithm, then the keys had to be shared with the intermediate nodes. This enables the intermediate nodes to steal the bandwidth of the communication session by modifying or injecting the packets into a communication session.

For performance reasons, we used smaller keys for packet signatures, compared to the full public key. Use of larger keys poses two performance issues. It requires more processing time to sign packets at the senders and to verify the packets at the intermediate nodes. A larger key size also leads to larger signatures, leading to higher per-packet overhead. In DIPLOMA, signature keys are useful only during the active lifetime of the keys, which is the duration for which sender uses that key for signing packets. Once all the packets signed using a key reach the destination, the key is no longer useful for the attacker. DIPLOMA also refreshes the keys every few minutes, by sending the capability refresh messages. As the lifetime of the keys is short, smaller keys are sufficient for packet signatures. Table 3.2 gives the time required to break RSA keys for different key sizes [Pre09; Ecr09]. It takes couple of hours to break a 512 bit keys. As the security requirements for packet signatures are weak (couple of minutes), we use 256 and 512 bit keys in our experiments. Systems

	Block Cipher	RSA	Elliptic Curve	DSA
Export grade	56	512	112	512/112
Traditional recommendations	80	1024	160	1024/160
	112	2048	224	2048/224
Lenstra/Verheul 2000	70	952	132	952/125
Lenstra/Verheul 2010	78	1369	146/160	1369/138

Table 3.3: Minimal key lengths in bits for different grades. (Source: *http://www.rsa.com/rsalabs/node.asp?id=2264*)

with stronger security requirements may benefit from larger keys, at the expense of larger packet overhead.

3.7.1 Choice of the algorithm

We considered three available public key cryptographic algorithms: RSA [RSA78], DSA [NIS94] and Elliptic Curve DSA (ECDSA) [Sim92; IEE97]. Table 3.3 gives the key size comparison for all the three algorithms [RSA10]. The RSA key size refers to the size of the modulus. The Elliptic Curve key size refers to the minimum order of the base point on the elliptic curve; this order should be slightly smaller than the field size. The DSA key sizes refer to the size of the modulus and the minimum size of a large subgroup, respectively (the size of the subgroup is often considerably larger in applications).

Even though RSA signatures are larger, which leads to larger per packet overhead in terms of the header, we use RSA for the per packet signatures due to the asymmetry in the signature generation and verification time. Table 3.4 shows the signature and verification time for these algorithms for various key sizes. These numbers were obtained by running *openssl* [ope] on an Orbit lab machine.

As seen from the table, RSA signature verification time is considerably lower than that of the DSA and ECDSA. In DIPLOMA, the signature verification is part of the packet forwarding operation. To optimize the packet forwarding operation, as opposed to packet generation operation, we use RSA algorithm in DIPLOMA. If packet forwarding were to

Algorithm	Key Size	Sign (ms)	Verify (ms)	Sign/sec	Verify/sec
rsa	512	3.522	0.260	283.9	3851.7
	1024	15.898	0.718	62.9	1392.1
	2048	87.478	2.302	11.4	434.4
dsa	512	2.649	3.089	377.5	323.7
	1024	7.109	8.439	140.7	118.5
	2048	22.573	27.027	44.3	37.0
ecdsa	160	1.9	9.0	537.4	110.9
	192	1.8	8.8	553.5	114.0
	224	2.3	11.3	437.0	88.7

Table 3.4: Performance of signature and verification operations for different algorithms

require lot of processing, an attacker could launch denial of service attacks by injecting fake packets into the communication flows. Then the intermediate nodes would have to spend bulk of its processing power in detecting and filtering out the attack packets. In addition, the total number of signature verifications in a MANET using DIPLOMA is more than the number of signing operations. This is because, a signature is generated only once per packet at the sender, but, it is verified at each hop of the packet takes, including the receiver.

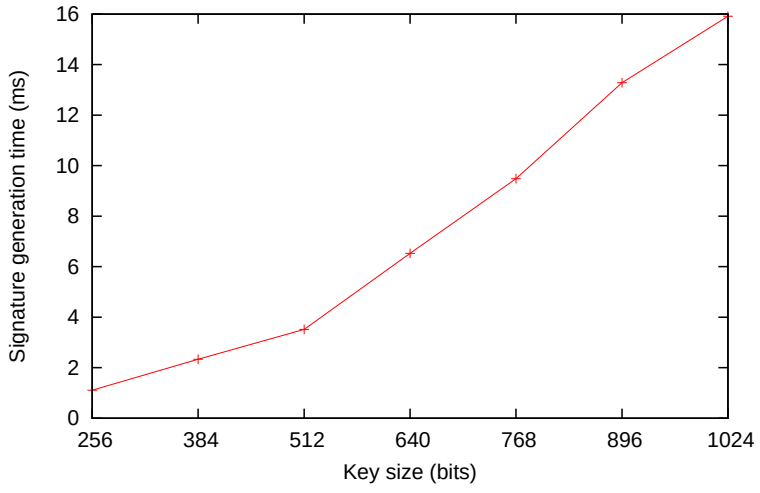

Figure 3.11: Processing time required to sign messages using RSA

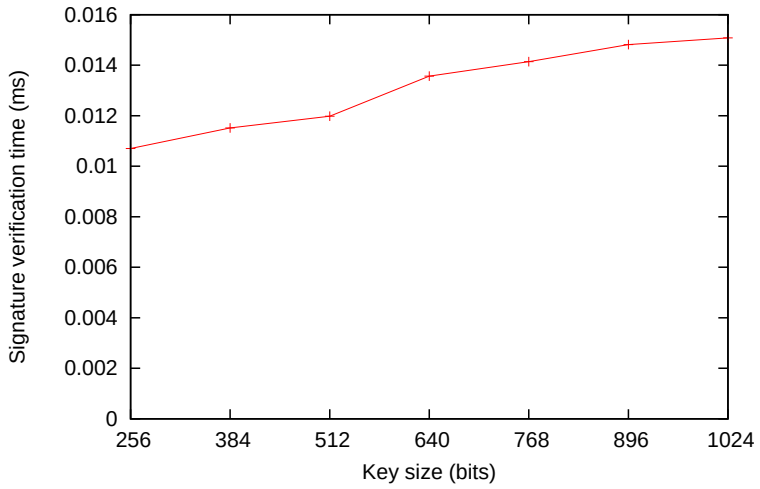

Figure 3.12: Processing time required to verify messages using RSA

3.7.2 Key size for the packet signatures

Figures 3.11 and 3.12 show the processing time required to sign and verify the messages using RSA for various key lengths. The plots were generated by signing and verifying the messages using the openssl library routines in the Orbit lab machines. In both the plots, there is a close to linear increase in the processing needed as the key size increases. The slope of the curve is significantly higher for the signature operation than the verification operation. Hence, if the nodes were to use larger key sizes for the packet signatures for better security, then most of the overhead will be borne by the sender. The additional processing needed by the intermediate nodes for the signature verification while forwarding the packets that use larger key sizes is minimal. The signature time at the sender increases by a factor of 2.18 and 3.52 respectively when moving from 256 bits to 512 bits and from 512 bits to 1024 bits. For the same cases, the signature verification at the forwarding nodes increases only by a factor of 0.12 and 0.26 respectively.

Chapter 4

Simulation Evaluation

In the last chapter, we introduced DIPLOMA, a deny-by-default architecture that enforces trust relationships and traffic accountability between mobile nodes through a distributed policy enforcement scheme for MANETs.

In this chapter, we provide a preliminary evaluation of such a deny-by-default system using the GloMoSim [glo] simulator. Because GloMoSim does not include any packet checking functionality, we added another layer between IP and the AODV routing processing, where we implemented our protocols. Our primary concern in the evaluation is the network overhead of DIPLOMA given the cryptographic operations required. Therefore, we focused our measurements on comparing the packet latency and bandwidth with and without our system in a variety of mobility scenarios and topologies. We show that the collaborative effort of enforcing the policies provides strong security benefits without incurring much performance overhead. We discovered that DIPLOMA imposes an 8% overhead on the end-to-end latency and a 5% reduction on available bandwidth. We believe that this is not a high price to pay given that there are scenarios where a MANET becomes completely unusable even when a single node misbehaves. We also show that our system allocates the network resources in a fair manner, even in the presence of misbehaving nodes.

Rest of the chapter is organized as follows. We briefly present the GloMoSim simulator and our modifications in Section 4.1. We evaluate our architecture through simulation, with results given in Section 4.2.

4.1 Implementation on GloMoSim

Global Mobile Information System Simulator (GloMoSim) is network protocol simulation software that simulates wireless and wired network systems [glo; XZ98]. It is a discrete event simulator, designed using the parallel programming language Parsec [par]. Parsec is a C-based simulation language, developed by the Parallel Computing Laboratory at UCLA, for sequential and parallel execution of discrete-event simulation models. In GloMoSim, the communication protocol stack for wireless networks is divided into a set of layers, each with its own API. Models of protocols at one layer interact with those at a lower (or higher) layer only via these APIs. Figure 4.1 shows the layering architecture of GloMoSim.

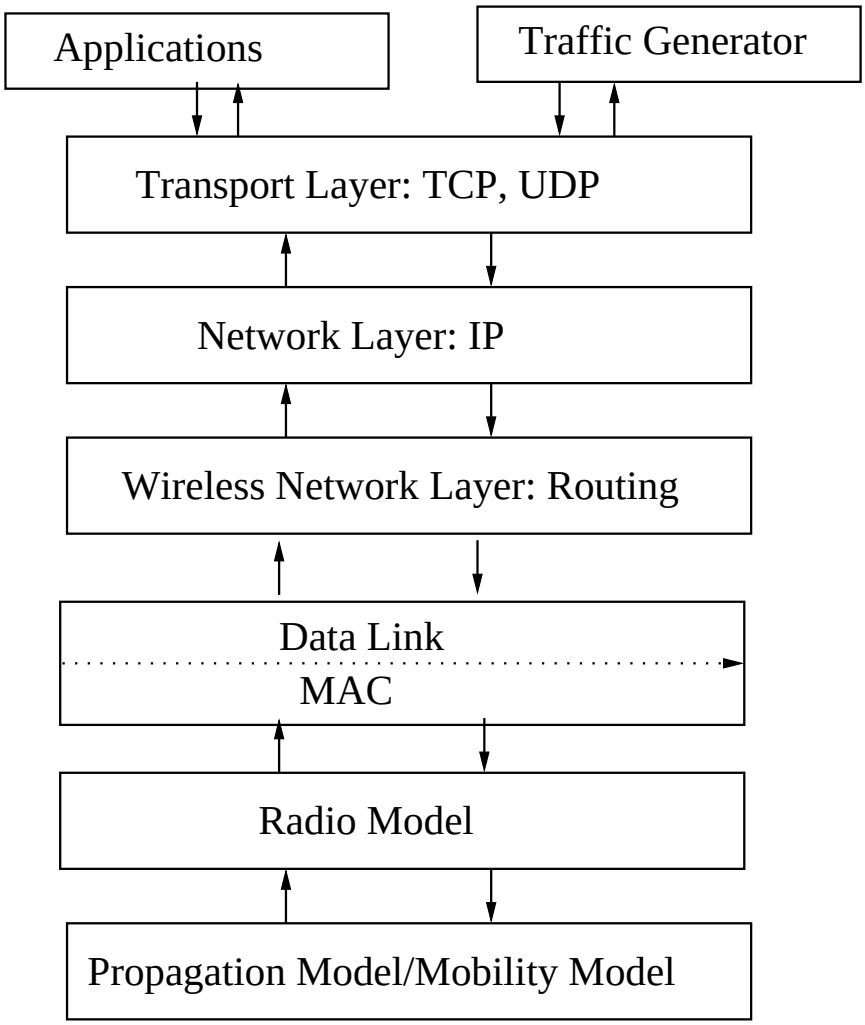

Figure 4.1: GloMoSim Architecture

We implemented the DIPLOMA protocols as a routing layer and replaced the existing wireless routing layer of GloMoSim with the new layer. This DIPLOMA layer processes protocol packets, establishes the capability database entries, and enforces the capability for the data packets as described in Section 3.5. Then the DIPLOMA layer invokes the routing protocol APIs, which is mapped to the selected routing protocol (AODV, ODMRP, etc).

GloMoSim did not have support for including protocol processing delays for packets. Therefore, we added the support for packet processing delays by introducing timers for the packets in the DIPLOMA layer. We implemented bandwidth enforcement using a token bucket algorithm [Tan96].

4.2 Simulation Results

Here, we compare the performance of DIPLOMA (referred to as *dip*) with a system that does not use capabilities (referred to as *original*). Note that *original* is inherently vulnerable to a number of attacks, including DoS and unauthorized access, which are not feasible in DIPLOMA. However, our experiments are aimed at quantifying the performance impact of using DIPLOMA, relative to an unsecure MANET.

We conduct a number of experiments, of increasing complexity in terms of topology and MANET parameters, in order to build up our understanding of the system behavior. Initially, we use a simple "line" topology, where seven nodes (numbered 0 through 6) are arranged in a line 200 meters apart. We use this simple topology for computing the basic overhead of DIPLOMA, as it is easy to analyze the results. Then, we measure our system using more complex topologies like grid topology and random topologies with node mobility.

In our experiments, we keep the default radio parameters of GloMoSim: radio range 376.782m and link bandwidth 2 Mbps. We use 802.11 as the MAC protocol. We introduce a packet processing delay in both models. This is set to 0.01 milliseconds. This is the time required to process 128-byte packets on a 100 Mbps link. To protect the integrity of the capability tokens and verify the identity of the sender and the receiver, we employ 256-bit RSA for signing the individual data packets and 1024-bit RSA for signing the capability itself. 256-bits are sufficient for very-short-lifetime data packet signatures as we change

this key periodically (Chapter 3). Headers related to DIPLOMA introduce an additional 36 bytes per data packet; 2 bytes for the transaction identifier, 2 bytes for the sequence number and 32 bytes for the signature. The intermediate nodes always verify packets containing policy tokens, as they constitute relatively low traffic. However, to improve the latency performance of our system, we chose to verify data packets probabilistically (this can depend on the path length). Upon detection of an unauthorized packet, we can revert back to deterministic packet checking and isolate the misbehaving node. The cost of all packet operations are (per packet): inserting a capability token (identifier) in the capability database costs an average of 0.01 milliseconds and the record lookup operation costs 0.005 milliseconds. In addition, generating a signature requires 0.168 milliseconds, while verification takes 0.0275 milliseconds for data packets. Network capabilities require 3.159ms for signature generation and 0.140ms for signature verification. A capability refresh packet is sent every 8 seconds. Simulations were run on a Pentium-4 3.20GHz CPU with 1GB memory.

Each intermediate node verifies the signature of a packet with probability 0.2063. As this verification decision is taken independently by each node, a signature of a packet is verified by at least one node in a 3-hop path with probability 0.50 (*i.e.,* 1-(1-0.2063)(1-0.2063)(1-0.2063)). With the same packet verification probability, a packet on a 5 hop path is verified by at least one node with the probability 0.685. The performance overhead for a system in which the nodes verified the signature of all the packets was also similar, as the signature verification did not incur high overhead.

We implemented a token bucket algorithm to enforce the bandwidth limitation at the intermediate nodes. This enables us to limit both burst and average rate of the flow. Each of the experiments was run 20 times with different seed values, and the average of the parameter of interest was taken.

We present the results for overhead of DIPLOMA in terms of latency, throughput and resilience to mobility using the line topology in following three subsections. The results for larger topologies including grid and random topologies is presented in Section 4.2.4.

4.2.1 Packet latency

We compare the latency of packet processing in DIPLOMA with that of *original* (also shown as *org* in the figures)). We send 1000 packets of size 512 bytes at 100 ms intervals from a source node to a destination node n hops away, where $n = 1, \ldots 6$ in the line topology. We measure the latency of the packet as the time from the creation of packet in the source node to the time it reaches the final destination.

The latency of the first packet is larger than the rest of the packets. This is because the route needs to be discovered in both schemes (DIPLOMA and *original*), and credentials need to be established in DIPLOMA. The packet processing for DIPLOMA also includes capability database lookup and probabilistic verification of packet signatures.

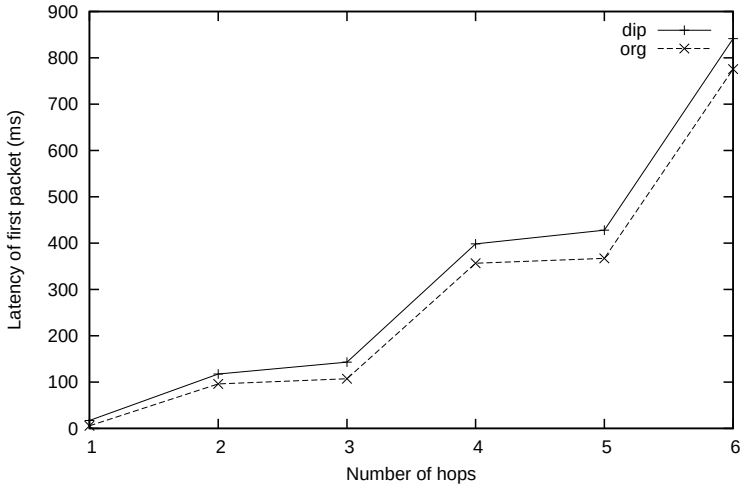

Figure 4.2: Latency of the first CBR packet of size 512 bytes

Figure 4.2 shows the latency for the first packet to reach the different destination nodes. The higher latency in DIPLOMA is due to the credential establishment, capability database lookup and signature verification, as well as the size overhead (36 bytes) in the packet. This average overhead is 35.8ms, 41.6ms and 60.9ms respectively for nodes 3, 4 and 5 hops away. The average overhead is 20.5%. The overhead increases with hop length, as the overhead is added at each node. It can also be seen that the latency increases considerably from 3 hops to 4 hops in both schemes. This is an artifact of using AODV as the underlying routing protocol, because AODV had to increment the TTL once more and retransmit the RREQ

packet while finding the routes to the node that was 4 or 5 hops away. The same is true for 6 nodes.

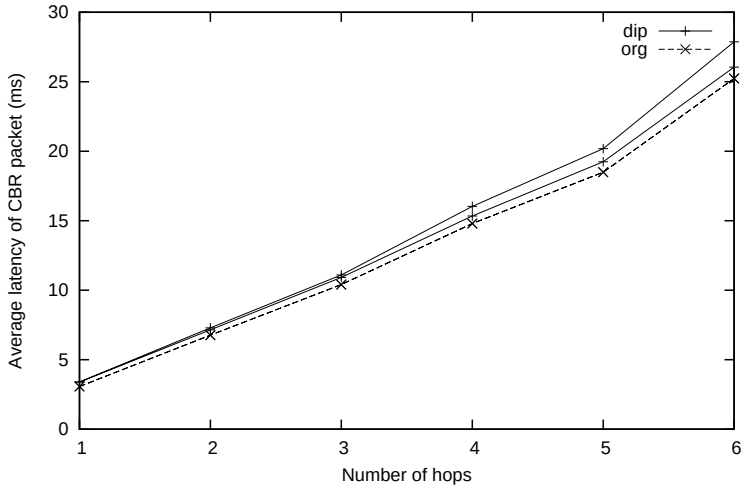

Figure 4.3: Average latency of 1000 CBR packets of size 512 bytes

Figure 4.3 shows the average latency for all 1000 packets to reach their destination node, in each of the different measurements (transmission to nodes n hops away, varying n in each experiment). The effect of the high latency for the first packet is amortized over a large number of packets. The average overhead is only 0.6ms, 1.2ms and 1.6ms respectively for nodes 3, 4 and 5 hops away. The average overhead is only 8% for these paths.

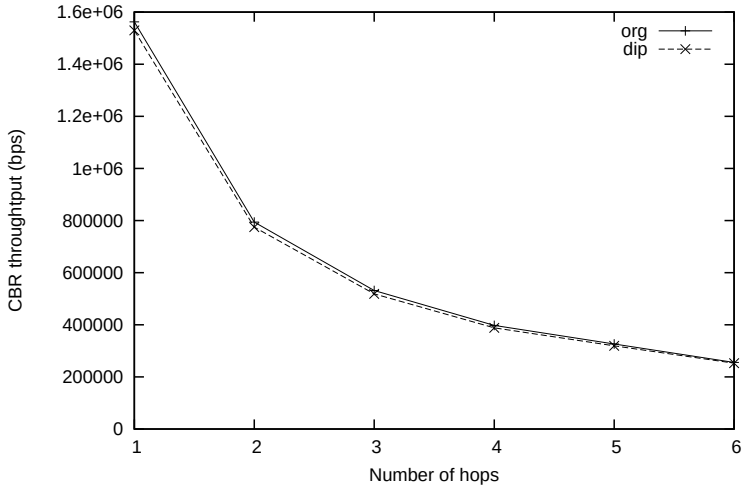

Figure 4.4: CBR throughput

4.2.2 Throughput

UDP throughput

We now compare the throughput of DIPLOMA with that of *original* on an 802.11-based MANET. We use the line topology and pump large packets (1400 bytes) at high rate (every 1ms). We set node 0 as the source of the CBR traffic and send the traffic to destination nodes at different hops. We measure the number of bytes received within one minute from the start of the data transfer and compute the data throughput. The results are shown in 4.4. As expected, the throughput in both schemes decreases as the number of hops in the path increases. The throughput of DIPLOMA is only 2% lower than the original (insecure) scheme.

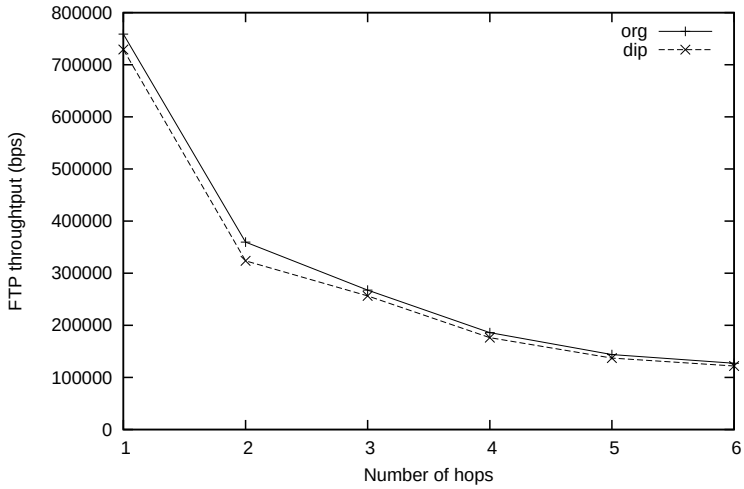

Figure 4.5: FTP throughput

TCP throughput

To measure the performance of TCP in DIPLOMA, we compare the throughput of FTP on both schemes on a line topology. An FTP client at node 0 transfers data to an FTP server at $1, 2, \ldots, 6$ hops away. In each experiment the client sends 10 application-layer items of random sizes. The application layer item sent was the same for both schemes in the same experiment. The results are plotted in Figure 4.5. The behavior of TCP performance is similar to that of CBR, but at lower bandwidth due to TCP congestion control and in-order

guaranteed delivery. On average, TCP throughput for DIPLOMA is 5.3% lower than the original (insecure) scheme.

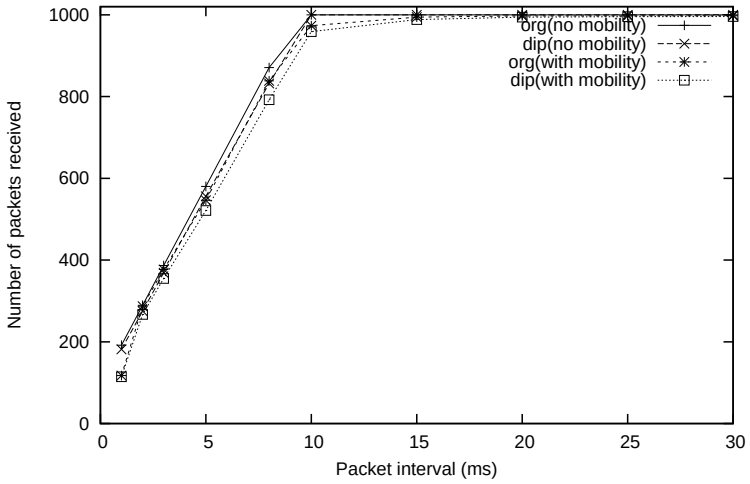

Figure 4.6: Number of packets received after a route change

4.2.3 Resilience to mobility

To verify the validity of our approach in a MANET environment, we evaluate the effects of mobility on the capability scheme. As the nodes keep only soft state about the capabilities, when the route changes due to node mobility, the new node needs to receive the credentials (policy token and network capability) for existing sessions.

Figure 4.6 shows the effect of mobility on the number of packets received for various inter-packet intervals. In this experiment, 1000 packets of 512 bytes were sent at a constant rate to a node at three hops, starting at time 0. At time 0.5 seconds, the node at two hops was removed and a new node introduced. Figure 4.6 shows the number of packets received at the destination for both schemes, with and without this mobility. As expected, the number of packets received decreases at lower inter-packet interval. On average, DIPLOMA drops 155ms worth of traffic, whereas the original scheme drops 108ms worth of traffic. This higher loss is due to the need for propagating the network capability to the new node.

		3-93	6-96	30-39	60-69
FTP	Original	16395	15546	14759	14694
	DIPLOMA	14062	17722	14176	15147
CBR	Original	65711	162293	57535	177303
	DIPLOMA	54113	148027	57793	148153
	Ltd bw capability	129164	131437	129844	134230
CBR Mobility	Original	74124	150718	52510	157779
	DIPLOMA	59864	117728	57933	129347
	Ltd bw DIPLOMA	113111	136975	100924	138040

Table 4.1: FTP and CBR throughput (bps) on a grid topology

4.2.4 Larger topology

Next, we evaluate our system in the context of a larger and more complex topology, and in the presence of mobility.

Grid Topology

We use a grid topology containing 100 nodes (10x10 grid), each node 300m apart. We ran four FTP sessions, two of them from nodes on the top of the grid to nodes on the bottom of the grid; specifically, between node pairs $(3, 93)$ and $(6, 96)$. The other two FTP session were from left to right, between node pairs $(30, 39)$ and $(60, 69)$. We also ran traffic of 1400 bytes with 10ms inter-packet interval for those source-destination pairs and computed throughput.

Table 4.1 shows the average throughput of the four sessions, for both FTP and CBR. The average throughput of DIPLOMA and *original* is comparable. DIPLOMA's throughput is only 0.5% lower for FTP and 11.8% lower for CBR.

The CBR experiment in the table contains 3 rows. The original scheme does not limit the bandwidth a node can use. DIPLOMA in the second row allowed nodes to use unlimited bandwidth. In both cases, two of the sessions get most of the bandwidth. In the third experiment, the network capability permitted limited bandwidth. In this case, each of the

sessions received a fair share of the available bandwidth.

The last set of rows shows the effect of mobility in both schemes. In this set of experiments, the second node in the route from source to destination of all the traffic pairs was removed 2 seconds after the experiment began. The average throughput of DIPLOMA was 16.1% less than the original scheme. This reduction is more than the CBR traffic without mobility. This is because DIPLOMA needs more time to recover, due to the need for restoring the capability database in the new route. The last row shows the results when the capability had a limited bandwidth. Here the average bandwidth dropped by 6.8% compared to DIPLOMA without mobility.

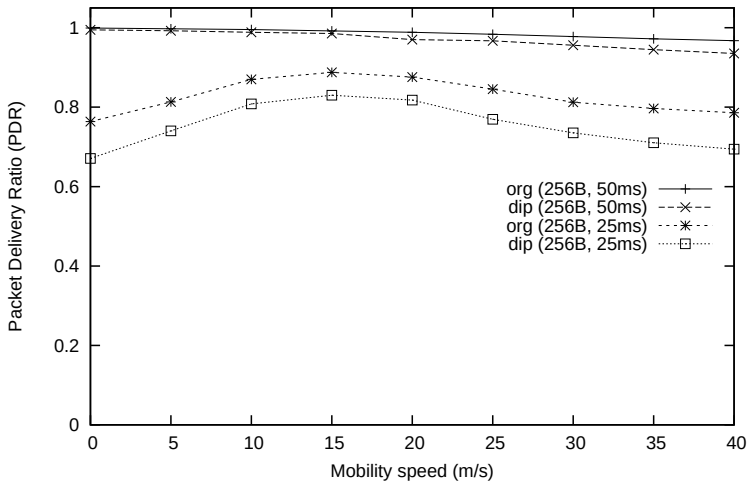

Figure 4.7: Packet delivery ratio for unicast CBR traffic at various mobility speeds

Random Topology

We placed 50 nodes at random on a terrain of 1200×1200 meters. There were five random source-destination pairs that were sending CBR traffic of 256 bytes at a packet interval of either $50ms$ or $25ms$ (*i.e.,* data rate of $40kbps$ and $80kbps$ respectively). In each experiment, all the nodes were mobile with a constant speed using the *Random Waypoint* model. Each of the experiments was conducted 20 times using different seeds and the average was taken. Experiments where the topology was partitioned were discarded, as some of the source-destination pairs may not be able to communicate in such topologies.

Figure 4.7 shows the packet delivery ratio (PDR) for both schemes at various mobility

speeds. On average, the PDR for DIPLOMA was only 1.6% and 9.14% lower than for *original*, for 50ms and 25ms inter-packet interval respectively.

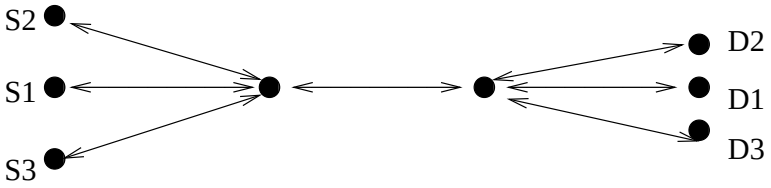

Figure 4.8: Topology to study the misbehaving nodes

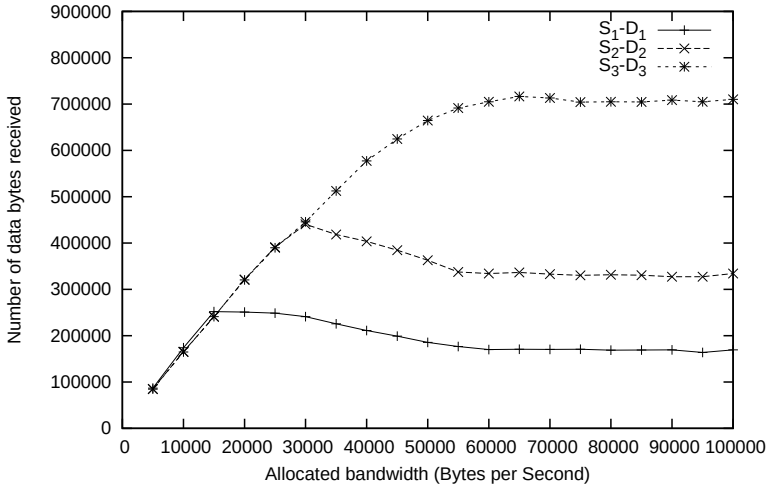

Figure 4.9: Limiting bandwidth of misbehaving nodes

4.2.5 Resilience against misbehaving nodes

Another important characteristic is the system's behavior in the presence of malicious or misbehaving nodes. To that end, we study the attack resilience of our protocol. The topology for this experiment is shown in Figure 4.8. There are three source nodes S_1, S_2 and S_3 sending traffic to respective destination nodes D_1, D_2 and D_3. The traffic is CBR with packets of size 512 bytes, sent at packet intervals 40ms, 20ms and 10ms, originating respectively from S_1, S_2 and S_3. All the nodes are allocated the same bandwidth. Even though all the source nodes have the ability to send to their destination, they try to send more than they are allowed. S_3 misbehaves the most and the node S_1 misbehaves the least. The network tries to limit each flow to its capability.

The results are shown in Figure 4.9. Even though S_3-D_3 traffic is four times the S_1-D_1 traffic and two times the $S_2 - D_2$ traffic, each of them gets the same bandwidth till the allocated bandwidth is less than the rate of $S_1 - D_1$ traffic. Any increase in the allocated bandwidth after reaching the rate of $S_1 - D_1$ gives the same increase for the other two flows. Once the allocated bandwidth reaches the rate of $S_2 - D_2$ traffic, $S_3 - D_3$ bandwidth increases to its full rate. Hence, DIPLOMA is able to allocate bandwidth in a fair manner, even in the presence of misbehaving nodes. We see that the number of bytes received is slightly less than the allocated bandwidth as the IP, UDP, and DIPLOMA headers are excluded from the bytes received.

Chapter 5

Implementation in Linux

In the last two chapters, we have introduced the DIPLOMA architecture and studied the performance of the architecture using simulations. In this chapter, we look at the aspects of implementing DIPLOMA in real systems.

We implement DIPLOMA on Linux systems using a user level protocol engine that interfaces with rest of the packet processing system through *netfilter-queue* APIs. Our implementation does not require any changes to the existing applications. However, the applications see the benefit in terms of receiving only authorized traffic and being able to send the allocated bandwidth even in the presence of rogue nodes that are trying to send large amounts of traffic. Our implementation uses a simple way of representing and enforcing bandwidth constraints using token bucket parameters. In DIPLOMA, we are doing a clean-slate design of the network. In our design, we leave the IP layer intact; hence, we can make use of existing routing and packet forwarding capabilities of the nodes.

For our simulation evaluations given in previous chapter, we did not use the packet signature optimization scheme presented in Section 3.6. During our experiments, we found that per packet RSA signature computation was a barrier to achieving high throughput. Hence, we used the signature optimization in our implementation. We also study various tradeoffs that were presented for the signature optimization through experimental evaluations.

We also conduct extensive experiments for evaluating the performance and the effectiveness of our system. To that end, we implement our system on the Orbit-lab test-bed [orb]. Most of the indoor wireless testbeds create multi-hop topologies using MAC address fil-

tering. A major problem with that approach is that an attacker can get unfair share of the channel, affecting all the other nodes in the topology. We propose a novel solution for creating multi-hop topologies in indoor environments in which an attacker cannot cause unlimited damage to the nodes in its proximity by hogging the channel.

Our experiments show the effectiveness of the new authentication scheme. They also show that the there is minimum overhead on throughput, latency, and jitter for DIPLOMA. Our experiments on attack resiliency show that good nodes and bandwidth of the network can be protected from the attackers. We also show that our system works well in the presence of multiple flows. We also show that existing applications like FTP and *wget* work without any changes and achieve good performance.

The simulation evaluation presented in the previous chapter reported 5% overhead in throughput. Our real system implementation reports about 19.8% to 23% reduction in the throughput. This is because the simulator is a discrete event simulator that deals with only certain packet events. It cannot accurately model the processing at the nodes, and the load the nodes may encounter due to that processing. It also cannot accurately model packet delays and losses due to characteristics of the environment.

We describe the details of implementation using netfilter framework in Section 5.1. We describe the testbed and the method of creating the multihop topology in Section 5.2. We present the experimental results in Section 5.3.

5.1 Implementation details

We implement DIPLOMA in Debian Linux system running 2.6.30 kernel by creating a user space DIPLOMA engine that interacts with the Linux packet processing system using netfilter queue framework. In our implementation, the user applications do not require any change, and the engine does all DIPLOMA related processing. We use a popular implementation of AODV from University of Uppsala (AODV-UU) for MANET routing [AOD]. In this section, we describe the details of our implementation.

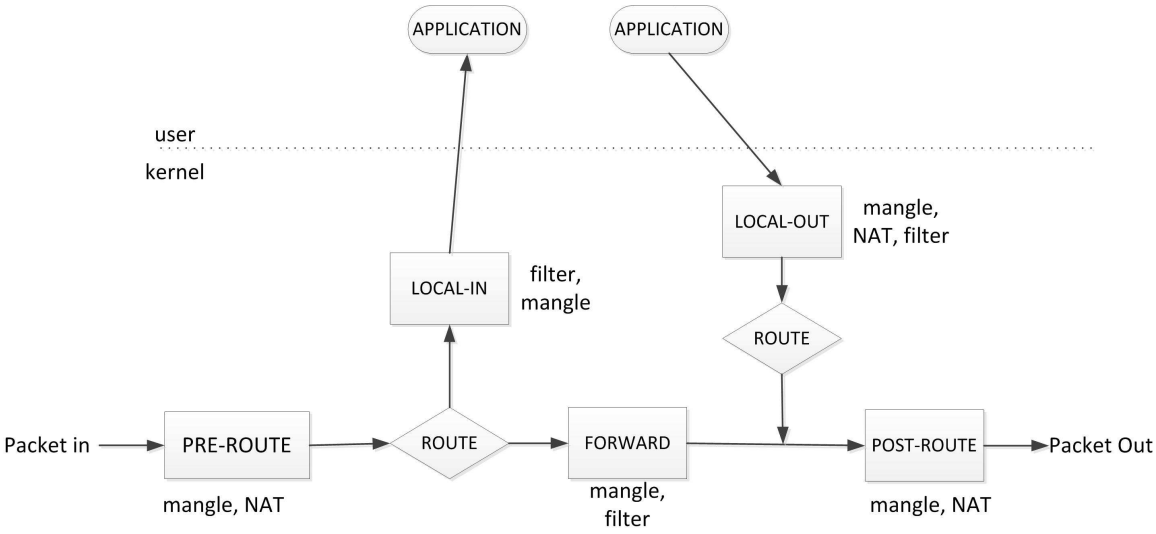

Figure 5.1: Linux Netfilter Architecture

5.1.1 Netfilter overview

Linux kernel provides a series of hooks for intercepting and manipulating packets at various points in the protocol stack through the *Netfilter* framework [net]. Figure 5.1 shows the Linux netfilter framework. There are five types of hooks [RW02]. When the packet enters the system from the network it passes through the PRE-ROUTE hook. Then the packet goes through the routing module. If the packet is destined to that host, then the packet goes through the LOCAL-IN hook, before it is passed on to any local process. If the packet is destined for another network interface, then it goes through the FORWARD hook. Then the packet is send to the POST-ROUTE hook, before it is put in the wire again. If the packet is created locally, then the LOCAL-OUT hook is called before taking any routing decision. The packet will also go through the POST-ROUTE hook after the routing is done, before it leaves on the wire.

A kernel module can register to receive the packets on one or more of these hooks by providing the netfilter a callback function. The callback function is invoked when a packet hits the registered hook. This function can modify the packet and send a "verdict" on the packet. There and five verdicts: continue the normal processing (NF_ACCEPT), drop the packet (NF_DROP), the module has consumed the packet (NF_STOLEN), queue the packet

for user space handling (NF_QUEUE), or call the hook again (NF_REPEAT).

A set of tables has been built on top of the netfilter for packet selection. A user level program called *iptables* allows system administrators to configure these tables. The tables can be used to filter packets ('filter' table), perform a network address translation ('nat' table), or do a pre-route packet mangling ('mangle' table). Figure 5.2 also shows which tables are present with each of the netfilter hooks. Using iptables, one can also send the packet to the user space.

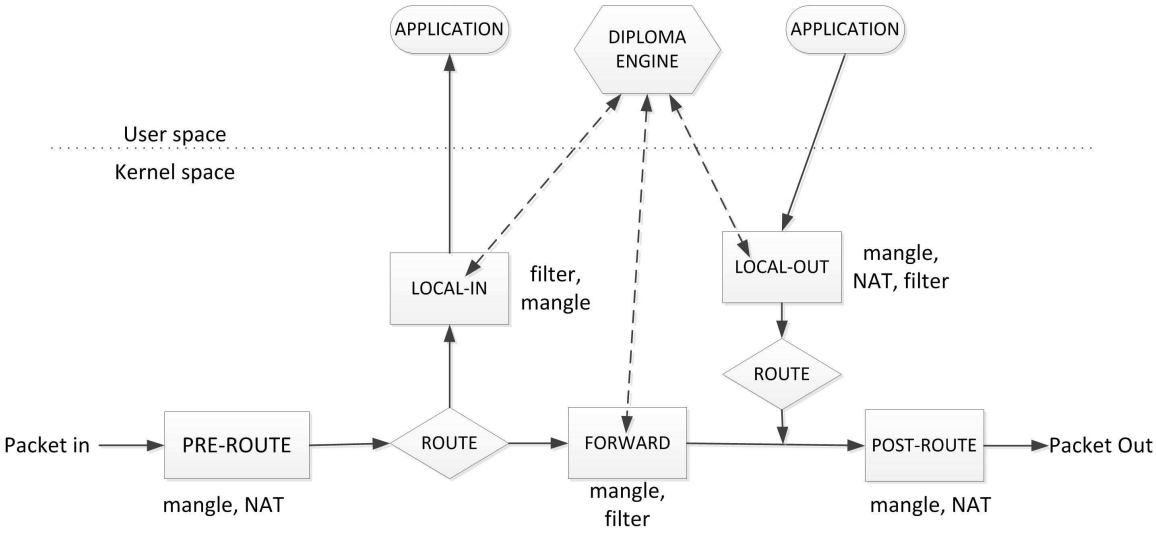

Figure 5.2: DIPLOMA implementation on Netfilter Architecture

A user-level library called *netfilter-queue* provides APIs for manipulating packets that has been queued by the kernel filter. This API can bring a queued packet into user space, manipulate the packet, and provide a verdict on the packet. This library can be used in conjunction with iptables to implement any user level protocol processing.

5.1.2 DIPLOMA Implementation

DIPLOMA is implemented as a user-level daemon (called the DIPLOMA engine), using the netfilter-queue library. At the system startup, iptables rules are added to the mangle table, so that all the packets that are leaving, entering or transiting the system on selected interfaces are received by the engine. All the processing related to the DIPLOMA protocol

are handled solely inside the engine. The application processes do not have to be aware of this processing, and our implementation does not require any changes to the applications. The DIPLOMA engine performs the following packet processing operations:

- It adds DIPLOMA headers and packet signatures to all the outgoing packets from the system. It also enforces the outgoing bandwidth constraints.
- It removes DIPLOMA headers of the incoming packets, before the packet is given to the application.
- It verifies the packet signatures and the bandwidth usage of transiting and incoming packets.
- It handles all the DIPLOMA protocol packets and their processing, including capability establishment, capability refresh, error handing, *etc.*

All the data packets transiting through the network contain a new header, called *capability header*. This is placed between the IP header and the transport header (TCP/UDP). The protocol field in the IP header is changed to indicate the presence of the new header. The capability header contains the type of the packet, a transaction identifier to identify the capability associated with the packet, a sequence number, and the signature for the packet. The size of the capability header in our implementation is 36 Bytes. The capability header is added by the DIPLOMA engine at the source node after receiving the packet from the application. The header is removed by the engine at the destination, before the application receives the packet.

5.1.2.1 Capability Establishment

Capability establishment is the process of establishing the mapping between the transaction identifier, and the actual capability and the keys for the packet signatures on the path from the source to the destination for any flow. The intermediate nodes store the relevant information in a data structure for easy lookup and update. Once the capability is established the source node does not include the actual capability in the data packets. Instead, the

transaction identifier and the packet signatures are used by the intermediate nodes to verify the packets.

5.1.2.2 Sending data

When an application sends a packet, the kernel protocol stack receives the packet. The packet hits the OUTPUT hook, and it is queued by the iptables mangle rule for the user space processing. The DIPLOMA engine receives the packet using netfilter-queue API. If the packet is the first packet in the flow, a capability needs to be established for that flow. The DIPLOMA engine looks for a matching policy token in its possession. If there is a matching policy token, then the engine queues up the packet and initiates the capability establishment protocol. Once the capability establishment is complete, the engine adds a capability header to the original packet and sends the modified packet with an ACCEPT verdict to the kernel. The kernel continues the normal packet processing for the packet, and sends it to the corresponding network interface for transmission. If the application sends more packets while establishing the capability, those packets are also queued by the DIPLOMA engine. If the engine receives a packet from an application for which it does not possess a policy token or a network capability, it gives a DROP verdict and the packet is dropped by the kernel.

The DIPLOMA engine also enforces the bandwidth constraints of the capability associated with the outgoing packet. If the sender does not enforce these bandwidth constraints, the packet will eventually get dropped at the next hop.

5.1.2.3 Forwarding data

All the transit packets are queued by the kernel filters and send to the DIPLOMA engine using iptables rule on the FORWARD chain on the mangle table. The engine in the forward path verifies the data packet against the capability and gives a verdict. The verification involves checking whether there is a capability associated with the packet based on the source address and the transaction identifier, whether the packet signature is correct, and if the packet is in conformance with the bandwidth constraints. The engine does not modify the packet during this process.

A *token bucket algorithm* is used to enforce the bandwidth allocated to a capability. A capability contains two bandwidth parameters: the *rate* and the *burst size*. The rate is number of bytes per second the node possessing the capability is allowed to transmit. The burst size is the size of the bucket. Whenever the engine sees a packet, it first updates the available tokens in the bucket based on the rate and the bucket size, and then removes the number of tokens equal to the size of the packet. If the available token is less than the size of a packet, then the packet is dropped. The engine updates the token bucket only when there is a corresponding packet. It keeps track of the time the token bucket was last updated to perform the proper accounting.

5.1.2.4 Receiving data

All the incoming packets are queued by the kernel filters for the DIPLOMA engine processing by adding iptables rules on the INPUT chain on the mangle table. When the packet is received by the engine, it first verifies that the packet is in conformance to the capability, similar to the packet forward path. If it is not conformant, the packet is dropped by giving a DROP verdict. Otherwise the capability header is removed from the packet, the IP header is modified to reflect the correct network protocol and size, and the modified packet is sent back to the kernel with an ACCEPT verdict. From that point, the normal packet processing happens inside the kernel and the packet is sent to the user application.

5.1.2.5 Fragmented packets

A Packet received on the OUTPUT hook by the DIPLOMA engine is usually of the MTU size. The DIPLOMA engine adds a capability header to the packet, making the size of the packet more than the MTU size. When this modified packet is send back to the kernel with an ACCEPT verdict, the protocol stack will fragment the packet. These fragments pose two issues in systems running DIPLOMA. First, the fragments cannot be associated with the capability, as they do not contain the transaction IDs. Second, the intermediate nodes will be unable to verify the fragment signatures as the DIPLOMA engine computes the signature for the whole packet.

The fragmentation problem is arising because our implementation of the DIPLOMA

engine is outside of the network stack. If the engine was integrated into the network stack, then it can take care of the fragmentation issue. We cannot solve the fragmentation issue by reducing the MTU size of the system, as the network stack will end up fragmenting the packets using the new MTU size.

We address the problem in two ways. First, we forward a fragmented packet at the intermediate node only after receiving all the fragments and verifying them against the capability. Second, for TCP we reduce the maximum segment size (MSS) of TCP packets by the capability header size, using iptables rules. Hence, the TCP stack will packetize the data that leaves enough room for the capability header without exceeding the MTU.

5.2 Testbed

We implement the DIPLOMA engine as described in Section 5.1 in Linux systems running Debian Linux with kernel 2.6.30. We ran the resulting system on multiple nodes in the Orbit lab [orb] wireless testbed. Orbit is an indoor wireless testbed consisting of 400 nodes arranged as a 20x20 grid on a physical area of (20m x 20m). Each node contains 1-GHz VIA C3 processor, 512 MB RAM, a 20 GB hard disk, two wireless mini-PCI 802.11 a/b/g interfaces, and two 100BaseT Ethernet ports. Most of the cards are Atheros AR5212-based cards, although there are a few Intel Pro-wireless 2915-based cards as well. In our experiments, we use only the nodes that have Atheros cards. We use only the wireless interfaces for experimental traffic.

As the DIPLOMA engine is a user-level process, all packets are queued for user-level processing before transmission. To make a fair comparison, we also do a similar queuing of the packet to a user level process on systems not running the DIPLOMA (called *original*). The user level program gives an ACCEPT verdict on all the packets, without any processing.

5.2.1 Topology creation

Every node in the Orbit grid is reachable by every other node using wireless links, as the grid is housed in a relatively small physical area. The large range (about 300ft) of 802.11 makes it very difficult to create true multi-hop topologies in indoor environments. A solution used

is to create a network layer topology by filtering out packets of the non-adjacent senders based on their MAC addresses at the receivers. Another approach is to generate large noise [KGS06]; so that the receivers farther from the sender will not have enough signal to noise ratio (SNR) to decipher the signal. Orbit has four noise generator antennas for this purpose. Unfortunately, only a limited set of small topologies can be created with this approach, and the topology creation requires extensive trial and error to find the right noise levels.

MAC address filtering based topologies are not useful for studying the security properties of a system like DIPLOMA. This is because an attacker node can cause damage in its communication radius by transmitting large amount of data, even if it is not conforming to the protocol. Since the packet filtering is done at the receivers, the attacker can occupy most of the available bandwidth. In an indoor wireless setting like Orbit, the communication radius is the whole grid.

We use a different approach for creating the multi-hop topologies; we use different non-overlapping channels for each link. Since orbit has two 802.11 antennas at each node, the intermediate nodes switch the channels when forwarding the packet. We primarily used 802.11a channels in our experiments, as the channels are not overlapping. We have also used non-overlapping 802.11g channels (channels 1 and 11). Even though we cannot create all the possible topologies similar to the MAC address filtering, it is possible to create a good number of multi-hop topologies with this approach.

Figure 5.3 shows an example multi-hop topology. For the simplicity of discussion, let the nodes be numbered $1, 2, 3, \ldots$. Let the useful channels also be numbered $1, 2, 3, \ldots$. We assign the IP address *10.1.c.i* to an interface on the node i that is transmitting on the channel c. Two nodes can communicate directly, if they have antennas with a common channel. In our example, two antennas share a channel if they are on the same 255.255.255.0 subnet. In the figure, the nodes are numbered first from left to right (1 to 5), and then from bottom to top (6 to 9). The connectivity is shown using dashed lines. Note that nodes 2, 3 and 7 have pair wise connectivity due to sharing of channel 2. In general, the connectivity graph is a set of cliques, one for each channel.

We modify the Linux AODV implementation, AODV-UU to handle multiple interfaces,

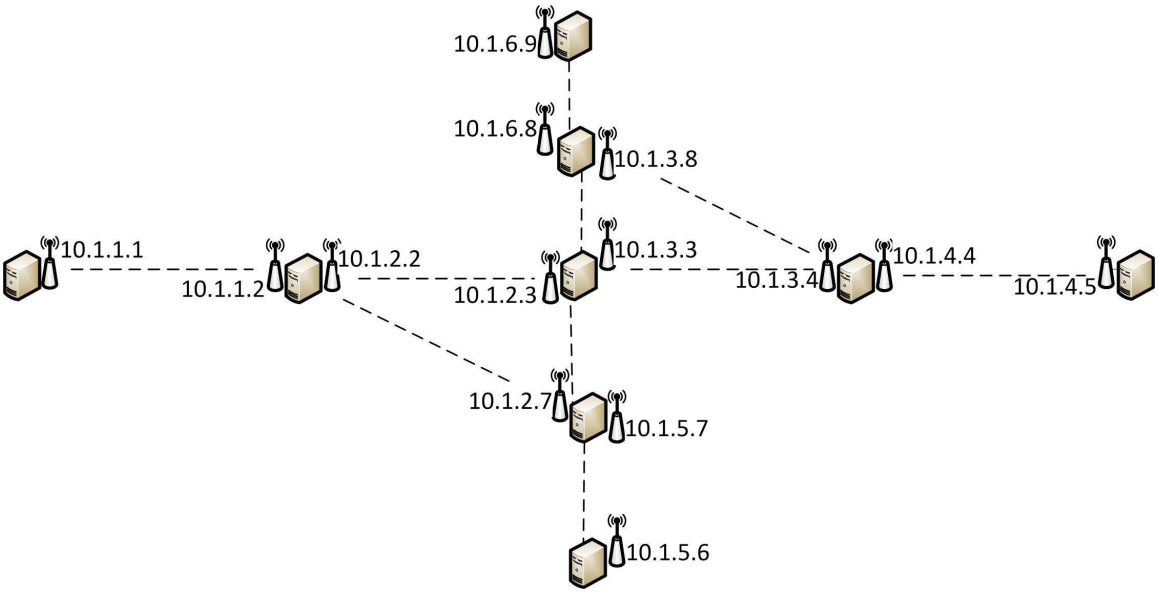

Figure 5.3: A Multi-hop topology

as it did not have the support for multiple interfaces.

5.3 Experimental Evaluation

In this section, we describe the experiments conducted to study the effectiveness of DIPLOMA, and the results and the analysis of those experiments. Our goal is to study the impact on performance on systems running DIPLOMA, and on effectiveness of containing attacks on those systems. To that end, we study the throughput of the systems running DIPLOMA to identify the block sizes that provide the optimal throughput. We also study the effectiveness of DIPLOMA in enforcing the bandwidth constraints. We compare the throughput, latency and jitter for systems with and without DIPLOMA. We also show how the DIPLOMA can protect the end-hosts and the network bandwidth in the presence of attackers. Finally, we show that DIPLOMA can work well in the presence of multiple flows.

5.3.1 Throughput

In this section, we study the throughput of TCP and UDP at different packet block sizes. In this set of experiments, we create a linear topology of multiple nodes, and send high data rate traffic using *iperf*. For the DIPLOMA scheme, we allocate unlimited bandwidth to the capability.

5.3.1.1 Block size

We study the effect of block size on the throughput for TCP and UDP, and select the best block size for further experiments. Recall that the larger block sizes require smaller processing at the sender, but it also has higher per packet overhead and larger penalty on packet losses. The results are average of four iperf for 10 seconds each. The block timeout is 25 ms. The results are similar for 10 ms block timeout.

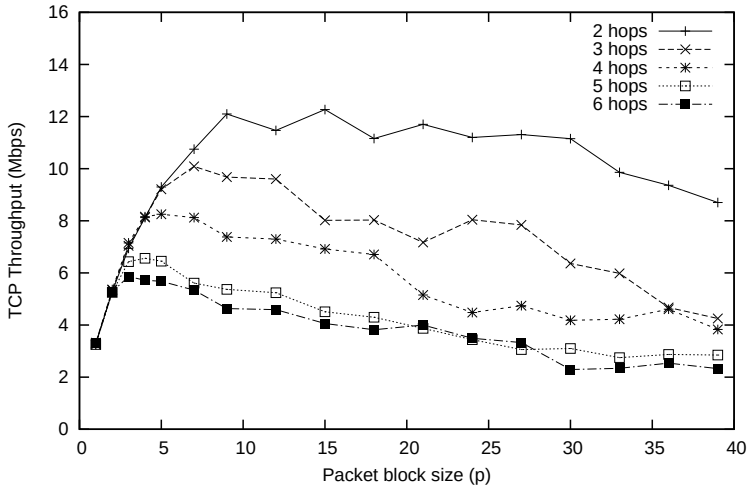

Figure 5.4: TCP throughput for different block sizes

Figure 5.4 gives the TCP throughput for different block sizes on paths of increasing hop lengths. The labels on the plots indicate the path length. For smaller block sizes, the throughput increases linearly. The throughput at that point is limited by the sender's ability to pump traffic, which is limited by the signature computation. As we increase the block size, the processing needed per packet at the sender decreases, and the send rate catches up with the available bandwidth. This linear increase in the throughput as the block size

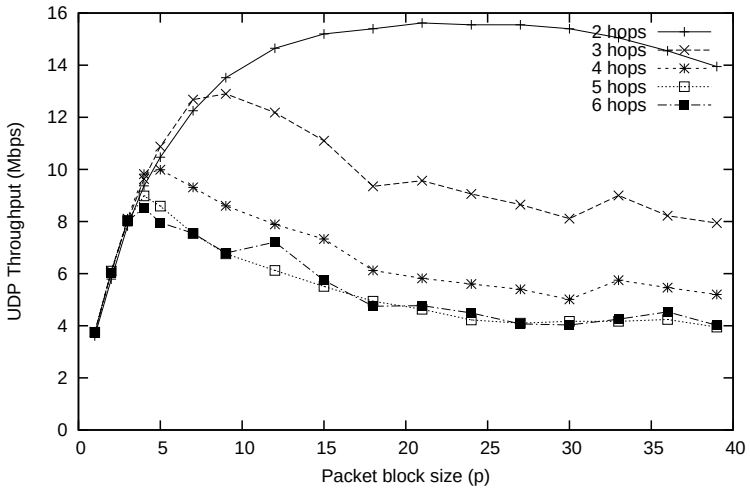

Figure 5.5: UDP throughput for different block sizes

increases stops at certain block size. This block size, where the throughput increase stops, is larger for smaller hop counts. This is because, the end-to-end bandwidth decreases as the hop count increases. Once the throughput reaches a maximum point for a given hop length, further increase in the block size decreases the throughput. This is due to two factors. Firstly, the header size increases as the block size decreases, reducing the MSS of the TCP. Second, packet loss of a DATA-FIRST packet will force the entire packet block to be retransmitted, losing usable bandwidth. The best packet block size for the TCP is between 4 and 9. The best block size decreases as the number of hops increases, which can also explained using a similar argument.

Figure 5.5 gives a similar result for UDP. Here, the sender was pumping traffic at 20 Mbps. The results for UDP are similar to that of TCP, except that the UDP has higher bandwidth. This is because UDP does not have to deal with the complexities of TCP like guaranteed delivery. Because of the higher bandwidth, the block size at which it achieves maximum throughput is also larger. The block size between 5 and 10 gives the maximum throughput for hops 3 to 6.

We use a block size of 7 for our experiments, as it is a good compromise for UDP and TCP, and for different hop counts.

5.3.1.2 Comparison with original scheme

Now we compare the throughput of the system with and without DIPLOMA. For the DIPLOMA, we use the block size of 7 and allocate unlimited bandwidth to the capabilities.

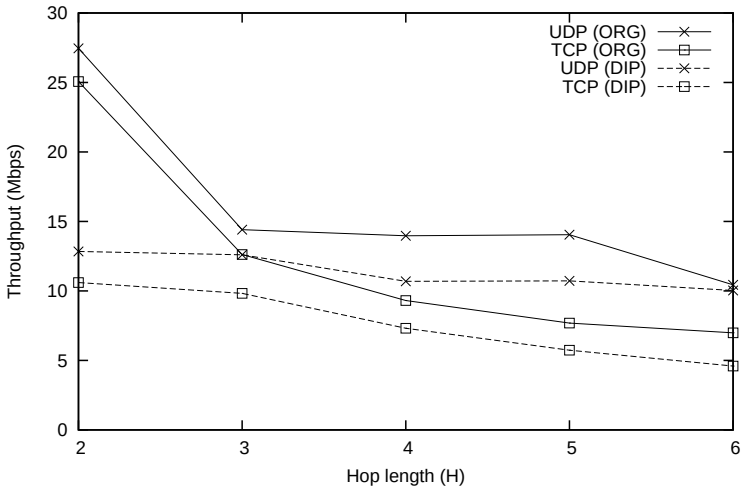

Figure 5.6: TCP and UDP throughput comparison

Figure 5.6 shows the iperf throughput for TCP and UDP for both the DIPLOMA and the original schemes for various hop lengths. The DIPLOMA throughput is about 23% lower for TCP and 19.8% lower for UDP for hop lengths between 3 and 5. This is because of the extra headers and the extra processing required for the DIPLOMA. For 2 hop distance the bandwidth for the original scheme is substantially higher than that of the DIPLOMA scheme. This is because, at 2 hops the available bandwidth is high, and hence the bottleneck becomes the processing delay.

5.3.1.3 FTP traffic

Now we study the system with the real file transfer application. Recall that our system does not require any changes to the existing application programs. We study the performance of FTP using a linear topology. The first node in the linear topology runs a FTP server. The client, which is at $2, \ldots, 7$ hops, downloads a file of size 20 MB from the server using wget. The FTP throughput was reported by wget. We report the results based on average of 8 FTP downloads. For the DIPLOMA scheme, we allocated unlimited bandwidth to the

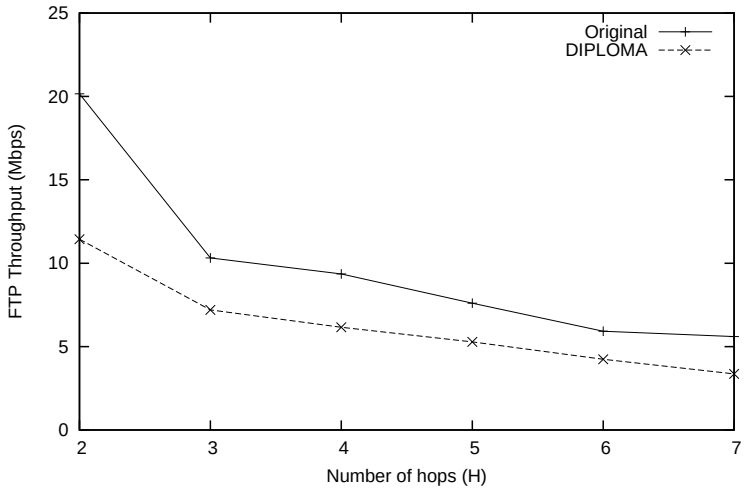

Figure 5.7: FTP throughput

capabilities.

Figure 5.7 shows the FTP throughput for both the DIPLOMA and the original schemes for different hop lengths. The results are very similar to TCP throughput results presented in Section 5.3.1.2. At a hop length of two, the throughput for the original is considerably higher than the DIPLOMA, because of the cryptographic operations. At hop lengths between two and six, the DIPLOMA throughput is between 28% and 34% lower than the original.

5.3.2 Bandwidth Enforcement

In this subsection, we study the effectiveness of the bandwidth enforcement capabilities of our implementation. We use a linear topology for these sets of experiments. We allocate different bandwidth to the capabilities and measure the end-to-end throughput using iperf. Recall that we use the token bucket parameters of rate (in bytes per second) and bucket size (in bytes) to represent the bandwidth constraints. In our experiments we have used the bucket size as $rate/4$, which allows for a burst equivalent to 250 ms.

Figure 5.8 shows the TCP throughput for various bandwidth allocation for capabilities on paths of different hop lengths. The labels on the plots indicate the bandwidth allocated to the capability. The system is able to enforce the bandwidth constraints, as long as there is sufficient available bandwidth. The throughput seen by the TCP is lower than the allocated

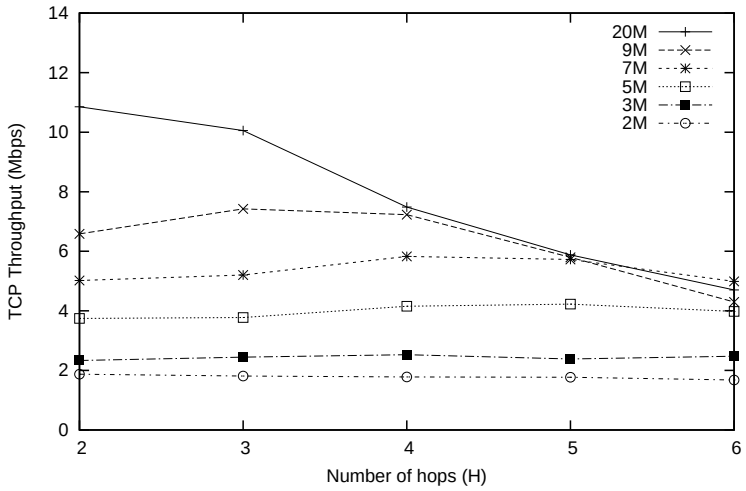

Figure 5.8: Throughput received at various bandwidth allocated to capabilities for TCP

bandwidth due to capability header overhead. The block size used was 7, which could cause the DATA-FIRST header to be as large as 152 bytes. Even though DATA-NEXT header is smaller (40 bytes), the MSS of TCP was set to 1308 (MTU - TCP/IP header - 152). When the allocated bandwidth is more than the available bandwidth, then the flow receives all of the allocated bandwidth. The plot labeled 20 Mbps is practically the maximum throughput achievable by the system, because of the limited available bandwidth.

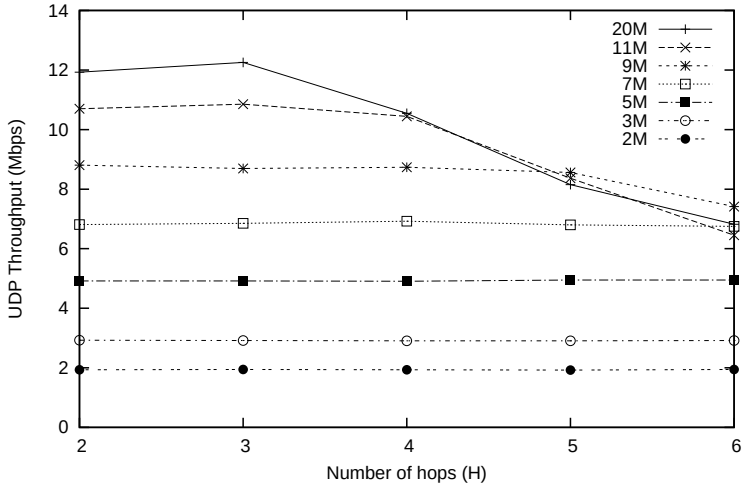

Figure 5.9: Throughput received at various bandwidth allocated to capabilities for UDP

Figure 5.9 shows similar results for UDP. The enforcement on the UDP traffic gives

close to the allocated bandwidth uniformly across the hops. This is because, unlike TCP, there is no flow control in UDP, and the excess bandwidth is dropped closer to the source. There are enough tokens available at the intermediate nodes, so the packets that are allowed to leave the source, except for the synchronization case (Section 3.6.1). As expected, the throughput drops for larger bandwidth or hop lengths, when the available bandwidth is less than the allocated bandwidth.

Hence, DIPLOMA is able to enforce the bandwidth constraints of the capability. If the allocated bandwidth is more than the available bandwidth, then DIPLOMA allocates the available bandwidth to the capability.

5.3.3 Latency and Jitter

We now study the packet latency and jitter for the DIPLOMA, and compare it with the original scheme.

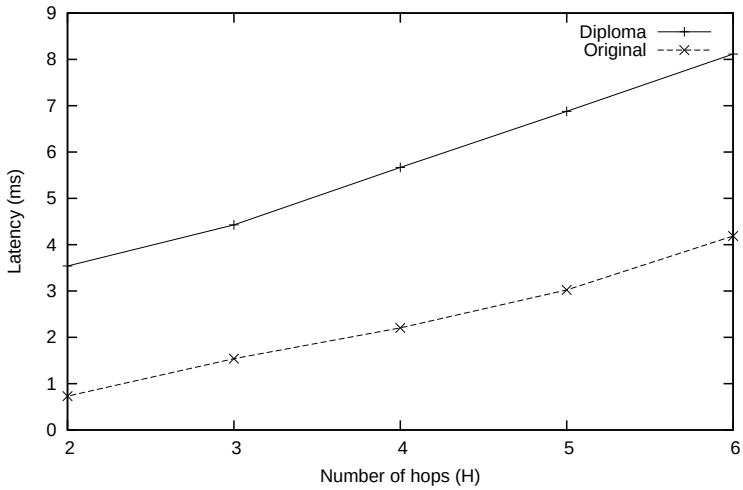

Figure 5.10: Packet latency for DIPLOMA and original schemes

5.3.3.1 Packet Latency

To measure the latency, we use *ping* command to send ICMP echo request/reply. Ping gives the round trip latency, which involves the capability processing delay at both the ends. One important parameter that affects the latency is the block timeout, which is the amount of time the sender has to wait for the packet block to fill before sending the packet. To remove

the effect of packet block time, we use the packet block size of one. In this case, the engine sends the packet as soon as it receives the packet.

Figure 5.10 shows the average round trip latency reported by ping for 20 packets for different hop lengths. As expected, the latency increases close to linear, as the hop length increases. The average latency for the DIPLOMA is 2.8 ms to 3.9 ms higher than the original scheme. This is mainly coming from the packet signature operation at the senders. Since, we are measuring the round trip latency of ping packets, there are two signature operations: at the ping source and the ping destination. There is also a small additional processing required at the intermediate nodes for packet validation.

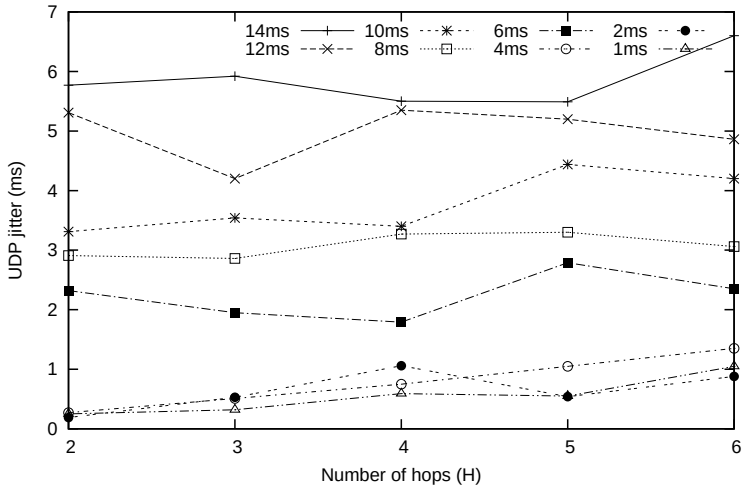

Figure 5.11: Packet jitter for DIPLOMA and original schemes

5.3.3.2 Packet Jitter

Packet jitter is an important factor in some of the real world applications like Voice over IP (VOIP). Jitter is defined as the average deviation from the mean latency. To measure the jitter, we send UDP packets at low rate (1 Mbps) using iperf. iperf for UDP reports the jitter values. One important parameter that affects the jitter in DIPLOMA is the block timeout; the packet may wait for the timeout period before the engine sends it, if the block is not full. This is especially true for low bandwidth applications. For small bandwidth latency and jitter critical applications, it is beneficial to use small block timeout, since the signature operation will not be a bottleneck.

Figure 5.11 shows the jitter for different block timeout as a function of hop length. The labels on the plots indicate the block timeout. The block size in these experiments was 7. As expected the jitter increases with increase in block timeout. Surprisingly, we do not see any correlation between the hop count and the jitter values. The maximum jitter value for up to six hops is 1.35 ms, for a block timeout of 4 ms. The maximum jitter value in this set of experiments is only 6.60 ms; this occurs on a six hop path at the block timeout of 14 ms.

5.3.4 Attacker resiliency

A major goal of the DIPLOMA is to protect the end-host resources, including the protection against denial of service attacks, by dropping unauthorized traffic closer to the source. In this section, we study the resiliency of DIPLOMA towards any attacker that does not conform to the protocol and the bandwidth allocation.

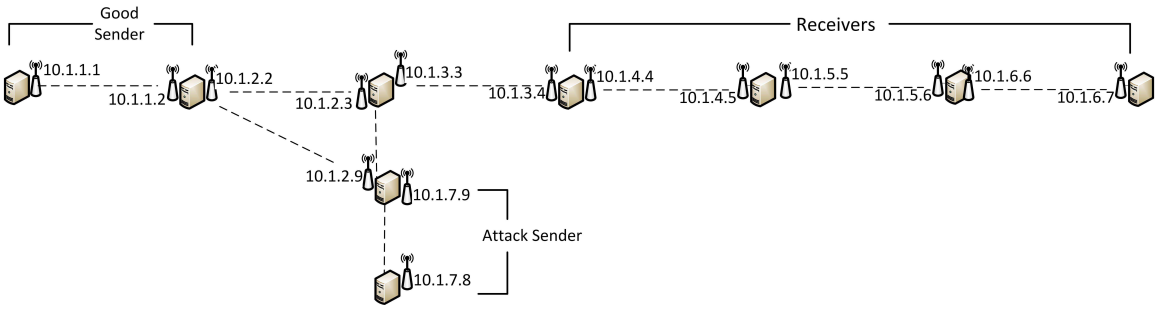

Figure 5.12: Topology to study attack resiliency

To study the attacker resiliency, we used the topology on the Figure 5.12. We conduct two sets of experiments here. In the first experiment, the good node and the attacker share the channel. In this case the good node is node 2 and the attacker is node 9. In the second experiment, there is no sharing of channel between the attacker and any node in the path from the good node to the receiver. In this case, the good node is node 1 and the attacker is the node 8. Both the nodes send traffic to the same destination at different hop lengths: to nodes 3,4,5, 6 and 7. In both sets of experiments, the attacker is allocated only 1 Mbps bandwidth, but the good node is allocated either 2Mbps, 3 Mbps, or 5 Mbps. The attacker

sends as much data as it can, but the good node conforms to the allocated bandwidth.

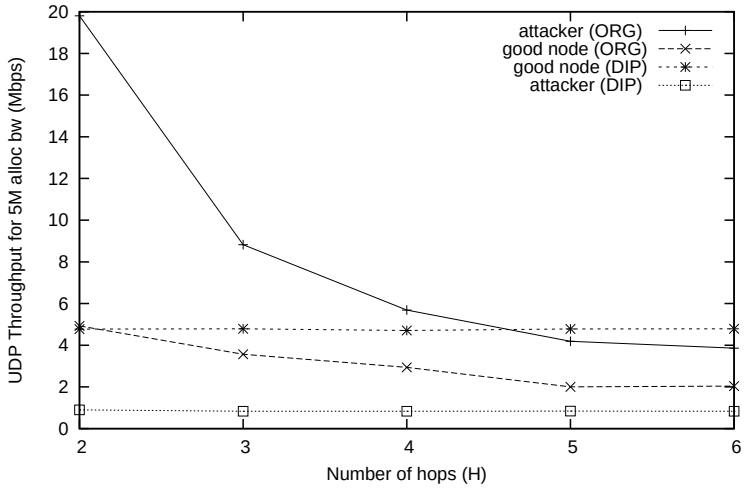

Figure 5.13: Resilience to an attacker who is sharing the spectrum

5.3.4.1 Attacker and the good sender sharing the spectrum

Figure 5.13 shows the iperf bandwidth for both the good node and the attacker for both the schemes, when the bandwidth allocated to the good node is 5 Mbps. As can be seen, for DIPLOMA, both the nodes receive the allocated bandwidth. In spite of attacker sending large bandwidth, the two hop node drops any excess bandwidth. This essentially protects the end host (receiver), from receiving the unnecessary traffic. The spectrum sharing did not have an effect here, as the good node was able to deliver the allocated bandwidth to the two hop node. In the original scheme, the node that sends the maximum traffic (i.e. the attacker) gets the maximum bandwidth, at the expense of the other nodes.

5.3.4.2 Attacker and the good sender not sharing the spectrum

Figure 5.14 shows similar results when the attacker and the good sender are not sharing the spectrum. Here the good node and the attacker get the allocated bandwidth for the DIPLOMA scheme. The attacker hogs most of the bandwidth in the original scheme. It is also interesting to see that the good node in the DIPLOMA scheme ends up getting more bandwidth than the attacker gets in the original scheme for six-hop path. The same thing happens at the hop 5 for the shared spectrum case.

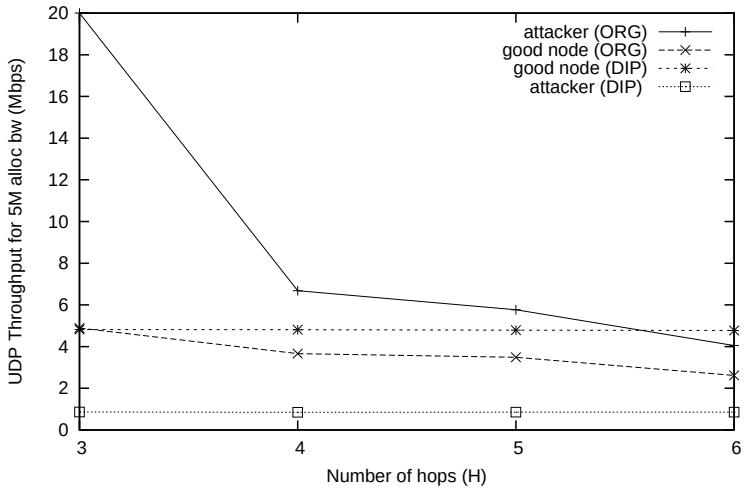

Figure 5.14: Resilience to an attacker not sharing the spectrum

5.3.5 Multiple flows

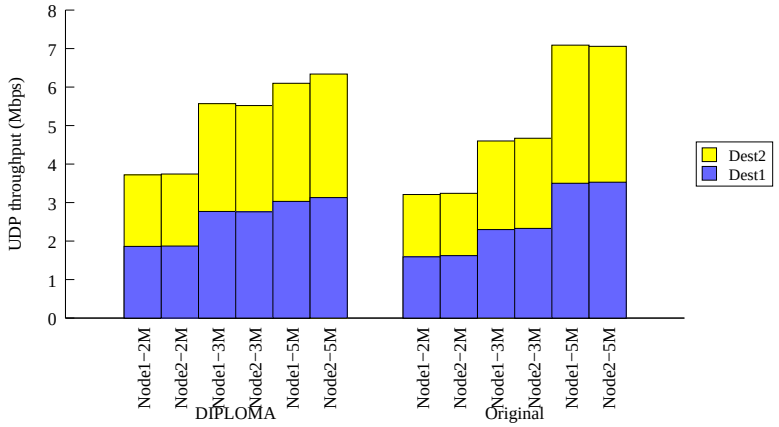

Figure 5.15: UDP throughput when there are multiple flows in the system

Now we study the performance of the system for multiple flows. We use the topology given in Figure 5.3 with two source nodes and two destination nodes. Each of the source nodes sends UDP iperf traffic of varying bandwidth to each of the destination nodes. Hence, there are four parallel flows in the system. The source nodes are nodes 1 and 6 in the figure. The destination nodes are either nodes 4 and 8 (i.e. 4 hop paths), or nodes 5 and 9 (i.e. 5 hop paths). Figure 5.15 shows the resulting bandwidth for both the schemes for 4 hop paths, for varying UDP bandwidth. The label on the x-axis shows the source node and the requested

UDP bandwidth. Each bar shows the received bandwidth at both the destinations. For the DIPLOMA scheme, the capabilities have the same allocated bandwidth as the requested UDP bandwidth. The results are similar for five hop paths, but with lower bandwidth.

It can be seen from the histogram that both the schemes receive roughly same amount of bandwidth for all the four flows. The DIPLOMA receives slightly higher bandwidth for the requested bandwidth of 2 Mbps and 3 Mbps. At 2 Mbps speed, the packet loss are minimum, and the send rate by iperf is higher for DIPLOMA. This may be because the iperf datagram size for the DIPLOMA was 1430 bytes, whereas the size was 1470 for the original. When the requested bandwidth is 5 Mbps, DIPLOMA and the original receive only 3.11 Mbps and 3.53 Mbps respectively. This is because, the available bandwidth was less than 5 Mbps. Since DIPLOMA has larger overhead, its throughput was less than the original.

Chapter 6

Securing Multicast Traffic

Multicast enables delivery of information from one source to many destinations efficiently, without the source unicasting to individual destinations. In multicast, the nodes send packets over a link only once. They create copies of the packets, and send to multiple links when the packets need to go on multiple links to reach the destinations. Multicasting is used for content distribution applications, like audio and video streaming.

Mobile ad-hoc networks are increasingly used in tactical military and civil rapid-deployment networks, including emergency rescue operations and disaster relief network, due to their flexibility in deployment. Audio and video content distribution is an important application on these networks, making support for multicast an absolute necessity. Multicast also improves the efficiency of wireless links in MANETs, due to the broadcast nature of the medium.

The set of nodes receiving the messages that are addressed to a common *multicast address* form a *multicast group*. Traditionally, there are three properties of multicast group [Amm03]:

1. All the members receive all the packets send to the multicast group.
2. Any node can join the multicast group.
3. Any node can send packets to the multicast group.

All these properties have security implications, and there are solutions proposed for them in the context of the (wired) Internet. Most of these solutions differentiate the routers from

the receiver nodes (or multicast group members), as it is the case in wired networks. The routers are secure and well behaved. These solutions are not suitable for MANETs, since the nodes play the dual role of receivers (and senders) of the traffic and routers for forwarding other node's traffic. Furthermore, exploiting these properties increases the resource usage, making multicast an easy tool for launching denial of service attacks on resource constrained MANETs. In this chapter, we propose extensions to DIPLOMA architecture, to provide multicast security in MANETs.

Multicast security protocols for wired networks have treated receiver access control and sender access control as two separate problems [JA02]. Receiver access control is provided using a group policy management system and a group member authorization system [Amm03]. Sender access control can be provided using source specific multicast (SSM), in which only single source can transmit to a multicast group. A MANET node's IP address can change when moving between networks, which require explicit sender access control. Furthermore, because of the broadcast nature of the medium, it is much easier to do IP address spoofing in MANETs.

In this chapter, we provide a unified solution for both receiver access control and sender access control for MANETs by extending DIPLOMA to secure multicast traffic. We define capabilities for use with multicast traffic. There are separate capabilities defined for sending and receiving multicast traffic. A node will not be able to send, or join the multicast group without possessing these capabilities. These capabilities also provide bandwidth constraints for the multicast sessions, preventing resource hogging by the multicast group members. The nodes in MANET enforce the access control and bandwidth constraints of the capability in a distributed manner. We propose modifications to multicast protocols to incorporate capabilities and show the modifications for two popular multicast routing protocols On Demand Multicast Routing Protocol (ODMRP) and Protocol Independent Multicasting Spare Mode (PIM-SM).

We implement the multicast DIPLOMA on Linux. Our implementation does not require any changes to existing multicast applications or the PIM-SM multicast daemon. However, the applications see the benefit in terms of receiving only the authorized traffic, and being able to send the allocated bandwidth even in the presence of rogue nodes that are trying

to conduct a DoS attack.

We implement our system in the Orbit Lab testbed. We conduct extensive experiments to evaluate the performance and effectiveness of our system. We show that multicast DIPLOMA incurs minimal overhead in terms of throughput, packet loss, and inter-arrival times. We also study the effect on video streaming in our system. Finally, we show that multicast DIPLOMA is effective against attackers. Note that we do not address confidentiality of the multicast messages. *Group key encryption* is used to encrypt the multicast traffic using symmetric keys. Group key management is used for efficient re-keying for dynamic group memberships [Amm03].

We also implemented support for multicast DIPLOMA in GloMoSim simulator. We use the ODMRP protocol as the multicast routing protocol for the simulation. In this chapter, we also present the results of our simulation studies.

We describe the multicast capabilities in Section 6.1, extensions of DIPLOMA architecture to secure multicast in Section 6.2, and the implementation in Section 6.3. We describe our experimental methodology and results in Section 6.4.

6.1 Multicast Capability

DIPLOMA use multicast capabilities for access control and bandwidth limitations. They have same syntactic structure as unicast capabilities (Chapter 3).

```
serial: 1307467
owner: unit01.nj.army.mil (public key)
destination: 225.1.1.8
service: video
bandwidth: 512kbps
expiration: 2010-12-31 23:59:59
flags: MCAST RW
issuer: captain.nj.army.mil
signature: sig-rsa 23455656769340646678
```

The above represents a policy token assigned by the node captain.nj.army.mil to unit01. This is a multicast capability, since the destination address is a multicast address. Unit01 can multicast video traffic up to 512 kbps to the group 225.1.1.8. The destination field and the flags identify that the capability is for multicast traffic.

There are two types of multicast capabilities: **Multicast Send Capability (MSC)** and **Multicast Receive Capability (MRC)**. The flags in the capability indicate the type of the multicast capability. The nodes possessing a MSC can send traffic to the multicast group, limited by the bandwidth allocation on the capability. They can also join the multicast group and receive traffic from the group. The nodes possessing a MRC can join the multicast group only to receive data; They do not have authority to send data to the group.

The group controllers allocate MSCs, and hence they are of type policy tokens. MRCs can be either a policy token or a network capability. The group controller or a sender that has authority in the form of a policy allocates them.

Unlike the unicast case, network capabilities for the multicast traffic cannot be assigned using the 3-way capability establishment protocol. They need to be allocated outside of the multicast protocol. This is because, the receiver node cannot receive the multicast traffic until it is part of the group; it requires the capability to be part of the group. Furthermore, there are multiple receivers in the multicast group, and the traffic is uni-directional.

6.2 DIPLOMA for Multicast Protocols

Unlike the unicast implementation of DIPLOMA, the multicast implementation depends on the underlying multicast protocol used. This is because a multicast forwarding node does not know about the receiver nodes to enforce the multicast receive capability, without interfacing with the multicast routing protocol. Hence, our implementation, even though it does not directly modify the multicast protocol processing modules, influences the protocol by snooping and filtering the packets. DIPLOMA may also modify the packet immediately before the packet is sent to the physical interface and immediately after it is received on the interface.

There are two types of multicast routing protocols. The first type is flooding based protocols, where the multicast tree is created for the entire topology based on flooding. Later, part of the tree that does not have any receivers is pruned by explicit prune or status discovery messages. An example of this type of protocol is Protocol Independent Multicasting in Dense Mode (PIM-DM). This type of protocol is useful when most of the nodes in the network are members of the group. The second type, which is more predominant, creates a tree (or mesh) based on the membership. A branch in the tree is created only if there a node in that branch that wants to receive the multicast traffic from the group. There is no wasted data bandwidth in this protocol, even though efficiency of the bandwidth usage depends on the type of the tree construction. Examples of this type of protocol include Protocol Independent Multicasting in Sparse Mode (PIM-SM), MAODV, ODMRP, *etc.* In this chapter, we focus on implementing DIPLOMA on this type of protocols.

The receivers are required to send explicit messages to join the multicast tree. This message may traverse multiple intermediate nodes to reach the tree or the node in charge of constructing the tree. Depending on the protocol, the intermediate node may directly forward this message, or send a different message to the same effect to the upstream node. We call these messages collectively **Join-Tree** messages. In PIM-SM protocol, Join-Tree messages constitute IGMP *membership report* message, as well as *Join/Prune* message. In ODMRP protocol, it is the *Join Reply* message serving this role. In DIPLOMA, we make use of Join-Tree messages to send the MRCs. The nodes drop the Join-Tree messages that do not contain valid MRCs. When there are multiple downstream receivers, the forwarding node needs to send only one of the MRCs upstream.

Join-Tree messages forwarded by a node may contain the MRC of its downstream node, instead of its own. This happens when the node is just a forwarding node but not a member of the multicast group. To avoid MRC reuse by rogue forwarding nodes for future multicast sessions, the receivers add an expiration time to the MRC in Join-Tree messages. Receivers sign the (capability, time stamp) pair with their public key. We call that message *time stamped MRC*.

Many multicast protocols have explicit messages initiated by the sender to form the tree. For example, ODMRP has *Join Query* message send by the sender to initiate the tree

creation. Not all protocols have this mechanism. For example, PIM-SM does not require the sender to join the multicast group for sending multicast packets. Hence, we do not rely on any of the protocol messages to send the MSC. Instead, we send the MSC when data traffic starts flowing, like in the unicast case. This has an advantage of treating multicast and unicast data the same way, independent of the underlying protocol. To provide added security, we also send an MSC on protocols that require explicit tree create message from the sender.

An intermediate node forwards a multicast data packet only if both of the following conditions are satisfied:

1. The data packet has an associated MSC from the sender in the node's capability database, and the data packet is conformant to the capability in the form of valid packet signature and the bandwidth constraints.

2. The node has a valid multicast receive capability from one of the receivers in the downstream path. The intermediate node forwards the packet on an interface only if it has a time stamped MRC for a receiver that is reachable on that interface.

A receiving node may leave the multicast tree in two ways depending on the multicast protocol. Some protocols support explicit leave messages. Since it may not be always possible to send a leave message (*e.g.,* the receiver node crashed), the protocols also has periodic membership query. When the receiver node receives a query, it sends some form of a Tree-Join message. In a DIPLOMA enabled system, the receiver node also sends a time stamped MRC in those messages. Then the intermediate (forwarding) node forwards one of the time stamped MRC to the upstream node in its Tree-Join message. When a node does not receive any time stamped MRCs from the downstream nodes on an interface, that interface is pruned from the multicast tree (or mesh).

6.2.1 DIPLOMA on ODMRP

In this section, we describe how to incorporate DIPLOMA on systems running On Demand Multicast Routing Protocol (ODMRP) [LGC99]. We implemented this protocol in GloMoSim simulator. The results are presented in Section 6.4.6.

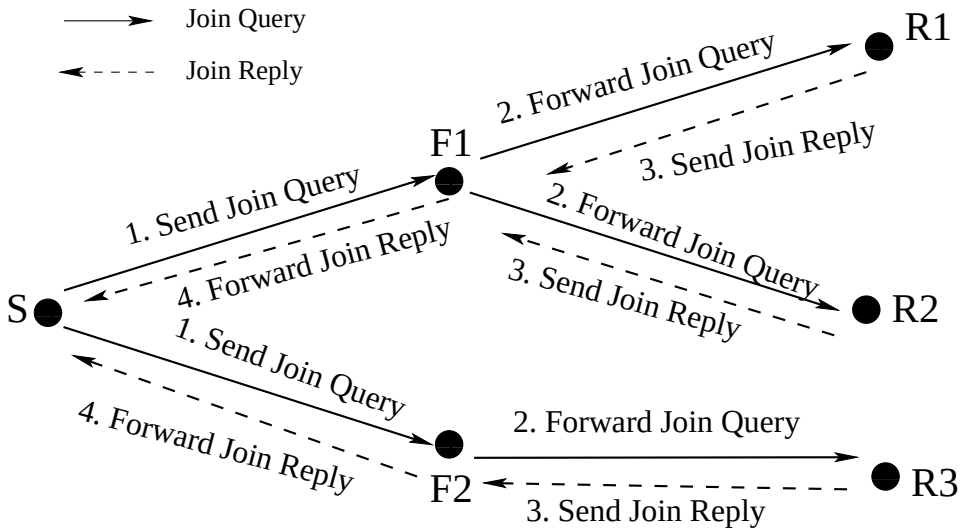

Figure 6.1: ODMRP Protocol

Figure 6.1 gives a high level overview of ODMRP protocol. There is a sender node S that wants to multicast data into a group. Three receiver nodes $R1$, $R2$ and $R3$ are part of the multicast group. Two nodes $F1$ and $F2$ are in the path from the node S to the receivers. We call those nodes intermediate nodes. When the node S has data to multicast, it broadcasts a *Join Query* message to the neighboring nodes to discover a multicast tree. This message is received by the intermediate nodes $F1$ and $F2$, which in turn broadcasts to their neighbors. The nodes $R1$ and $R2$ receives the Join Query from the node $F1$, and the node $R3$ receives the Join Query from the node $F2$. The receiver nodes send a *Join Reply* message back to the nodes from which it received the Join Query message (i.e. the upstream nodes $F1$ and $F2$). Once the nodes $F1$ and $F2$ receive the Join Reply messages they become part of the *forwarding group* and forwards the Join Reply messages to node S.

In DIPLOMA systems that are running over the ODMRP protocol, Join Query messages are modified to contain the MSC of the sender, and the transaction id and the key that will be used by the sender for subsequent communication. The intermediate nodes store this capability information temporarily and forward the Join Query message to their neighbors. On receiving this Join Query, a receiver node in the multicast group responds with a Join Reply message. This Join Reply message is modified to contain the receiver's time stamped MRC that authorizes the node to be part of the multicast group. On receiving a Join Reply,

the intermediate node becomes part of the Forwarding Group. The intermediate node installs the saved MSC in its capability database. The intermediate node then forwards the Join Reply to its upstream node (i.e. towards the sender). It is possible for the intermediate node to receive Join Replies from multiple receivers with different MRCs. The intermediate node needs to forward only one of them to its upstream node. Then the forwarding node starts forwarding the multicast data traffic to the downstream nodes. Similar to the unicast case, the forwarding nodes enforce the capability for all the multicast packets.

Whenever the time stamped MRCs expire, a forwarding node stops forwarding any multicast packet received by the node. ODMRP is a stateless protocol that does not have any multicast leave or prune messages. Instead, the tree is valid only for certain duration. The tree is completely dissolved when that timer expires. Furthermore, the receiver nodes can respond with Join-Reply messages only when it receives Join-Request message from a sender. There is no mechanism for a new receiver to add itself to an existing multicast tree. The sender maintains the multicast tree, and adds new receivers by periodically sending the Join-Query message. The DIPLOMA keeps the time stamped MRC up to date through this periodic tree maintenance protocol. Whenever a receiver gets a new Join-Query message, it creates a new time stamped MRC to respond back in the Join-Reply. To maintain continuous multicast data session, it is important for the period in which a new Join-Query is generated to be less than the validity duration of the time stamped MRC.

6.2.2 DIPLOMA on PIM-SM

In this section, we describe how to incorporate DIPLOMA on system running Protocol Independent Multicast - Sparse Mode (PIM-SM). We implemented this protocol in Linux systems. The implementation and the results are presented in Sections 6.3 and 6.4 respectively.

PIM-SM is a popular multicast routing protocol that is independent of the underlying unicast protocol. This protocol works in conjunction with the Internet Group Membership Protocol (IGMP). The protocol explicitly creates a tree from the sender to the receivers.

In PIM-SM, one of the router is designated as a *Rendezvous Point (RP)* for a multicast group. All the other routers need to join the group through RP. Whenever a node wants to

join a multicast group, it conveys the message through an IGMP *membership report* message. A *designated router (DR)* for the node sends periodic PIM it Join/Prune messages towards the RP for the multicast group. Each router along the path to RP updates the packet forwarding state (routing entries) and sends the Join/Prune message towards the RP.

Whenever a node wants to send the traffic to the multicast group, its DR encapsulates the data in PIM *Register* messages and unicasts it to the RP. The RP decapsulates the message and sends the data towards the receivers in the multicast tree. If the data-rate from the sender is high, then the RP sends a source specific Join/Prune message towards the sender. This extends the tree to the sender, and the sender can directly send multicast messages to the tree without encapsulating the messages. If the data rate warrants it, any DR can join source specific shortest path tree by sending a Join/Prune message towards the sender, and prune the shared tree towards the RP.

We can enable DIPLOMA in multicast systems running PIM-SM by including multicast capabilities in the IGMP and PIM messages. Whenever a receiver sends an IGMP membership report message, its time stamped MRC is included. DIPLOMA systems reject any membership report without a capability. A DR includes one of the time stamped capabilities of the downstream nodes in the Join/Prune messages it sends towards the RP or the source node. When a router receives a prune message, the corresponding time stamped MRC is removed from its tables. The node stops forwarding the packets, when it does not have any valid time stamped MRC from the downstream receivers.

Multicast packets are sent similarly to the unicast case. Before sending a packet, the sender multicasts its MSC in a capability request packet with the capability, transaction identifier and the key for the packet signatures. This packet goes to the RP as a regular multicast packet or a Register packet; the RP in turn sends the packet to the multicast group (after decapsulation for the Register packets). All the nodes in the multicast tree add the capability to their capability database. If it is a register packet, then the nodes in the path between the sender and the RP will also extract the capability, transaction id and the key for the signature from the capability request, and install in their database. Any subsequent data packet multicast by the sender contains the transaction id and the packet signature. The signature is verified and the bandwidth is enforced by all the nodes in the

multicast tree, and by the nodes between the sender and the RP in the case of the Register packets.

If a receiver node joins the multicast tree after the transmission of the initial capability request packet by the sender, then it will not be able to validate the multicast data packets. DIPLOMA solves this by two means: First, the sender periodically multicasts the capability request packet. The new receiver node can start accepting the data packets after the periodic multicast. Second, the receiver sends a request for the capability towards the sender using a DIPLOMA control (or error) packet. On receiving this request, either an intermediate node or the sender responds with the capability and the public key for the signature.

6.3 Linux Implementation

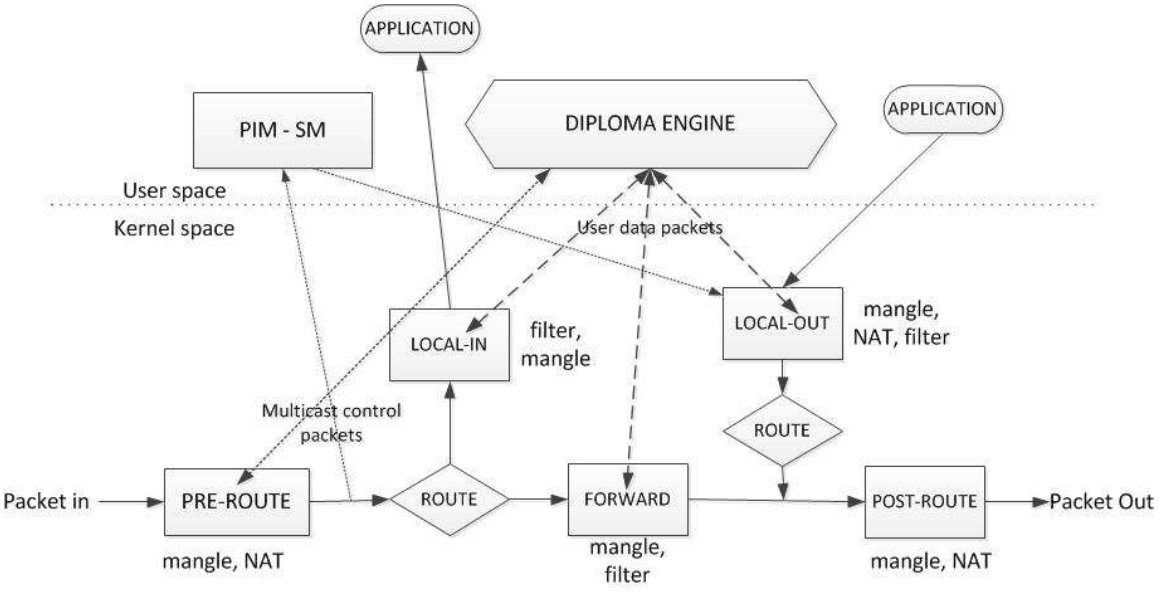

Figure 6.2: Multicast DIPLOMA Implementation on Linux

We now describe the implementation of Multicast DIPLOMA on Debian Linux running kernel 2.6.30. For multicast routing, we use *pimd*, a PIM-SM package that comes with the Debian distribution. Since PIM-SM requires a separate unicast routing, we use University of Uppsala's AODV implementation called AODV-UU. Our implementation does not require

any changes to the application program, routing module or PIM-SM daemon.

Multicast DIPLOMA is implemented as a user level process, called the DIPLOMA engine that interfaces with rest of the Linux packet processing subsystem using the netfilter framework. We use netfilter queue to receive, modify, and filter packets in the DIPLOMA engine.

Figure 6.2 shows how the DIPLOMA engine interfaces with netfilter subsystem. A brief description of Netfilter framework and how the DIPLOMA uses it for handling the unicast traffic can be found in Chapter 5. The dotted lines are the hooks used only for multicast traffic. The solid lines show the hook for both unicast and multicast traffic. The reason for requiring additional hooks for multicast traffic is due to the implementation of PIM-SM in Linux. It uses raw sockets to send and receive traffic; these packets do not go through the LOCAL-IN and the LOCAL-OUT hooks, but traverse the PRE-ROUTE and the POST-ROUTE hooks.

Next, we describe the packet flow for control (*i.e.,* IGMP and PIM packets) and multicast data packets.

6.3.1 Membership Messages

When the system sends a membership message, in the form of an ICMP message or a PIM Join/Prune message, the DIPLOMA engine receives the packet on the LOCAL-OUT hook. It checks for a valid MRC for the message in its database. The valid capability may be either its own capability, or a capability it received from a downstream node. It adds the capability in the packet and sends an ACCEPT verdict on the hook.

When the system receives a membership message on the PRE-ROUTE hook, it validates the packet. A valid packet needs to contain a valid MRC. The node saves the MRC in its tables for subsequent request to the upstream node. The capability is removed from the packet and an ACCEPT verdict is given. The PIM-SM daemon receives this packet over the RAW socket. The engine drops any membership message without a valid capability.

6.3.2 Capability establishment

When a sender needs to multicast data, it creates a transaction identifier for use with subsequent packets to identify the session. It also creates an RSA key for signing the data packets. The sender sends the transaction id, the public key and the MSC authorizing the sender to send the multicast packet as a DIPLOMA control message. The DIPLOMA engine sends this message when it first sees a packet from the sender for a multicast group. The application program sending the multicast data need not be aware of this step.

When a multicast member node or a forwarding node receives this message, it validates the capability and stores the transaction id, the public key, and the MSC in its capability database. These nodes validate the subsequent data packets coming from the sender against the capability and verify the packet signatures. The sender periodically sends the capability establishment packet for updating new receivers or new intermediate nodes after a route change. A receiver can also make a unicast request to the sender to send a capability establishment packet, when it does not have that information due to late joining or a route change.

6.3.3 Multicast Data Packets

All multicast data packets need to contain an associated capability. The DIPLOMA engine at the sender modifies the outgoing packets in the OUTPUT hook by including a capability header, which contains the transaction identifier and the packet signature. Packets sent to a multicast group are treated together as a *block* for the signature computation, similar to the unicast case as discussed in Chapter 5. A packet block contains maximum of *block size (P)* packets that are sent with in the interval *block timeout (T)*. The packet signatures for a block consist of RSA signature for the first packet and SHA-1 hashes for the remaining packets. The RSA signature is verifiable with the key send in the capability establishment phase. The SHA-1 hashes are integrity protected by including them in the first packet.

The engine at the intermediate node receives multicast packets on the FORWARD hook. The engine validates each packet against the capability using the transaction identifier. The validation including checking if there is a valid MSC in its database associated with the transaction identifier, if the packet has valid signature, and if the packet conforms to the

bandwidth constraints of the capability. If the packet is valid, then the engine gives an ACCEPT verdict for forwarding the packet.

If the packet is destined to the node as a receiver on the multicast group, DIPLOMA receives the packet on the INPUT hook. The engine validates the packet as above, removes the capability header from the packet and gives an ACCEPT verdict, causing the kernel to deliver the packet to the application.

6.4 Experimental evaluation

In this section, we evaluate the effectiveness of multicast DIPLOMA. First, we compare the throughput, packet loss and inter arrival times of the systems with and without multicast DIPLOMA by sending periodic traffic. We also study these parameters by sending real video streaming traces. Finally, we study the effectiveness of DIPLOMA in containing attacker nodes.

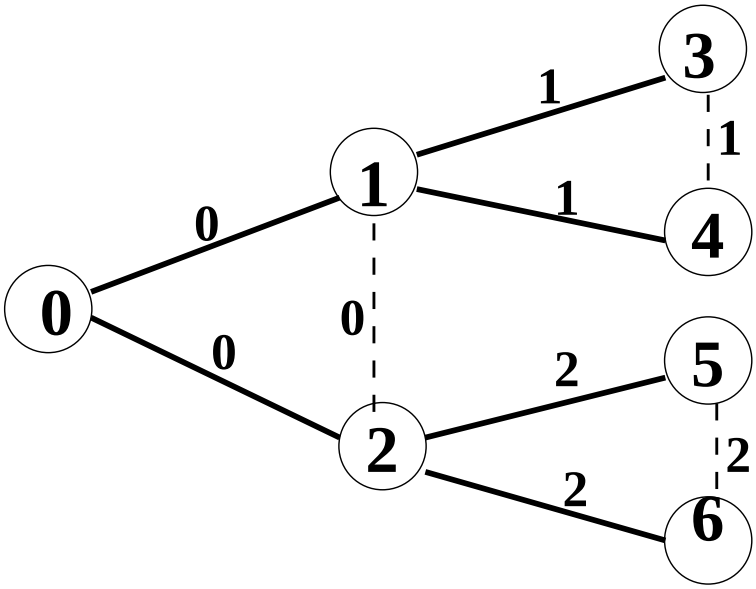

Figure 6.3: Tree topology

6.4.1 Testbed

We implemented the multicast DIPLOMA engine as described in Section 6.3 in Debian Linux with kernel 2.6.30. We use AODV-UU for routing unicast traffic, modified to handle multiple interfaces. For multicast routing, we use PIM-SM implementation called *pimd* that is available with Debian Linux distribution. We ran the resulting system on multiple nodes in the Orbit lab wireless testbed. We created multi-hop topologies using channel hopping as described in Chapter 5. To make fair comparison, we also send the packets of *original* to user level as discussed in that chapter.

For measuring the performance of the DIPLOMA, we use two topologies: a line topology and a tree topology. In the line topology nodes are allocated channels in such a way that each node can directly communicate only with its neighbors on either side (except for the first and last, each of which has only one neighbor). In this topology, the first node is the sender of the multicast. All the remaining nodes subscribe to the multicast group. The tree topology is shown on figure 6.3. The links are labeled with the channel with which the nodes communicate. Here the sender is the node 0 (root), and the multicast receivers are nodes 3,4,5,6 (leaf nodes). In the figure, the solid lines show the multicast tree and the dashed lines shows the links that are not participating in the multicast.

We use the multi-generator tool *mgen* [mge] from Naval Research Laboratory to send and receive traffic in our experiments. Compared to iperf, mgen provides finer control over the experiments and collecting results, and has better support for multicast. Each data points in this section represent an average of running six experiments, each experiments sending traffic for 30 seconds each.

6.4.2 Line Topology

In this set of experiments, we study the performance of the DIPLOMA and the original schemes for the line topology. The sender sends periodic traffic of size 1024 bytes at the rate of 100, 300 and 500 packets per second. This corresponds to rates of 819.2 Kbps, 2.4576 Mbps and 4.096 Mbps respectively.

Figure 6.4 shows the throughput received by the nodes at different hop lengths for different transmission rates. For the rate of 100 and 300 pkts/sec, both the DIPLOMA

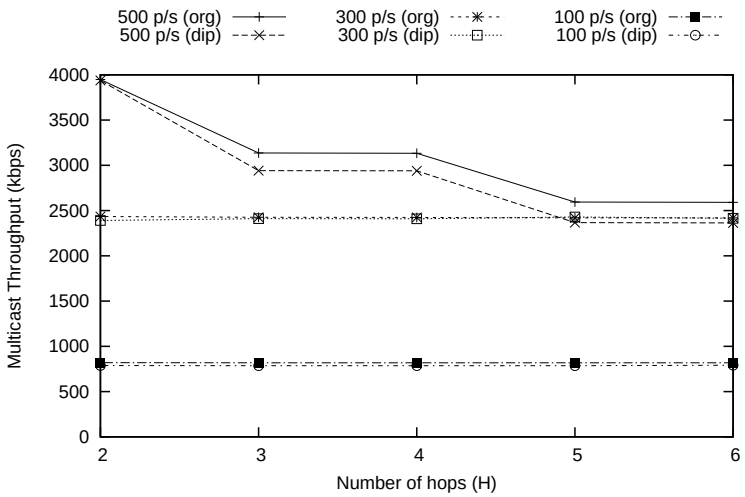

Figure 6.4: Throughput for line topology

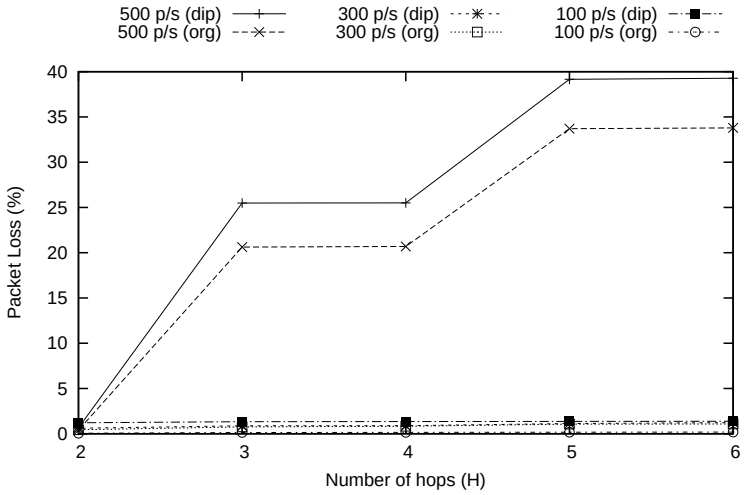

Figure 6.5: Packet loss for line topology

and the original schemes receive bandwidth close to the send bandwidth for all the hops. The bandwidth for the DIPLOMA is minimally (0.7% and 3.7% respectively) lower than the original. For the rate of 500 pkts/sec, the received bandwidth reduces as the hop count increases. This is because the available bandwidth decreases as the number of hops increases. The bandwidth for the DIPLOMA is 6.6% lower than the original. This is due to larger headers and processing required for the DIPLOMA.

Figure 6.5 shows the packet loss for the same experiment. The packet losses are less than 1% for both the schemes for the rate of 100 and 300 pkts/sec. The packet losses are higher for the rate of 500 kbps, which explains the lower throughput as the hops count increases. The packet loss is about 5% more for DIPLOMA, due to larger headers, which require more bandwidth.

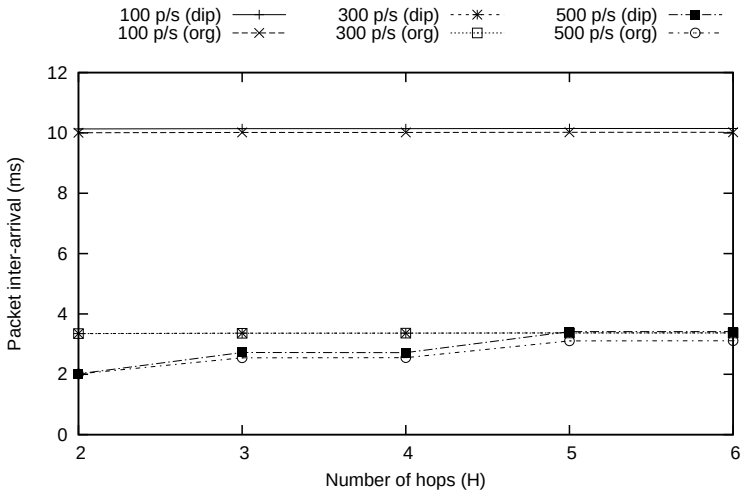

Figure 6.6: Packet inter arrival times for line topology

Figure 6.6 shows the packet inter arrival times for the same experiments. For the rates 100 and 300 pkts/sec, the inter arrival is close to the inverse of their send rate. The inter arrival for diploma is slightly higher than the original, due to larger processing required. For the 500 pkts/sec rate, inter-arrival time increases with hop count, due to correspondingly higher packet loss.

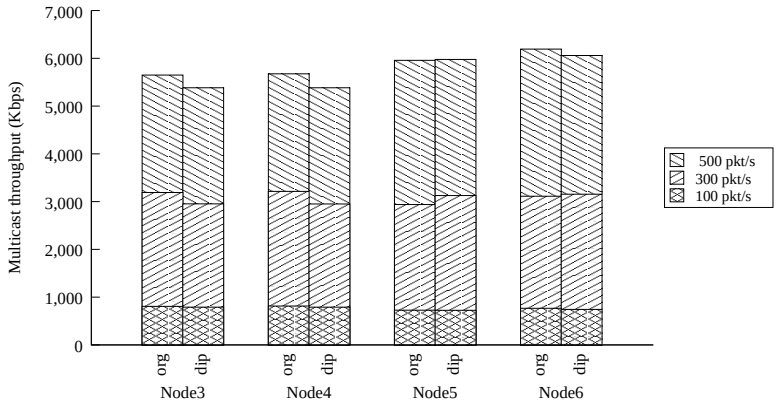

Figure 6.7: Throughput for tree topology

6.4.3 Tree topology

We study the throughput, the packet loss, and the inter arrival times for the tree topology given in Figure 6.3. Here the root node 0 is the sender and the leaf nodes 3,4,5,6 are the receivers.

Figure 6.7 shows the throughput at the nodes for both the schemes. Though the nodes are at same distance from the root, they receive different bandwidths. This may be because of the channel conditions and the packet scheduling. For some nodes, the DIPLOMA receives higher bandwidth than the original. The sum of the bandwidth received by all the four receivers is slightly higher for the original scheme compared to the DIPLOMA scheme. This total bandwidth is 2.1%, 2.1% and 3.6% higher respectively for the rates 100, 300 and 500 pkts/sec for original compared to DIPLOMA.

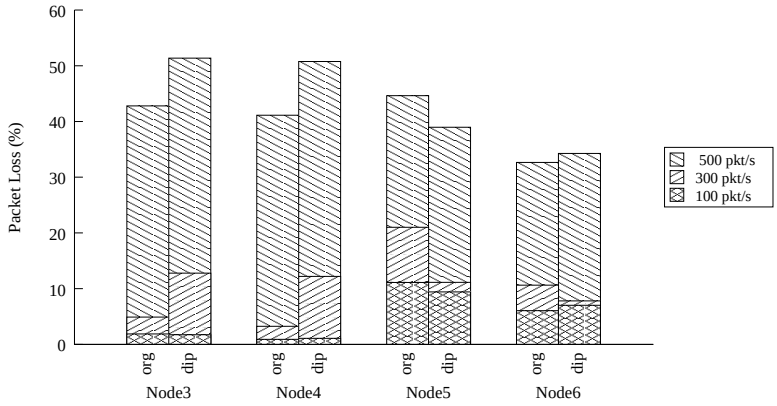

Figure 6.8: Packet loss for tree topology

Figure 6.8 shows the packet loss at the nodes for both the schemes. Unlike the line topology, there was some packet losses (6% to 9%) for the rates of 100 and 300 pkts/sec for both the schemes on some of the nodes. This may also be due to channel conditions.

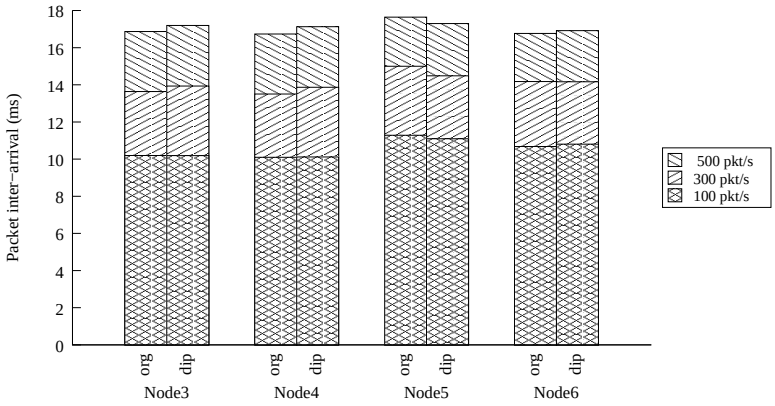

Figure 6.9: Packet inter arrival times for tree topology

Figure 6.9 shows the packet inter arrival times for both the schemes. Here also for some receivers, the inter arrival times were shorter for the DIPLOMA. However, on average, the inter arrival times for the DIPLOMA was slightly higher than the original, due to larger processing delays and the extra headers in DIPLOMA.

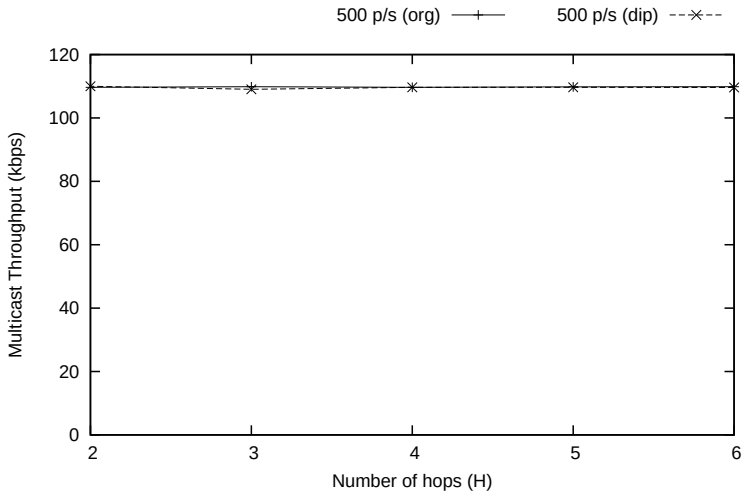

Figure 6.10: Streaming video throughput for line topology

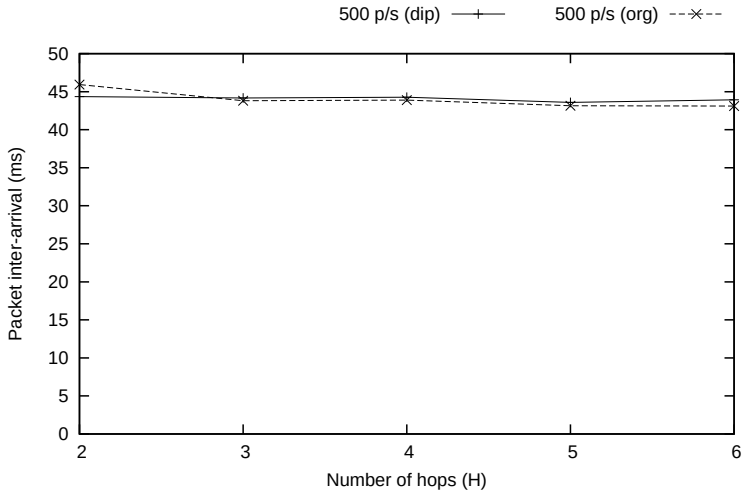

Figure 6.11: Streaming video packet inter arrival times for line topology

6.4.4 Streaming video

In this set of experiments, we study the performance of streaming video. The experiments were conducted by creating a trace of streaming video using *evalvid* [Kla], and sending packets based on that trace using *mgen*.

Figures 6.10 and 6.11 shows the throughput and the inter arrival times for the streaming video for the line topology. The results show that both the DIPLOMA and the original schemes receive the full bandwidth of the video, and the packets are received at constant inter-arrival times.

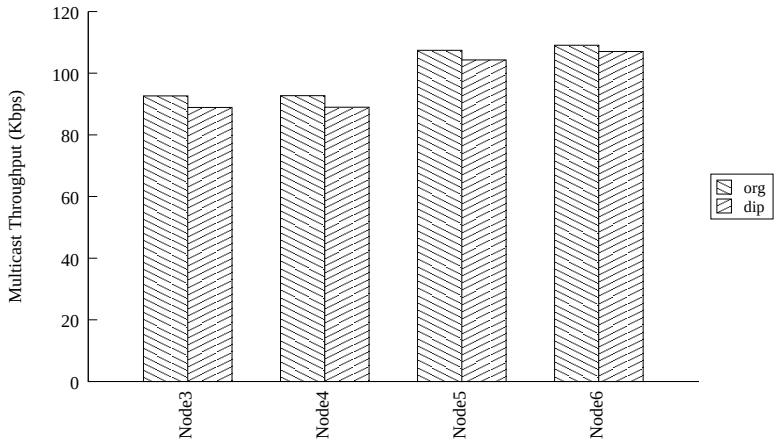

Figure 6.12: Streaming video throughput for the tree topology

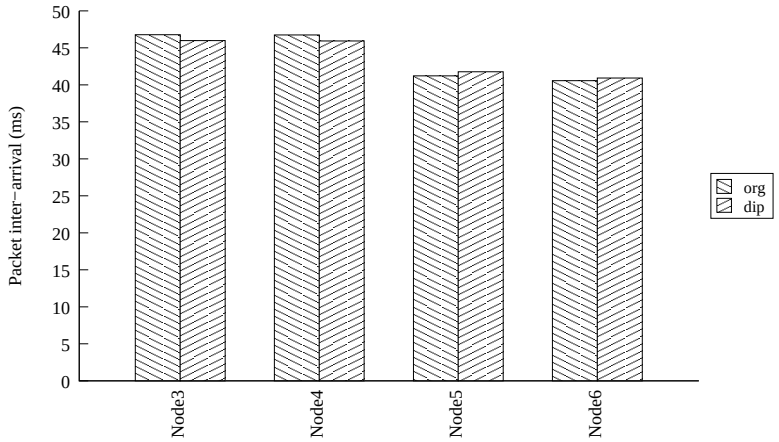

Figure 6.13: Streaming video packet inter arrival times for the tree topology

Figures 6.12 and 6.13 shows the results for the tree topology. There was a small loss in two of the nodes for both the schemes. This behavior is similar to the results for the periodic traffic.

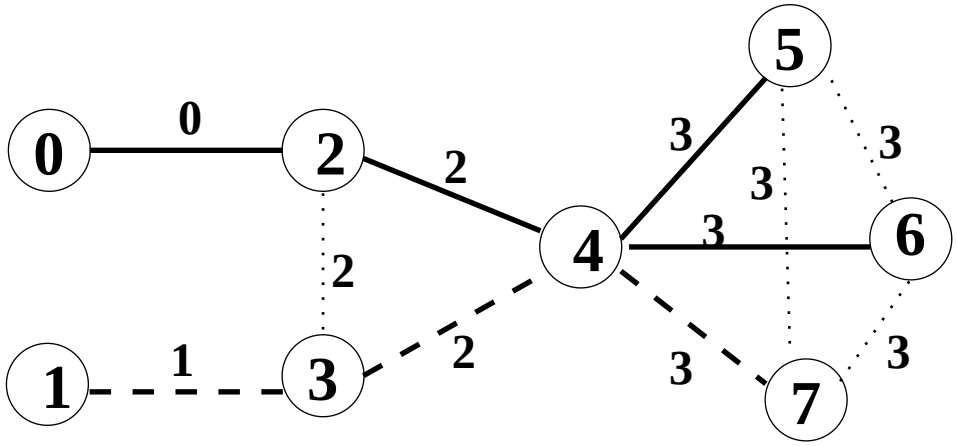

Figure 6.14: Attack topology

6.4.5 Attacker resiliency

Now we study the effectiveness of multicast DIPLOMA in containing attackers. We use the topology given in figure 6.14. The solid lines show the multicast tree and the dashed lines show the unicast path. The labels on the links show the channels. In the experiments below, the nodes 0 and 1 are the senders. These nodes have only its neighboring nodes 2

and 3 respectively in it communication radius. Hence only nodes 2 or 3 cannot be protected by DIPLOMA, when these nodes misbehaves at Physical or MAC layer.

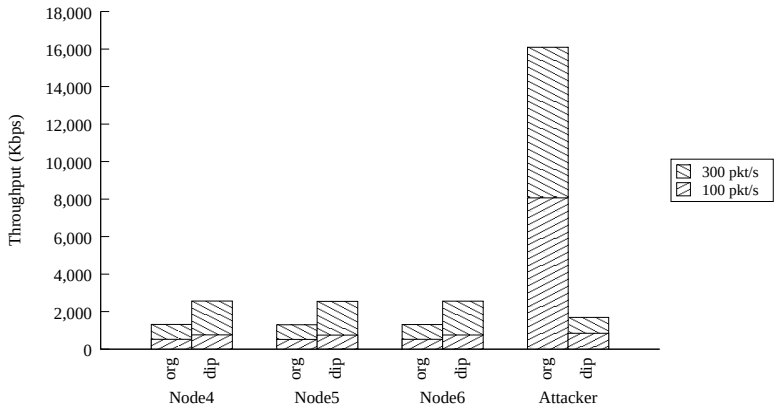

Figure 6.15: Throughput in presence of unicast attacker

We study how DIPLOMA can protect multicast sessions when there is a DoS attacker sending high-rate traffic. Node 1 (attacker) sends periodic traffic of size 1024 bytes at the rate of 1000 packets per second (*i.e.,* rate of 8.19 Mbps) to node 7. The allocated bandwidth for the attacker was 1 Mbps. Simultaneously, node 0 multicasts periodic packets of size 1024, to receiver nodes 4, 5 and 6, at rates 100 pkts/s (*i.e.,* 819.2 Kbps) or 300 pkts/s (*i.e.,* 2.45 Mbps).

Figure 6.15 shows the throughput at the three multicast receivers and the unicast receiver (attack traffic). In DIPLOMA, the attacker is able to achieve a bandwidth of 844 Kbps, which is the allocated bandwidth (minus the overhead). The multicast receivers receive close to their send bandwidth. The multicast receivers receive on average 749 Kbps and 1.80 Mbps respectively for 100 and 300 pkt/s traffic. For the original scheme, the attacker is taking up most of the bandwidth, at 8.04 Mbps. The multicast traffic receives only a fraction of its send bandwidth. The multicast receivers receive on average only 517 kbps and 788 kbps respectively for 100 pkt/s and 300 pkt/s traffic.

6.4.6 Multicast Simulations

In this subsection, we provide the results of the GloMoSim implementation by comparing the performance of our capability-based extension of ODMRP (i.e. DIPLOMA ODMRP)

with that of the unmodified ODMRP. In the DIPLOMA scheme, the Join Query and Join Reply packets contain the multicast capabilities and the associated transaction identifiers and the signatures. Data packets on DIPLOMA also contain the transaction identifiers and the signatures. Like the unicast case in Chapter 4, nodes participating in the multicast verify the signatures probabilistically. The simulations were conducted using the same parameters as described in Section 4.2.

6.4.6.1 Multicast Latency Overhead

To measure the overhead, we use a simple line topology. Node 0 multicasts CBR traffic of 512 bytes at the packet interval of 100ms (i.e. data rate of 40kbps). The even-numbered nodes subscribe to the multicast message. Both the even and the odd numbered nodes act as forwarding group nodes. The latency of the first packet and the average packet latency of 100 packets are shown in Figure 6.16.

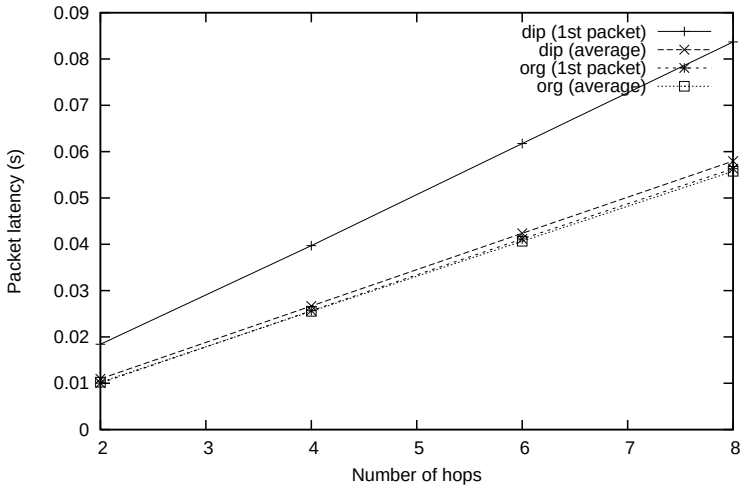

Figure 6.16: Latency of the CBR packet of size 512 bytes for multicast

In the figure, we see that latency of the first packet and the average Packet latency for the original scheme is almost the same. However, the latency of the first packet is higher than the average latency in DIPLOMA. The additional latency in the first packet is about 3.2 ms per hop. This latency is coming from the signature generation and verification time due to the capability establishment as well as the latency due to the increase in packet size because of the additional headers in DIPLOMA. There is no additional latency for the

first packet in the original scheme since the data packet is piggybacked in the Join Query request. In DIPLOMA also, MSC and the first data packet are piggybacked on the Join Query request but the capability processing overhead is not negligible. The average latency of the DIPLOMA scheme is very close to that of the original scheme. The overhead is only 5%, which is mostly due to the signature verification and the larger packet due to the transaction id and the signature fields.

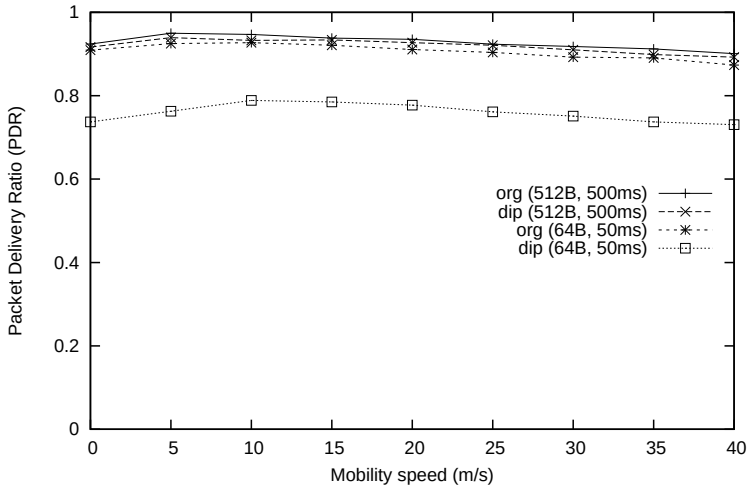

Figure 6.17: Packet delivery ratio for multicast CBR traffic at various mobility speeds

6.4.6.2 Random Topology with Mobility

We compare the performance of multicast traffic for large topologies with mobility. We used random 50 node topologies on a 1200×1200 meter terrain. There were two multicast groups of 10 nodes each. In each multicast group, two source nodes transmitted CBR traffic with packet size of 512 bytes and 64 bytes with the inter-packet interval of $500ms$ and $50ms$ respectively. The nodes were mobile using random waypoint models of various speeds. The pause time in the random waypoint was set to 1 second. Each data point was collected by running the experiments 20 times with different seeds.

Figure 6.17 shows the packet delivery ratio (PDR) for both schemes at inter-packet interval times of $500ms$ and $50ms$ for the packet sizes of 512 bytes and 64 bytes respectively and various mobility speeds. On average, the PDR for the DIPLOMA scheme was only 0.9% lower than the original scheme for 512 bytes packets at $500ms$ interval. For very high packet

rate multicast of 64 byte at $50ms$, the PDR for DIPLOMA was 19.4% lower. This is because of the relatively high overhead for smaller packets in the form of the headers for DIPLOMA.

Chapter 7

Capability Misuse Detection

Consent-based networking is emerging as a "clean-slate" design for providing security against multiple attacks in the Internet [SNW+09; ARW03; AKS09]. In consent-based networking, a sender needs to have permission to send traffic to a destination. Consent-based architectures may support permission to send to a destination on a particular path (*path-based*) or on any path (*destination-based*). In path-based consent architectures, every node (or realm) in the path from a source to a destination need to give consent to send traffic. This gives the nodes control over the traffic passing through them, making it suitable for networks like the Internet, where there are multiple providers (or administrative domains) and the paths are mostly static. In destination-based consent architectures, permission is given to send traffic to a destination on any of the available paths; intermediate nodes honor those permissions and forward the traffic. This architecture is useful for networks where the paths are dynamic, as in mobile ad-hoc networks.

In a consent-based system, senders are given the permission to send traffic in the form of verifiable proofs of consent (capabilities). The nodes perform bandwidth enforcement by rate controlling the bandwidth used for the flows that are part of the capability. In destination-based consent architectures, it is possible to use the capability to reach a destination on multiple paths. In those cases, not all traffic corresponding to a capability may go through a node. Hence, any single node may not be able to enforce the bandwidth constraints of the capability. Furthermore, a node that has authority over multiple destination nodes may assign permission to reach those destinations in a single capability. Hence, the

same capability may be used for multiple unrelated flows. Even if all the traffic passes through a node, the node may be unable to enforce the bandwidth constraints across unrelated flows due to high processing required to account for traffic across the flows. It is also possible for certain nodes to collude with senders allowing for larger bandwidth than the one allocated in the capability. When misuse prevention is not feasible, we need a detection mechanism. Once misuse is detected, the capability may be revoked or temporarily not honored, or the node misusing the capability may be isolated.

DIPLOMA provides two main functionalities: access control of the end host services, and protection of the network bandwidth. The access control is enforced using capabilities, which are cryptographically verifiable entities. All the nodes in the path from a source to a destination can verify the capabilities for the access control. Hence, it is not possible to bypass the access control mechanism under the normal assumptions of cryptography.

In this chapter, we identify the sources of misuse in DIPLOMA and provide solutions for detecting those misuses. A misuse may constitute either the use of a capability in multiple paths to a destination, or the use of the same capability to multiple destinations. The detection of misuse may be done based on the information locally available to the node (local detection), or based on the information exchanged among the nodes (distributed detection).

To provide solutions for detecting misuses, we modify the capability establishment protocol to enable nodes to detect the misuses. We also describe the protocols for communicating the information about the flows going through the nodes to enable distributed detection. We also provide efficient algorithms for detecting misuses.

The node detecting a misuse should be able to provide the proof of the same, so that other nodes can take action based on the misuse. Our solution can provide the proof of the misuse, so that rogue nodes cannot exploit the misuse detection algorithms itself. Our solution also handles privacy issues associated with the exchange of information about the flows.

We implemented our algorithms in the Orbit lab testbed [orb]. We show that the algorithms require minimum processing and memory. We also show that the amount of information exchanged for the misuse detection algorithm is minimal. We also conduct

extensive experiments on capability misuses, and show that our system effectively detects and contains these misuses.

The rest of the chapter is organized as follows. In Section 7.1 we identify the sources of misuse in consent-based systems like DIPLOMA. In Section 7.2 we give the details of the misuse detection architecture, including a capability encoding scheme that facilitates the detection, and the protocols for exchanging information among the nodes for a distributed misuse detection. Section 7.3 gives detailed algorithms for misuse detection and their analysis. We describe the experimental results in Section 7.4.

7.1 Misuse in consent-based systems

In this section, we identify ways of misusing capabilities in destination-based consent systems like DIPLOMA. Some of the misuses are not preventable by forwarding nodes. These includes simultaneous use of a capability on multiple paths to get more than allocated bandwidth, or misusing a policy to create network capabilities more than the policy is entitled to.

There are other misuses in consent-based systems that are directly preventable by the forwarding nodes. For example, if a sender tries to send more than the allocated bandwidth on a capability, the nodes forwarding the packets can detect the misuse and drop the packets. DIPLOMA uses a token bucket algorithm to enforce bandwidth (Chapter 5). In fact, in the DIPLOMA system a node forwards a packet only if the following conditions are satisfied:

1. There is a valid capability associated with the packet.
2. The packet has a valid signature.
3. The packet satisfies the bandwidth constraints of the capability.

Thus, our focus is in detecting the misuses that cannot be prevented by systems like DIPLOMA, due to use of a capability in multiple paths or destinations.

7.1.1 Misuse of policy tokens

Policy tokens are capabilities allocated by the group controllers to the nodes to access the services running on other nodes in MANET. The node for which the policy token is allocated is called *owner* of that policy token. A policy token contains the owner, the destination node, the type of service, the allocated bandwidth, and the signature of the group controller. The destination field of a policy token may correspond to a specific host or a group of hosts. A sender (*i.e.,* the owner) can send traffic to multiple receivers simultaneously using a policy token that has authorization to access those receivers. While accessing multiple receivers, the sender should not exceed the total bandwidth allocated to that policy token. A misbehaving sender may try to exceed this allocation by deliberately communicating with multiple receivers without satisfying the overall bandwidth constraints of the capability. We call this misuse as *concurrent-destination misuse*.

Another way to misuse the capabilities is to use multiple paths to the receiver. The sender may use the same capability on multiple paths, and may claim the bandwidth allocation of the capability in each of the paths. This way the sender can bypass the bandwidth enforcement that is performed by the intermediate nodes. Though the receiver can easily detect this kind misuse, it might be collaborating with the sender to receive a larger bandwidth. We call this misuse as *multi-path misuse*.

7.1.2 Misuses of network capabilities

Network capabilities are the capabilities issued by the receiver nodes to the senders that authorize sending traffic to those receivers. They are similar to policy tokens, except that the destination field cannot be arbitrary; it has to be the receiver that issued the capability. The capabilities also need to contain a signed policy issued by the group controller authorizing the receiver to issue such a capability.

Nodes can misuse a network capability in two ways: either by a receiver issuing more than it is entitled to, or by a sender sending more than the network capability. A sender could misuse the network capability by sending the capability over multiple paths. This is same as the multi-path misuse. Note that concurrent destination misuse is not possible with network capabilities, since those capabilities have fixed destination.

A receiver creating network capabilities may perform another form of misuse. The receiver needs to conform to the policy while creating network capabilities. A policy puts an upper bound on the amount of bandwidth a receiver may allocate to capabilities simultaneously. A receiver might not abide by this policy, and might allocate more network capabilities than it is entitled to. We call this misuse as *policy misuse*.

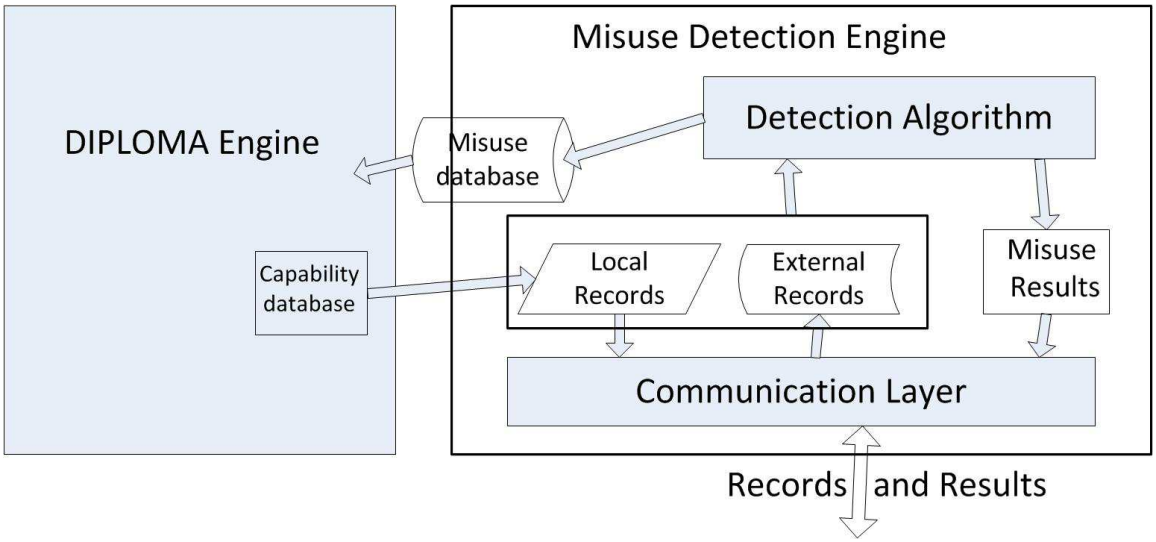

Figure 7.1: Misuse detection architecture

7.2 System Architecture

7.2.1 Misuse Detection Architecture

Figure 7.1 shows the architecture of the misuse detection system in DIPLOMA. The DIPLOMA engine, which is responsible for packet processing and capability enforcement, collects the information about the capabilities going through the node and provides them to the misuse detection engine. This information is stored in the local records table. The detection engine may also receive the information about the communication flows and the associated capabilities from other nodes, which are stored in the external records table. The detection engine periodically runs the misuse detection algorithm described in Section 7.3 on these records. Whenever the algorithm detects a misuse, it informs the local DIPLOMA engine

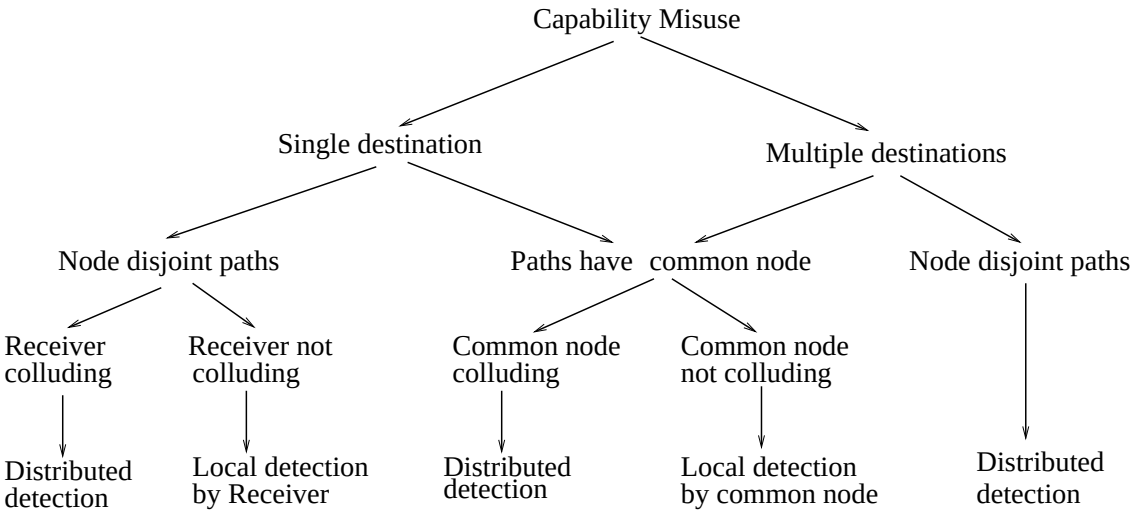

Figure 7.2: Various types of capability reuse and detection algorithm

as well as the misuse detection engines of the other nodes. The DIPLOMA engine makes use of this information while accepting the capabilities for connection establishment and packet forwarding.

Based on where the flow information is obtained, the misuse detection algorithm is classified as local or distributed. The algorithms run for both the local and the distributed detection are the same, except that local detection does not require phase 1 of the algorithm (Section 7.3).

Local detection

In many cases, we can detect capability misuse using the local information a node has, received either through the packets passing through that node, or by listening to the channel and snooping on the packets in its neighborhood. For example, a receiver can detect any misuse by a sender directed towards it. Nodes in the sender's neighborhood may be able to hear all the packets by listening to the channel. In those cases, a neighboring node will be able to detect any misuse by a sender, and provide a proof of the misuse.

Distributed detection

When it is not possible to detect the misuse based on local information, we resort to distributed detection. For example, if the sender is using a directional antenna, then its neighboring nodes may not be able to hear all the packets it has sent. A misuse may be targeted towards multiple receivers; hence, a single receiver cannot detect it. Even if a misuse involves a single receiver, the receiver may be colluding with the sender and may not report the misuse. In distributed detection, the nodes periodically exchange information about the flows passing through them. The misuse detection algorithm is run using the combination of the information a node collected locally and what it received from other nodes. In distributed detection, one or more nodes in the MANET are designated as *verifier nodes*. All the other nodes (called *collector nodes*) send information about the flows going through them to one or more of these verifier nodes. The verifier nodes run the misuse detection algorithm, and inform the collector nodes about any misbehaving nodes and the associated capabilities, along with proof of misuse.

Figure 7.2 illustrates the types of the detection methods that are useful in various misuse scenarios. Note that, a distributed detection algorithm can always detect the misuses that a local detection algorithm can detect. However, vice versa is not true. Hence, in the figure the local detection has priority over the distributed detection. In addition, the distributed algorithm will fail if all the nodes in a path collude with the sender. We ignore that fact in the figure and assume that at least one node in each of the path will co-operate with detection.

Whether distributed detection is required or not is dictated by whether there exists any common node in the misuse paths, and whether the common node is willing to co-operate. A common node can run the detection algorithm based on the local information alone to identify, and report the misuse. For the misuse involving multiple paths to a single destination, the receiver is always a common node. If all the common nodes are colluding with the sender, then a distributed detection is required.

7.2.2 Misuse Detection Components

A detection solution needs to have the following properties:

1. The nodes are required to store only minimal amount of information.
2. The amount of information that need to be communicated is minimum.
3. The algorithms to detect the misuse should be fast.
4. There should be a verifiable proof about the misuse.

The first three requirements are attributed to the limited resources available on the MANET nodes in terms of memory, bandwidth, and processing power. The last requirement is to avoid any misuse of the misuse algorithms itself. The nodes detecting misuse should be able to prove the rest of the nodes that a misuse had occurred, so that action can be taken against the misbehaving node.

Our solution for misuse detection has the following components:

1. An encoding scheme for capability requests and network capabilities, so that the detector nodes can easily extract verifiable information for misuse detection in an efficient manner.
2. Protocols for exchanging relevant information that aid in misuse detection. This also includes identifying the nodes to which the information is sent.
3. Efficient algorithms for the misuse detection.

Next, we describe the encoding scheme for the capabilities and the protocols for exchanging relevant information. We will describe the algorithms in the next section.

7.2.3 Capability encoding

In the DIPLOMA architecture, it is permissible to use a capability for multiple communication sessions concurrently. For example, a node possessing a policy token to communicate with a group of destinations may be simultaneously communicating with multiple nodes in that destination group. Similarly, it is possible to use a policy authorizing the issue of the network capability to create multiple capabilities simultaneously. For example, a node may be receiving packets from multiple source nodes, and may want to allocate network

capabilities to those senders based on a single policy authorizing the allocations. Both of these concurrent uses of policies are valid as long as the nodes do not use (or allocate) more bandwidth than allowed by the policies. When the nodes split the bandwidth of a policy into multiple capabilities, we need protocols that enable other nodes to check if these capabilities are within the limit. We use the term *owner* of a policy to denote the nodes using or creating these capabilities.

While sending the policy token or while creating receiver capabilities, the owner has to decide on how to split the available bandwidth. The protocol allows dividing the available bandwidth into 32 or 64 equal sized slots, which are represented using a bitmask of 32 or 64 bits. We call this bitmask as the *allocation vector*. A bit in the allocation vector is set, if the corresponding bandwidth slot is used. The allocation vector is included in the capability request packets, as well as on the network capabilities created by the receivers. For a policy token that a sender is using to communicate with multiple destinations, the allocation vector on a capability request indicates the portion of the available bandwidth allocated to that communication session. When a sender uses multiple paths to reach a destination, the allocation vectors on the capability requests on each of the paths indicate the portion of the available bandwidth from the capability (*i.e.,* policy token or the network capability) allocated to that path. If a receiver node creates multiple network capabilities, based on a policy, the allocation vector field in the capability indicates the portion of the available bandwidth allocated to that capability. Note that there could be multiple bits set in the allocation vector indicating bandwidth allocation proportional to number of bits set in the vector. The presence of capability requests for the same capability with a common bit set in their allocation vectors indicates that the sender is trying to misuse the capability. Similarly, the presence of two network capabilities for the same policy that has a common bit set in the allocation vector indicates a misuse by the receiver issuing that capability. Larger number of bits in an allocation vector gives finer granularity on the allocated bandwidth. It also allows for larger number of simultaneous flows associated with a capability or policy. However, larger number of bits in an allocation vector also leads to higher bandwidth and memory overhead. In our experiments, we use allocation vectors of size 32.

It is permissible to allocate the same bandwidth slot to different capabilities, derived

from the same policy, at different times. Every capability request and network capabilities contain a start time stamp and a validity duration, which indicates the time until they is valid. To extend its validity, the owner needs to create a new request. Hence, a misuse constitute the existence of two capability requests for the same capability, or two network capabilities for the same policy that has a common bit set in their allocation vector at the same time.

Our misuse detection algorithms do not depend on the data structure used for dividing the allocated bandwidth. Allocation vectors have easy representation, and allow for easy unions and intersection operations using bitwise operators. If finer granularity is needed in dividing the bandwidth, one could use other representations like slab allocation.

7.2.4 Communication protocol

Now we describe the modifications to the original DIPLOMA protocol to enable misuse detection. We define a modified capability request format that enables nodes to store minimal information for misuse detection as well as to provide proof of misuse.

When a sender wants to communicate with a receiver, it uses a *capability request* packet to inform the intermediate nodes about the capability that will be used for the communication.

In the original DIPLOMA protocol described in Chapter 3, a capability request contained the capability, the transaction identifier used for subsequent packets, and the keys for the packet signature. The sender signs the request with its private keys. To enable the misuse detection, we add the allocation vector, the start time stamp, and the validity period to this request. One straightforward way is to include these additional fields before signing the capability request. Unfortunately, this requires the nodes to store the complete capability request to provide the proof of misuse, if a misuse has occurred. This is because a proof of misuse needs to contain information that the sender cannot deny, which is any data the sender has signed. A capability request packet has information that is not essential for misuse detection. This includes the capability and the keys for the packet signature. A complete capability is not required, because its sequence number and the issuer identity can uniquely identify it. An alternative solution is for the sender to sign the information needed

to detect the misuse separately from rest of the information. This is memory efficient, because the nodes need to retain only essential information. Unfortunately, this approach requires the sender to perform two expensive signature operations.

As a compromise between the amount of information stored by the nodes and the processing needed at the sender, the sender signs the capability request as two steps. First, the sender computes the hash of the part of the capability request that is not essential for misuse detection, including the capability and the keys for packet signature (called *capability establishment block*). Then, it appends the hash with the information essential for misuse detection and sign the resulting block (called *misuse detection block*). This block includes the transaction identifier, the serial number and the issuer of the capability, the time stamp and the validity of the request, and the allocation vector. To prove the misuse of a capability by a sender, the nodes only need to keep track of the misuse detection block and the signature.

A capability request packet contains the capability establishment block and misuse detection block. A node can verify the capability request by first computing the hash of the capability establishment block and comparing with the one present in the misuse detection block. If the hash matches, then it verifies the signature of the misuse block. A valid signature on the misuse detection block indicates that there was no tampering of the packet. When the sender wants to use same capability on multiple paths, it is required to use different transaction identifiers and different allocation vectors.

We use a similar mechanism for signing the network capability generated by a receiver node. The network capability has two parts. First part consists of the sender identity and the policy. The second part consists of the unique serial number generated for the capability, the serial number and the issuer of the policy, the time stamp and the validity, and the allocation vector. The receiver computes the signature for the hash of the first part and entire second part.

Information exchange

For distributed detection, the detector nodes send information about the communication sessions and the capabilities passing through them to verifier nodes. For local detection,

the DIPLOMA engine provides the same information to the misuse detection engine. The information required to detect the non-conformant, simultaneous use of capabilities by a misbehaving sender consists of:

1. The identity of the sender node
2. The transaction identifier for the communication
3. The serial number of the capability
4. The issuer of the capability
5. The allocation vector
6. The start time stamp and the validity duration
7. The next and previous hops for the communication session.

The above information about the communication session is called a *record*. A node can send multiple records in a packet. The nodes sign the packet using their private keys. For detecting misuse of the policy authorizing the creation of network capability by the receivers, nodes exchange the following information about the network capabilities passing through them:

1. The identity of receiver that created the network capability
2. The serial number of the network capability
3. The serial number of the policy
4. The issuer of the policy (i.e. GC)
5. The allocation vector
6. The start time stamp and the validity duration

The algorithms used for detection of the misuse by the senders, and the receivers are similar. Hence, we will deal only with the sender misuse in rest of the chapter.

7.3 Detection Algorithms

In this Section, we describe the DIPLOMA misuse detection algorithms that are used for both local and distributed detection. Then we describe how a verifier node can provide a proof of misuse. Finally, we provide solution for handling the privacy issues in our misuse detection architecture.

Recall that there is a misuse if there are two communication sessions that use the same capability and have a common bit set in the allocation vectors with overlapping validity periods. Hence, the goal of the algorithm is to find such communication sessions. To that end, the algorithm first groups the records corresponding to a communication session. This is because there could be multiple records for a communication, received from different collector nodes. Once the records of communication sessions are grouped together, the algorithm look at records across the communication sessions to identify misuse.

In DIPLOMA, a communication session can be uniquely identified by the (transaction identifier, sender identity) pair. If the sender uses multiple paths to a destination, the sender is required to use different transaction identifiers for each path.

The misuse detection algorithm has two phases. In the first phase, it removes the duplicate records for each communication sessions from the collection of records it gathered locally and from other nodes. It also detects if a sender uses the same transaction identifier on multiple paths. The output of the first phase is a set of records, consisting of at most one record for a transaction identifier per sender. This phase is not required if all the records are obtained from the capability database of the local DIPLOMA engine, as the engine already prevents duplicates. In the second phase, the algorithm detects if there is any misuse on the filtered records output by the first phase.

7.3.1 Phase 1 - Duplicate removal and multipath detection

The goal of first phase is to remove duplicate records of communication sessions and to verify that the sender does not use the same transaction identifier in multiple communication sessions, including multiple paths to a destination. This phase is only required for distributed detection.

The removal of duplicate records for a transaction identifier is performed by sorting the records based on (sender, transaction id) pair and keeping only one record per pair. However, this step will not detect use of the same transaction identifier by a sender on multiple paths. To detect the multiple path misuse, we use the following property of the paths.

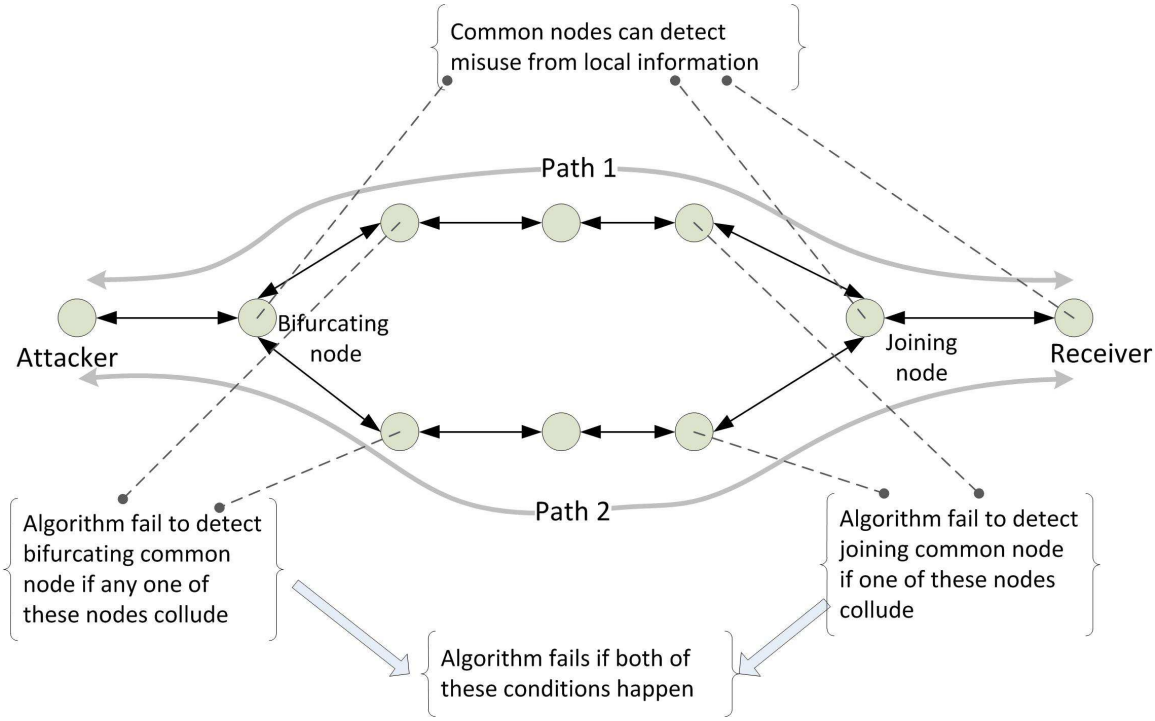

Figure 7.3: Properties of multiple paths in aiding misuse detection

If the same transaction id is used in two different paths to a destination, then there will be two records for it that has a common previous node or a common next node. This is because since the source and the destination nodes are common in both paths, the paths need to bifurcate at some node and join at another node. If the paths are bifurcating at any node other than the sender, or joins anywhere other than the receiver, then the DIPLOMA engine at the common nodes in the path can detect the misuse during the connection establishment stage. If the paths are node disjoint and the receiver is not colluding with the misbehaving sender, then the receiver can detect misuse. If all the common nodes are colluding with the sender, then the phase 1 algorithm can detect the misuse looking for the

common nodes. This is depicted in Figure 7.3.

Algorithm 1 describes the phase 1 algorithm. It goes through the records corresponding to the same (sender, transaction id) pair and verifies that all the records use the same capability, allocation vector and time stamps. It also stores the previous nodes and the next nodes of each record in temporary arrays. The presence of duplicates in these arrays indicates a misuse.

Analysis

The algorithm will fail to detect a multipath misuse if certain nodes collude with the sender and do not provide the relevant records to the verifier. If the common nodes, including the receiver, collude with the sender, then local detection of the multipath misuse will fail. If at least one of the nodes in the path next to the common node colludes, where the forking of the paths has occurred, then the algorithm will fail to detect that common node. Similarly if one of the nodes before the common node at which the joining of the paths take place, then also the algorithm will fail to detect the common next node. The algorithm will fail to detect a multipath to a destination, when it cannot detect both the common previous and next nodes. This is depicted in Figure 7.3. It is still possible for a verifier to detect that it has not received records from some of the nodes (which may be colluding with the sender), because of the existence of two path fragments (as opposed to one path) for the transaction identifier. However, we cannot use this against the sender, because of the possibility of packet losses, or the possibility of a node deliberately not sending the records to the verifier.

Running time

The algorithm sorts the lists of records based on the (sender, transaction id) pair, which can take $O(n \log n)$ time, where n is the number of records. Then it goes through each record to remove duplicates. Typically, the path lengths in MANETs are small constants (less than 6 hops). Hence, the operations on the arrays maintaining previous nodes and next nodes can be performed in constant time. If the path lengths are large, we can use more efficient data structures like heap or hash tables, without increasing the overall asymptotic complexity

of the algorithm. Other operations inside the *for loop* at line 3 can also be performed in constant time. Hence, the lines from 3 to 28 can be performed in $O(n)$ time. So the running time of phase 1 algorithm is $O(n \log n)$.

7.3.2 Phase 2 - Reuse of the capability detection

The second phase, the algorithm detects misuse of capabilities across the communication sessions. Recall that a misuse is identified by a common bit set in the allocation vectors at overlapping validity periods. The input to phase 2 is the records output by phase 1. Hence, there is only one record per communication session. The algorithm goes through all the records corresponding to each of the capabilities and detects misuse.

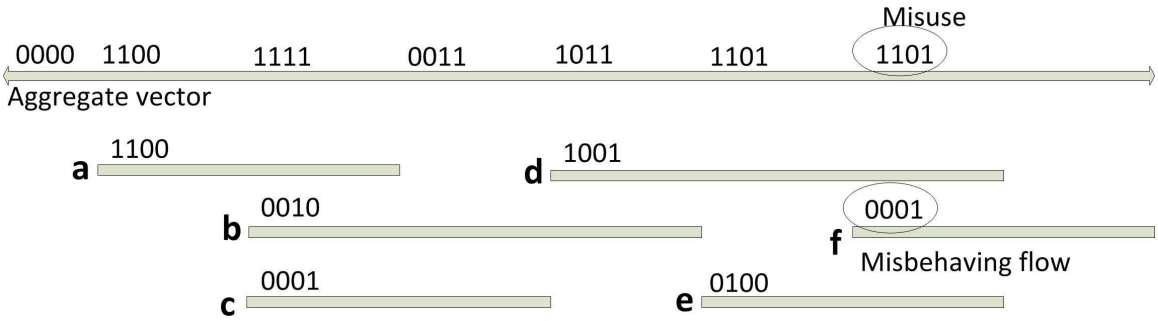

Figure 7.4: Computation of aggregate allocation vector and misuse detection using interval graphs.

The algorithm treats the records as an interval graph, where each record corresponds to an interval for which they are active. There is an allocation vector associated with each interval, which is the allocation vector of the corresponding record. We define the *aggregate allocation vector* at any point of time as the union of the allocation vectors of the intervals passing through it. There is misuse at any point in time if the intersection of any two intervals passing through it is not empty. This is depicted in Figure 7.4.

Once the interval graph is formed, we can detect any misuse in linear time in the number of intervals. The algorithm goes through the end points of the intervals in increasing time and updates the aggregate allocation vector. At the beginning of the algorithm, this vector is set to NULL. Whenever the algorithm considers the beginning of an interval, it checks if

Algorithm 1 Duplicate-record removal & multi-path detection

```
1: L_i ← List of all (source node, transaction id) pair
2: L_u ← NULL {Output list of unique records}
3: for all id ∈ L_i do
4:    L_r ← List of records for id
5:    H_prev ← NULL
6:    H_next ← NULL
7:    for all rec ∈ L_r do
8:       if rec = head(L_r) then
9:          cap ← capability(rec)
10:         time ← timestamp(rec)
11:         alloc ← allocvector(rec)
12:         Add rec to L_u
13:      else
14:         if cap ≠ capability(rec) then
15:            print Misuse. Different Capabilities for a transaction.
16:         else if time ≠ timestamp(rec) then
17:            print Misuse. Different time stamps for a transaction.
18:         else if alloc ≠ allocvector(rec) then
19:            print Misuse. Different allocation vectors for a transaction.
20:         end if
21:      end if
22:      if prevhop(rec) ∈ H_prev or nexthop(rec) ∈ H_next then
23:         Misuse. Same transaction in different paths
24:      end if
25:      Add prevhop(rec) to H_prev
26:      Add nexthop(rec) to H_next
27:   end for
28: end for
29: return L_u
```

the intersection of the aggregate allocation vector and the allocation vector of that interval is non-empty. If it is not empty, then there is misuse. Otherwise, the allocation vector of that interval is added to the aggregate vector. Similarly, when the algorithm considers the end of an interval, its allocation vector is subtracted from the aggregate allocation vector. If there are both entering and leaving intervals at any point, then the leaving operation is considered before the entering operation. This is because the new entering interval could use slots from the leaving interval, without causing misuse.

Figure 7.4 illustrates the computation of aggregate allocation vector and the misuse detection. There are six flows labeled as $a, b, \ldots, f$. They are represented as the intervals in which they are active. Their corresponding allocation vectors are also shown. For simplicity of illustration, we use the allocation vector of 4 bits. The vector on the top line shows the aggregate allocation vector when the flows enter or leave the system. The aggregate vector at any point is the union of the allocation vectors of the interval going through that point. There is no misuse for flows $a, b, \ldots, e$. The flow f uses one of the slots of flow d, hence there is misuse. The aggregate allocation vector before f entered the system was 1101. The flow f uses one bandwidth slot. The sender assigned it the slot 0001, which is a reuse of the existing slot. If the sender had assigned it the slot 0010, then there would not be any misuse. Hence, it is important that senders allocate the right bandwidth slot for flows.

Creating an interval graph from the records is performed by sorting the endpoints of the interval. In fact, our algorithm maintains two sorted lists: one for the starting points of the intervals and the second for the ending points of the intervals. The algorithm is given in Algorithm 2.

Running time

Creating an interval graph from the records can be done in $O(n \log n)$ where n is the number of intervals. Since we are representing allocation vectors as bitmaps, the set operations on the allocation vector is performed in constant time. The union of two allocation vector is the logical OR operation, and the intersection is the logical AND operation. The set difference is found by taking the logical AND of minuend with ones complement of subtrahend. Once the intervals are sorted based on the start time and end time, the algorithm goes through

each of these intervals twice: one each for the start time and the end time. Hence, the misuse detection part of the algorithm is linear in number of intervals. Hence, the total running time of the algorithm is $O(n \log n)$.

Algorithm 2 Phase 2 - checking for reuse of bandwidth slots

1: $L_i \leftarrow$ List of all (capability id, issuer) pair
2: **for all** $id \in L_i$ **do**
3: $L_r \leftarrow$ List of records for id
4: $L_s \leftarrow$ Records in L_r sorted on start time
5: $L_e \leftarrow$ Records in L_r sorted on end time
6: $aggregate \leftarrow \phi$
7: **while** L_s not empty **do**
8: $time_s \leftarrow starttime(head(L_s))$
9: $time_e \leftarrow endtime(head(L_e))$
10: **if** $time_e \leq time_s$ **then**
11: $rec \leftarrow head(L_e)$
12: $L_e \leftarrow L_e - rec$
13: $aggregate \leftarrow aggregate - allocvector(rec)$
14: **else**
15: $rec \leftarrow head(L_s)$
16: $L_s \leftarrow L_s - rec$
17: **if** $aggregate \cap allocvector(rec) \neq NULL$ **then**
18: **print** Misuse. Reuse of bandwidth slots.
19: **end if**
20: $aggregate \leftarrow aggregate \cup allocvector(rec)$
21: **end if**
22: **end while**
23: **end for**

7.3.3 Proof of misuse

Once the algorithm detects misuse, the node detecting the misuse (i.e. verifier) communicates about the misuse to other nodes. The nodes add the capability in the misuse database, and will not honor any requests based on that capability.

In order for nodes to accept that a misuse has occurred, the node detecting the misuse needs to provide a proof for the misuse. Otherwise, a rogue node can launch attacks on legitimate nodes by falsely claiming that a misuse has occurred. The proof of the misuse is obtained as follows. The detection algorithm identifies the records that constitute the misuse. Then the verifier requests that the nodes that sent those records send the misuse detection block corresponding to those records. The detectors have signed the packets containing those records; hence, they cannot deny the records. The sender, which was the source of the misuse, cannot deny about the misuse detection block, because the block has the sender's signature on it. Hence, the proof of the misuse consists of the misuse detection blocks from the capability request packets corresponding to the records identified by the algorithm.

7.3.4 Privacy issues

In the protocol presented so far, the detector nodes send the information about all the flows going through them to the verifier. Even though the records contain only the sender node identity and does not have the receiver identities of the flow, it is still possible to deduce the receiver identity by following the path using the previous and next hop information. Hence, the verifier can know about the source and destination of all the flows. Another privacy concern is the knowledge about the number of flows a sender is sending, even if the verifier is not interested in knowing the receivers.

We can modify the protocol to honor privacy, and still detect the misuse as follows. The detection algorithm continues to function even if all the fields in the records, except the time stamps and the allocation vector, were encrypted with a key that is common across all the flows corresponding to a capability. The detector nodes can create such a key by taking a known function (*e.g.,* hash) of the capability. Since the flows are going through the detector nodes, they know about the complete capability associated with the flow. However, verifier

knows only about their serial numbers, and cannot recreate the keys. Hence, verifier cannot decrypt the records for the flows not going through it. In this scheme, the verifier can still get information about all the flows that use any of the capabilities passing through it.

The proof of misuse also continues to work in this privacy preserving distributed misuse detection. There are two approaches to privacy when misuse is detected. In the first approach, one could argue that the privacy need not be honored for a node launching misuse attack on the system. In that case, the detector nodes having the proof misuse sends those proofs to the verifier, which then distributes those proofs to all the nodes that has a communication session with the misused capability going through them. An alternative approach is for those proofs itself to be encrypted with the encryption keys of the capability before sending it to the verifier. The verifier simply sends the proofs to the detectors that have the communication sessions going through them for the misused capability. The same arguments as the previous subsection, about the non-deniability of the proofs by the nodes possessing the proof of misuse apply here.

7.4 Experimental evaluation

In this section, we study the effectiveness of the misuse detection algorithms. The experiments were conducted in the Orbit Lab Testbed [orb]. The algorithms were implemented on DIPLOMA systems running on Debian Linux with kernel 2.6.30 (Chapter 5). First, we show that the algorithm incurs only minimal overhead in terms of processing capacity and memory requirements. Then, we measure the additional bandwidth required for distributed misuse detection.

7.4.1 Running time of the algorithms

Recall that the running time of the algorithm is asymptotically bound by the time to sort the misuse records. The algorithm performs two sets of sort operations. First, all the records are sorted in phase 1 based on the sender and the transaction identifier to remove the duplicates and check for multi-path misuse for the same transaction. Second, the records are sorted in phase 2 based on the capability issuer and the serial number for detecting the

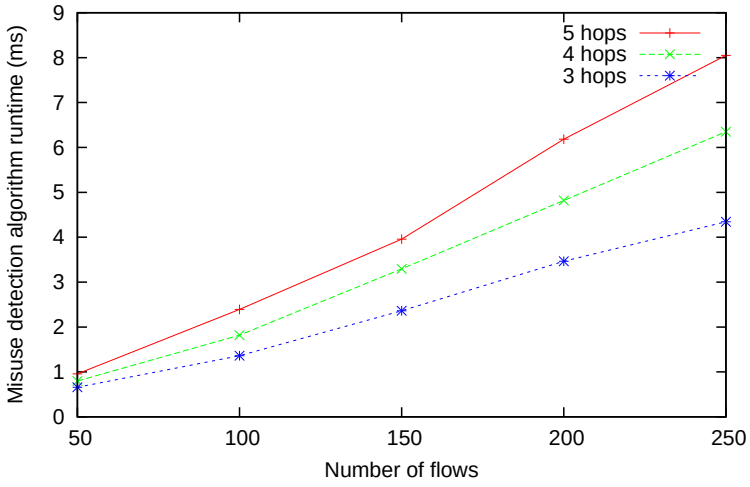

Figure 7.5: Running time of the algorithm

misuse of the same capability to different destinations or transaction identifiers. We use *quick sort* from the standard C library for these sorting. There are two additional sorting of the records belonging to same capability in phase 2 based on the start and end times of the capability requests. The number of records for this operation is expected to be small on normal running systems; hence, this sort operation is expected to be fast.

Figure 7.5 shows the running time of the algorithm against the number of flows for paths of different hop lengths. The label on the plot indicates the path lengths. In this set of experiments, the records were generated at random on a per path basis. Each flow had twice the path-length records, since the flows were bidirectional. Hence the algorithm processed $2 \times flows \times path\ length$ records. There were 20 nodes in the network, and there were four nodes capable of issuing capabilities. The experiments were run 20 times, and the average is reported. Figure shows close to linear time processing in number of flows. This is in spite of the $O(n \log n)$ processing time of the quick sort. This is because of the relatively smaller size of the input for the sort algorithm (i.e. in 100s), and the fact that the rest of the algorithm in phase 1 and phase 2 require linear time processing. The time required for the misuse detection is between 1 millisecond and 8 milliseconds. This is very low overhead, since the algorithm is expected to be executed only every couple of minutes. As expected the time taken for executing the algorithm increases as the number of flows or the path-length increases. This is because the number of records input to the phase 1

increases in both these cases. Furthermore, the number of records input to the phase 2 increases with the number of flows. However, the number of records in phase 2 remains the same when the path length increases, since phase 1 eliminates the duplicate records for a flow.

7.4.2 Memory requirements

Now we analyze the memory requirements of the misuse detection algorithm. Apart from the program needed to detect the misuse, the nodes send one record per flow going through it. This record contains the identifier of the node sending the record, the serial number and the issuer of the capability, the start time and the validity duration of the communication session, the transaction identifier and the owner of the capability, the allocation vector, and the previous and the next nodes in the path. Each of these fields takes 4 bytes, resulting in 40 bytes records. Hence, the nodes performing verification require a memory of $2 \times flows \times path-length \times 40$ bytes to store these information. The sorting of these records for phase 1 can be done in place and does not require additional storage. For phase 2 of the algorithm, only one record per flow is required. This can be achieved either by creating a copy of the required records, or by creating a list of indexes pointing to the original records. In the latter case, we need $2 \times flows \times 4$ bytes of storage. The phase 2 also sorts the records for the same capability, based on the start and end times. The number of records for this sorting will depend on the number of times a node use the capability for different flows in the given time period. Here also, the verifier node only needs to keep pointers to the actual records.

The detector nodes has to sign the packets containing the records, so that the they cannot deny about the records once the verifier detects a misuse and requests for proof. These signatures are based on the public keys of the detector nodes. The signature will take 128 bytes for 1024 RSA keys. The headers take 28 bytes for IP/UDP and 8 bytes for the capability protocol. Hence a 1500 byte packet can carry $\lfloor \frac{1500-28-8-128}{40} \rfloor = 33$ records. For providing the proof of misuse, the verifier node may need to keep track of these packets, except for the headers. If the signature is performed on per packet basis, and if there are f_i unidirectional flows going through the detector node i, then the detector will send $\lceil \frac{f_i}{33} \rceil$

packets to the verifier. If there are d detectors, and $f = \sum_{i=0}^{d} f_i$ uni-directional flows, then the verifier would have to maintain $\sum_{i}^{d} \lceil \frac{f_i}{33} \rceil$ signatures. This value is upper bounded by $d + \frac{f}{33}$.

7.4.3 Bandwidth requirements

Now we study the bandwidth requirement for the misuse detection. The amount of bandwidth required for detection depends on how fast one would like to detect a misuse. Periodically, the detectors send records about all the flows going through them to the verifier. If the period for sending the records by the detectors is large, then the time to detect the misuse will also be large. This is because the verifier needs to get the record after the misuse has started, for detecting it. On average, the misuse will get undetected for half the period. In this experiment, we study the amount of bandwidth used for sending the records before the misuse is detected, as a function of period at which the records are transmitted.

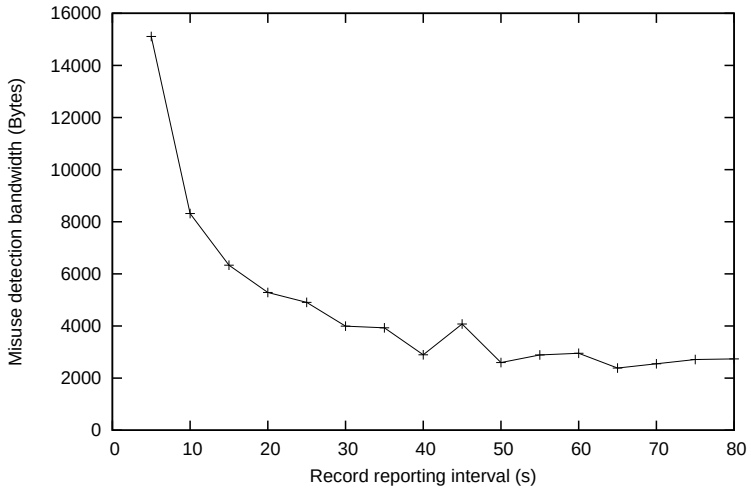

Figure 7.6: Number of bytes used to send the records before misuse detection

We use the topology given in Figure 7.7 in our experiments. The topology was created by filtering out unwanted links based on the MAC address using *iptables*. There are eight nodes in the topology. Node 8 is designated as the verifier. At time 0 seconds, node 2 and node 3 starts sending traffic to the nodes 5 and 6 respectively. Node 0 is the misbehaving node, which uses the same capability to send traffic to two destination nodes 4 and 7 at times 30 seconds and 60 seconds respectively. Hence, the misbehavior starts at time 60

seconds. All the nodes (except node 0) send the records to the verifier node 8 periodically. We measure the amount of traffic sent by the detector nodes, before the verifier detects the misuse. We ran the experiments five times and the reported the average.

Figure 7.6 shows the number of bytes sent to the verifier for various record reporting periods before the first misuse is detected. The time to detect the misuse is within the period at which the records are send. As shown in the graph, the number of bytes required decreases as the period increases. The graph does not show the continuous decrease. This might be due to packet losses that may be happening to the records packets, as well as the timing variations of the records arriving at the verifier. The verifier runs the detection algorithm periodically. If a record arriving at the verifier misses the period, then the detection will occur only in the next period. The plot also shows that it took only between 2700 bytes and 4075 bytes for the 4 flows, to detect the misuse in periods 30 seconds or more.

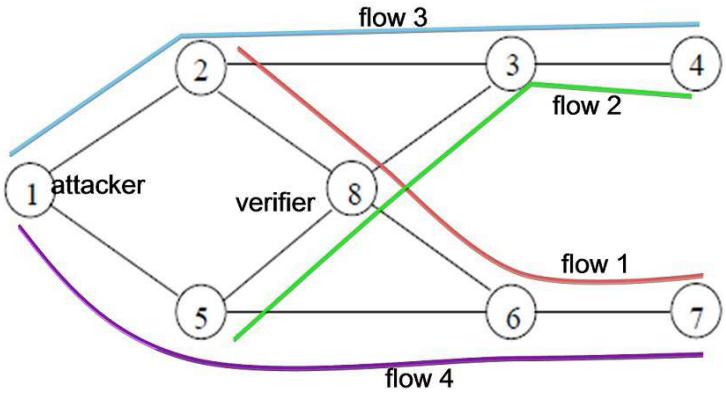

Figure 7.7: Topology to study the performance of detection algorithm

7.4.4 Effectiveness in Containing attacks

Now we study the effectiveness of misuse detection algorithm in detecting and containing the attacks. We used the topology given in Figure 7.7, created by assigning non-overlapping channels of 802.11b and 802.11a to the links as described in Chapter 5. When there are more than two links incident on a node, some of the links were assigned the same channel. This assignment is done in such a way that, the links with the attacker node 1 is not shared

with other nodes.

In this set of experiments, there are four flows: two by good nodes and two by the attacker. Each of the senders is allocated a capability of 4 Mbps each. All the flows were created using UDP *iperf*. The good nodes, 2 and 5, send traffic at 4 Mbps each to the destinations nodes 7 and node 4 respectively. We denote those flows as flow 1 and flow 2 respectively and call them good flows. Flow 1 takes the path $2-8-6-7$ and flow 2 takes the path $5-8-3-4$. These flows are started at time 0 seconds and last for 120 seconds. The attacker, which is the node 1, sends flows to nodes 4 and 7, which we denote by flows 3 and 4. We call these attack flows. Each of the attack flows are 12 Mbps, even though the attacker has one capability with the allocated bandwidth of 4 Mbps, which can be used for either destination. Flow 3 is started at time 30 seconds and takes the path $1-2-3-4$. Flow 4 is started at time 60 seconds and takes the path $1-5-6-7$. Both the flows last for 120 seconds, and use the same capability with all the bits in the allocation vector set. Hence, the attacker launches two types of attacks. First, it is sending higher bandwidth than that is allocated in the capability, which starts at time 30 seconds. Secondly, it uses the same capability to talk to multiple destinations simultaneously using the same bandwidth slot. This attack starts at time 60 seconds. As we will see from the experiments, the DIPLOMA without misuse detection can handle only the first attack, and the DIPLOMA with misuse detection can handle both the attacks.

We conduct three sets of experiments. The first is called the *original*, and does not require any consent for sending the traffic. This scheme cannot protect against both the attacks. Then we use the consent-based scheme, where DIPLOMA requires capabilities for sending traffic. This DIPLOMA without misuse detection, can handle the first bandwidth hogging attack but cannot prevent reuse of the capability across the flows. Finally, we use DIPLOMA with misuse detection to handle both types of attacks. For each of the experiments, we report the bandwidth of each of the flow over a period of time. All experiments were run 6 times, and we show the average bandwidth. The iperf servers (receivers) measured the bandwidths at 5 second intervals.

Figure 7.8 shows the results for the original system. In this system, until 30 seconds, where there are only two flows from the good nodes, both the flows get their requested 4

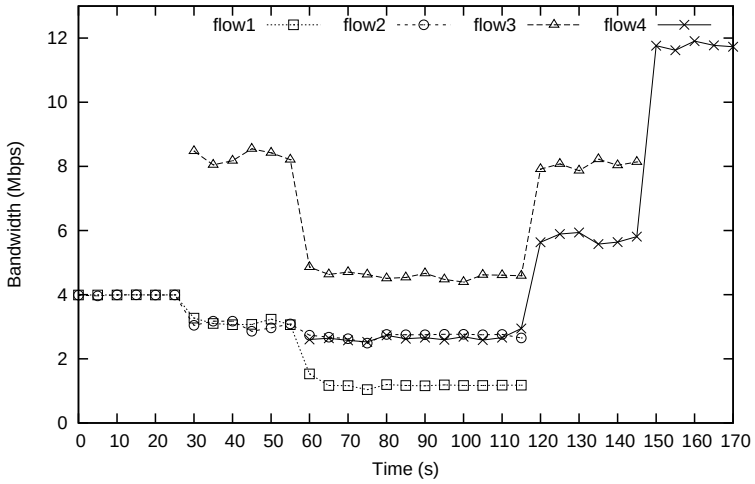

Figure 7.8: Bandwidth of flows in a system that does not require consent to send

Mbps bandwidth. At 30 seconds, when the first attacker flow (flow 3) arrives, the bandwidth of the good flows drop as the attacker flow takes up most of the bandwidth. The attacker receives a bandwidth between 8 and 8.5 Mbps, whereas the bandwidth of good flows drop to 3 Mbps. At 60 seconds, when the second attack flow arrives (flow 4), the bandwidth of the good flows further drops along with the bandwidth of the first attacker flow. The bandwidth of the flow 1 and flow 2 drops to 1.1 Mbps and 2.7 Mbps respectively. The bandwidth of first attacker flow drops to 4.6 Mbps. The new attacker flow receives 2.6 Mbps. This trend continues until 120 seconds, when the good flows end. At that point, the first attacker flow bandwidth recovers to its previous levels (8 Mbps) and the second attacker flow receives a higher bandwidth of 5.8 Mbps. At 150 seconds, when the first attacker flow ends, the second attacker flow receives 11.8 Mbps, which is close to the requested bandwidth of that flow.

Figure 7.9 shows the results for the DIPLOMA scheme when misuse detection is not in effect. Until 30 Seconds, where the attacker had not started sending any traffic, the good flows 1 and 2 get the bandwidth that are allocated in their capability. The bandwidth reported by the flows is 3.74 Mbps, which is slightly less than the allocated 4 Mbps due to the additional headers present in DIPLOMA packets. At 30 seconds, when the attacker starts sending the first attack flow (flow 3) at the rate of 12 Mbps, the bandwidth of the good flows drops only slightly to 3.71 Mbps. The attacker gets a bandwidth of 3.5 Mbps,

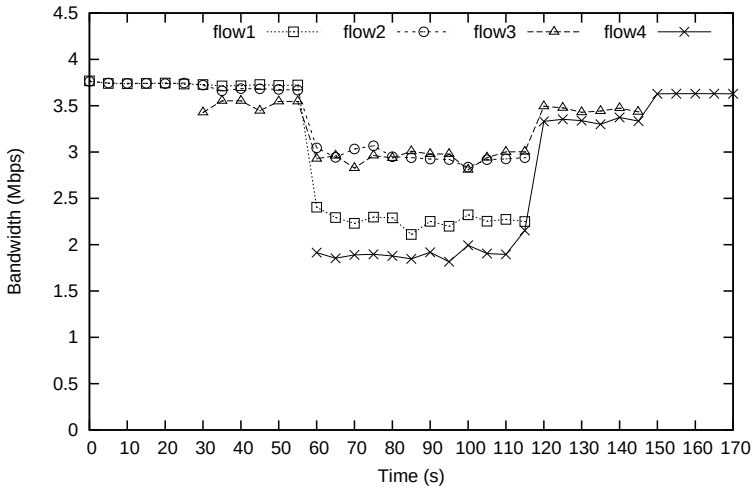

Figure 7.9: Bandwidth of the flows in DIPLOMA without misuse detection

which is closer to its allocated bandwidth. Hence, the consent-based DIPLOMA scheme is able to protect the good flows and contain the attacker to its allocated bandwidth. At 60 seconds, when the second attack flow (flow 4) starts, the bandwidth of the good flows and the attacker drops. This drop for the good flow is not as drastic as the original scheme. Here the bandwidth of the flow 1 drops to 2.3 Mbps and that of flow 2 drops to 3 Mbps. The existing attacker flow drops to 2.9 Mbps, and the new attacker flow receives 1.9 Mbps bandwidth. This drop in bandwidth is due to limited available bandwidth on the network. The attacker is reusing the capability at this point, and DIPLOMA ends up honoring the same capability in two node disjoint paths. At 120 s, the genuine flow ends. At that point, the first attack flow bandwidth moves back to its original level of 3.5 Mbps. The second attacker flow bandwidth ends up at 3.3 Mbps. This increase is due to the freed up capacity from the good flows. At 150 seconds, the first attack flow ends and the second attack bandwidth increases slightly to 3.6 Mbps. Even then, the bandwidth of the individual attack flows does not go above the allocated bandwidth of 4 Mbps, because DIPLOMA enforces the bandwidth. Therefore, in DIPLOMA without the misuse detection, an attacker cannot go above the allocated bandwidth in a single path. However, it can bypass that check by sending traffic to multiple destinations in disjoint paths, if the capability permits it. When this happens, genuine traffic is affected due to capacity sharing of the network.

Figure 7.10 shows the results for the DIPLOMA system with misuse detection. In this

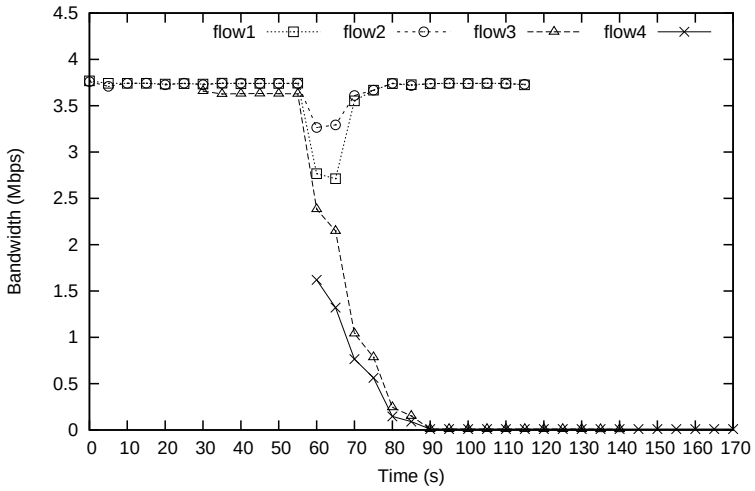

Figure 7.10: Bandwidth of the flows in DIPLOMA with misuse detection

experiment, when a verifier node detects a misuse using the distributed detection, it informs about the misuse to the forwarding nodes that have flows on the misused capability. Then the forwarding nodes sets the token bucket parameters of the corresponding flows to 0, which results in forwarding engine dropping those packets. The behavior of the system is same as DIPLOMA without misuse detection, until the misuse happens at 60 seconds. At 60 seconds, when the attacker sends the second flow (flow 4), which constitutes a capability reuse, the behavior changes from DIPLOMA without the misuse detection. In this case, the nodes send the record to the verifier (node 8), which detects the misuse. In this experiment, the period at which nodes send the record is 10 seconds. Hence, the verifier detects the misuse before 70 seconds, and informs the forwarding nodes. The forwarding nodes reset the token bucket parameters, which essentially block both the attack flows. The drop of the attack bandwidth is gradual due the nature of implementation of the token bucket algorithm in DIPLOMA to take care of packet losses as described in Chapter 5. The bandwidth of the good flows drops to 2.8 Mbps and 3.3 Mbps for a short duration (10 seconds) at 60 seconds while the misuse detection and recovery takes place. At 85 seconds, the recovery is complete and the bandwidth of the good flows moves back to the levels before the reuse attack. Even after the genuine flow ends at 120 s, the attacker flows continue to be blocked due to their misuse action. This continues even after the attacker stops the misuse at 150 seconds, when the first attacker flow ends. Hence, DIPLOMA can effectively contain capability misuse.

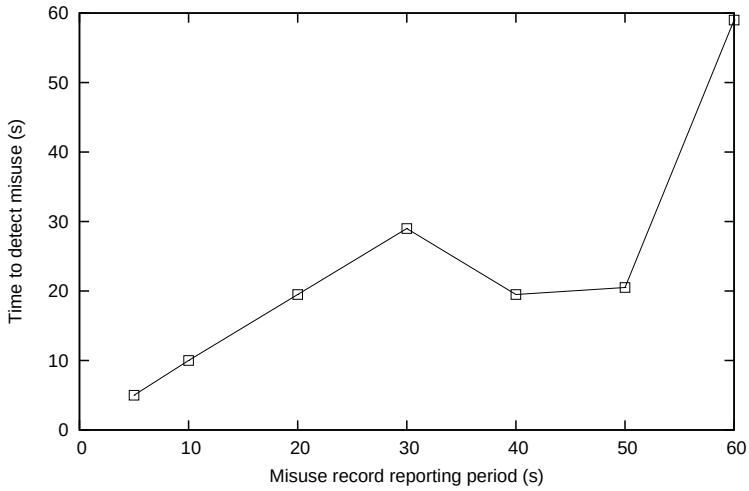

Figure 7.11: Time to detect the misuse for different record reporting periods

7.4.5 Speed of detecting misuse

Now we study how fast our scheme can detect the misuses and communicate with the affected nodes. The time to detect the misuse will depend on the period at which the records are send to the verifiers. Hence, we study the misuse detection speed as a function of that period.

In these experiments, the flows and their start times were the same as in previous experiments. Hence, the misuse happens at 60 seconds from the start of the first good flow, and 30 seconds from the start of the first attack flow. We varied the period at which the records were sent to the verifier, and measured the time difference between the time misuse notification arrived at a node and the time the misuse started.

Figure 7.11 plots the time required at the detector nodes to get the misuse notification after the misuse happened for various record reporting periods. For the periods up to 30 seconds, the time to detect is the same as the period. Hence, the misuse is detected as soon as the record is received to the verifier. For the periods 40 and 50 seconds the time to detect misuse was less than the period, and for 60 seconds the detection time was same as the period. This is because for 40 and 50 seconds experiments, the start of the attack and the start of the period may have not been synchronized. Hence, it is possible for the detectors to send the record to the verifier in time less than the period after the attack has

happened. The verifier may detect the misuse immediately after it receives the records.

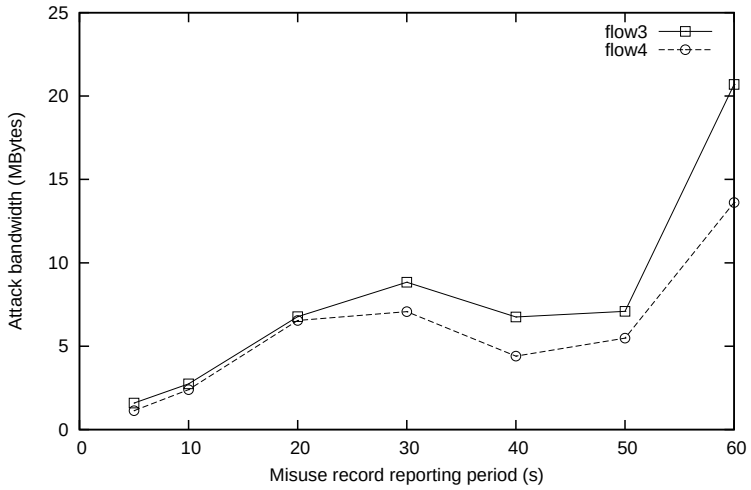

Figure 7.12: Additional attack traffic after misuse for different record reporting periods

7.4.6 Attack bandwidth after misuse

Now we study the amount of additional traffic the attacker is able to send after the misuse. This quantity depends on how fast the system is able to detect the misuse and take action against the attacker. Hence, we study the additional traffic as a function of record reporting period. We use the same flows and attack scenario as the previous experiment and measure the data received at the receiver for the attack flows (flows 3 and 4), after the misuse has started.

Figure 7.12 plots the amount of traffic received at the receivers for the attack flows after the misuse, for different record reporting periods. Up to 30 seconds, the attack traffic increases as the record reporting time increases. This is because the misuse traffic will be treated as the legitimate traffic and allowed to pass through until the misuse is detected; and the misuse detection time is proportional to the record reporting period. Note that even if the attacker is trying to send the traffic at 12 Mbps, the throughput the flows receive is less than 4 Mbps due to bandwidth enforcement by the intermediate nodes. In this set of experiments, the flow 3 had slightly higher bandwidth than flow 4, similar to the experiment in Figure 7.10. For 40 and 50 seconds, the attack traffic drops is less than that of 30 seconds, as the misuse is detected before the complete period as explained in the

previous experiment.

Chapter 8

Future Work

In this chapter, we discuss some of the future work. First, we discuss how DIPLOMA can be used for protecting routing protocols. Then we discuss the possible applications of the DIPLOMA architecture for network systems other than MANETs.

8.1 Protecting Routing Protocols using DIPLOMA

Defending against routing attacks has been a major challenge for MANETs [WCWC06; YLY+04; SP04]. Recently, there has been a lot of research in identifying and mitigating routing attacks on well-known routing protocols in MANETs. MANETs are very susceptible to these attacks because routing in MANETs depends on co-operation among the nodes, and unlike wired networks, these nodes are not specialized nodes whose sole purpose is routing. These nodes are not controlled by entities providing the services, unlike ISPs of the Internet. These nodes themselves compete for the resources for their own communication. Any compromised or hostile node can cause major interruption in the normal data flow on MANET.

Mobility poses another challenge in the routing of MANETs. When the nodes move, many of the routes in MANET are invalidated. These invalid routes trigger new route discovery, which can cause a flood of route discovery messages.

The solutions to flooding attacks on routing protocols involve limiting the number of route requests a node can issue and authentication of the routing messages. This limiting of

the routing messages is usually done in a uniform manner without considering the number of peers with which a node is communicating. A node that is sending traffic to a large number of nodes is likely to generate more routing requests than a node that has fewer communication peers.

The DIPLOMA architecture that we have presented so far in the context of data traffic can also be used for controlling routing messages. In on-demand routing protocols, a node does not have to initiate a route request for a destination unless it has data to send to that destination. It cannot send the data to a destination unless it has capability in the form of a policy token or a network capability. Hence, in DIPLOMA architecture, we require that the routing protocol packets have an associated capability. We can also extend policy tokens to allow access only for routing packets by specially marking them for that special purpose.

We can also restrict the rate of route requests for a capability to prevent a malicious node that has a valid capability from issuing too many route requests. This rate limiting needs to consider genuine route requests that come up due to changes in routes because of mobility. This scheme can restrict flooding attacks using routing packets but still allow nodes with a large number of communication peers to do a large number of route discoveries. We expect the overhead on the routing to be similar to that of data traffic, as routing also uses network layer packets.

8.2 Application of the Architecture to Other Environments

There are network situations other than MANETs that require distributed firewall solutions. We plan to explore the applicability of DIPLOMA in those cases as future work. Depending on the application, it may require modification or extension of the current protocol, or it may require the simplification of the current protocol. In this section, we describe some of the networks where the DIPLOMA architecture may be relevant.

8.2.1 Wired Networks

Even though we focus on MANETs, our system may also be used in wired networks. However, MANETs provide our architecture with both advantages and challenges. Specifically, the ratio of CPU cycles to available bandwidths (Hz/kbit) is normally higher in MANET nodes compared to their wired counterparts. This enables us to do more intelligent processing (and use cryptography) on most or all of the packets transiting through a MANET node. The number of traffic flows handled by a MANET node is also small due to the small network size. However, frequent route changes between a source and a destination node due to node mobility represent a difficult challenge in a distributed enforcement environment such as ours.

Unlike wireless networks, where the communication medium is broadcast in nature, wired network links are mostly point-to-point. In wired networks, routers can be trusted. Hence, in wired networks, we may also assume that the packets cannot be snooped and that malicious nodes cannot inject arbitrary packets. These assumptions depend on the physical security of the network, as well as the installation of existing packet filters (e.g., filtering for source address spoofing). In those cases, we will be able to reduce the protocol and the processing overhead by not requiring a per-packet signature.

8.2.2 Future Internet Architecture

The Internet has evolved over the years from its original design and architecture. At the beginning, the primary focus was connectivity. As it became widespread, security became an important problem, and various security solutions got incorporated into it. The Future Internet Design (FIND) [FIN10] and other clean-slate designs [cle10] propose security to be incorporated from day one. As future work, we will explore the applicability of DIPLOMA in those architectures.

8.2.3 Virtual Machine Clouds

The new model of computing that the industry is adopting today is cloud computing. Major industry players like Amazon, Google, and Microsoft are offering cloud computing solutions [ama10; goo10; mic10]. The popularity of cloud computing comes from its eco-

nomics: Its users can avoid capital expenditure on hardware, software, and services, paying for only what they use.

Cloud computing providers use virtualized resources for providing the service. Multiple virtual machines that provide services to different customers, sometimes competitors, may be running on the same hardware. To provide reliability and quality of services, the virtual machines may migrate among underlying hardware. Thus, the cloud-computing infrastructure behaves more like a mobile ad-hoc network, rather than like its wired counterparts. Providing security to the end customers of the cloud-computing infrastructure requires a distributed firewall like DIPLOMA. We will explore this problem in future work.

Chapter 9

Conclusions

Securing MANETs poses additional challenges that are not present in wired networks. MANETs do not have a well-defined perimeter. This calls for security to be enforced in a distributed manner. MANETs also have limited resources in terms of CPU processing, memory, and power. Any security solution has to consider the limited resources of the MANET nodes. Bandwidth is a scarce resource in MANETs. Hence, it is very important to protect the bandwidth resource if comprehensive security has to be provided. Mobility of the nodes causes routes between two nodes to change dynamically. Any security solution for MANETs needs to handle the route changes in a correct and efficient manner.

Other challenges in MANETs include the broadcast nature of the medium and the dual role of the MANET nodes as end hosts and routers. The broadcast nature of the medium makes the security solutions that rely on the secrecy of the router-to-router communication inadequate for MANETs. The dual nature of the nodes makes the routers on MANETs untrusted, causing security solutions that provide authority to routers to fail in MANETS.

In this thesis, we provide a comprehensive security solution for MANETs that can protect both end host resources and network bandwidth. Our solution considers the challenges of MANETS including the limited resources of nodes, mobility, the broadcast nature of the medium, and the untrusted nature of the routers. Our solution does not depend on the concept of perimeter and enforces security in a distributed manner.

Our architecture called DIPLOMA is a novel distributed security policy enforcement architecture designed specifically for MANETs. In this architecture, we extended the network

capability framework and tailored it to the resource-constrained MANET environment. The capabilities propagate both access control rules and traffic-shaping parameters that govern a node's traffic. In this deny-by-default model, nodes can only access the services and hosts they are authorized for by the capabilities given to them. The enforcement of the capability is done in a distributed manner by all the nodes in the path from the source to the destination. Compromised or malicious nodes cannot exceed their authority and expose the whole network to an adversary. The architecture helps mitigate the impact of Denial of Service (DoS) attacks because excess or unauthorized packets are dropped closer to the attack source. Thus, we avoided unnecessary data processing and forwarding at the target node and at the network itself.

Use of capabilities for access and bandwidth control provides certain benefits. Traditional firewalls and distributed firewalls use the access control list (ACL) model for access control. In the capability model, as the nodes that have the access carry the rule themselves, the rules are easy to update in distributed settings like MANETs. Furthermore, the resource-constrained nodes that are enforcing the policy do not have to maintain a large database of the rules. This is especially true for MANETs, as the nodes participating in a MANET at any point are not known in advance. Capability also prevents snooping attacks.

We evaluated our architecture using simulations before going for full implementation in real systems. Our main concern was performance bottlenecks. We implemented the architecture in the GloMoSim simulator. We implemented DIPLOMA as a layer between IP and AODV routing processing. As GloMoSim did not support packet processing delays, we also provided that support. Our evaluation showed that the performance impact of the system on latency and bandwidth was minimal for both TCP and UDP traffic. It also showed that the system performed well in the presence of mobile nodes. Mobility causes the route between the communication end points to change. The system was also able to allocate resources in a fair manner, even in the presence of attackers.

We implemented DIPLOMA in real systems running the Linux operating system. We adapted the original proposal for real implementation so that it would provide good performance and be effective against attacks. We used a novel signature scheme to integrity-protect the packets and drop the tampered packets closer to the source as a combination

of RSA signature and SHA-1 hashing. We implemented DIPLOMA as a user-level protocol engine that interfaces with the rest of the packet processing system through a *netfilter* framework. Our implementation worked at the network layer and did not require any changes to the existing applications. However, the applications saw the benefit of receiving only authorized traffic and were able to send the allocated traffic even in the presence of rogue nodes.

We evaluated the real system implementation on the Orbit lab testbed. For our evaluation, we devised a novel solution for creating multi-hop topologies on indoor environments, in which an attacker cannot cause unlimited damage to the nodes in its proximity by hogging the channel. Our experiments showed the effectiveness of the new signature scheme and low overhead on throughput, latency, and jitter. It also showed resiliency against attackers.

In this thesis, we also extended the DIPLOMA architecture to secure multicast traffic. Multicast traffic, such as audio and video streaming, is an important application for Mobile Ad Hoc Networks (MANETs). Due to the broadcast nature of the medium, multicast improves the utilization of the wireless links. The open nature of multicast, where any receiver can join a multicast group, and any sender can send to a multicast group, makes it an easy vehicle for launching Denial of Service (DoS) attacks in resource-constrained MANETs. We used capabilities to access control and limit the bandwidth of multicast traffic. This led to a unified solution for both receiver access control and sender access control, whereas previous researches looked at them as two separate problems. We presented how to modify popular multicast protocols such as ODMRP and PIM-SM to incorporate DIPLOMA.

We implemented multicast extensions to DIPLOMA in a GloMoSim simulator and in real systems running Linux. Our Linux implementation did not require any changes to the existing multicast applications or multicast routing daemons. We conducted extensive simulations in GloMoSim and extensive experiments in the Orbit lab testbed. Our evaluations showed that the multicast DIPLOMA incurs minimal overhead in terms of throughput, packet loss, and inter-arrival times and is effective against attackers. We also showed that it works well for streaming video.

DIPLOMA comes under the class of consent-based networking, where the nodes require

permission to send traffic. It uses capabilities as proof of consent. Specifically, DIPLOMA is a destination-based consent architecture where consent is given to send traffic to a destination on any of the available paths. Only destination-based consent architecture is suitable for MANETs, as the paths may change frequently. In DIPLOMA, consent may also be given to send traffic to a group of nodes in a single capability. Destination-based consent architectures are susceptible to multi-path misuse.

In this thesis, we identified the misuses in DIPLOMA architecture and provided solutions for distributed misuse detection and recovery. We provided capability encodings that aid in misuse detection. We presented protocols for the exchange of information to detect misuse in a distributed manner. We also presented an efficient algorithm for identifying misuses from the communication sessions and protocols for providing verifiable proofs of misuse so that affected nodes can take action against the attackers.

We implemented misuse detection algorithms as part of our DIPLOMA implementation in Linux. Once the misuse is detected, the nodes set the allocated bandwidth of the capabilities to zero. We conducted extensive experiments in the Orbit lab to study the effectiveness of our algorithms. We found that the algorithm required very minimal processing, that the amount of information exchanged was minimal, and that the system was able to detect misuses and recover from them.

As future work, we plan to study the architecture and extend it to other network scenarios including wired networks, future Internet architectures, and virtual machine clouds.

Bibliography

[ABC+09] Scott Alexander, Steve Bellovin, Yuu-Heng Cheng, Brian Coan, Andrei Ghetie, Vikram Kaul, Nicholas F. Maxemchuk, Henning Schulzrinne, Stephen Schwab, Bruce Siegell, Angelos Stavrou, and Jonathan M. Smith. The Dynamic Community of Interest and its Realization in ZODIAC. *IEEE Communications Magazine*, October 2009.

[ABL05] Mansoor Alicherry, Randeep Bhatia, and Erran L. Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. *ACM MOBICOM*, 2005.

[ACJ+02] P. Albers, O. Camp, B. Jouga, L. Me, and R. Puttini. Security in ad hoc networks: A general id architecture enhancing trust based approaches. *IEEE Network, WIS*, 2002.

[AGI05] F. Adelstein, S. K. S. Gupta, and G. G. Richard III. *Fundamentals of mobile and pervasive computing*. McGraw-Hill, 2005.

[AK10a] M. Alicherry and A. D. Keromytis. DIPLOMA: Distributed Policy Enforcement Architecture for MANETs. *International Conference on Network and System Security (NSS)*, September 2010.

[AK10b] M. Alicherry and A. D. Keromytis. Misuse detection in consent-based architecture, 2010. Manuscript.

[AK10c] M. Alicherry and A. D. Keromytis. Securing MANET Multicast Using DIPLOMA. *International Workshop on Security (IWSEC)*, November 2010.

[AKR07] R. Akbania, T. Korkmaz, and G.V.S. Raju. HEAP: A packet authentication scheme for mobile ad hoc networks. *Communications and Networking Simulation Symposium*, 2007.

[AKS09] Mansoor Alicherry, Angelos D. Keromytis, and Angelos Stavrou. Deny-by-Default Distributed Security Policy Enforcement in Mobile Ad Hoc Networks. *SecureComm*, September 2009.

[ama10] Amazon web services. http://aws.amazon.com/, 2010.

[Amm03] P. Judgeand M. Ammar. Security issues and solutions in multicast content distribution: A survey. *IEEE Network*, 17, 2003.

[AOD] Uppsala University AODV implementation. http://core.it.uu.se/core/index.php/AODV-UU.

[ARW03] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. *Proc. of Hotnets-II*, 2003.

[ASK09] M. Alicherry, A. Stavrou, and A. D. Keromytis. Evaluating a Collaborative Defense Architecture for MANETs. *IEEE Workshop on Collaborative Security Technologies (CoSec)*, December 2009.

[BC95] A. Ballardie and J. Crowcroft. Multicast-Specific Security Threats and Countermeasures. *SNDSS*, 1995.

[Bel99] S. M. Bellovin. Distributed firewalls. *;login:*, November 1999.

[BIK01] Matt Blaze, John Ioannidis, and Angelos Keromytis. Trust management for ipsec. *Symposium on Network and Distributed Systems Security (SNDSS)*, 2001.

[Blo82] Rolf Blom. Non-public key distribution. *In Proc. CRYPTO*, 1982.

[BR08] Azzedine Boukerche and Yonglin Ren. A trust-based security system for ubiquitous and pervasive computing environments. *Computer Communications*, 31(18):4343–4351, 2008.

[CB94] William R. Cheswick and Steven M. Bellovin. Firewalls and internet security: Repelling the wily hacker. *Addison-Wesley, Reading, MA*, 1994.

[CFP+07] Martin Casado, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown, and Scott Shenker. Ethane: Taking Control of the Enterprise. *ACM SIGCOMM*, August 2007.

[CGA03] C. M. Cordeiro, H. Gossain, and D. Agrawal. Multicast over Wireless Mobile Ad Hoc Networks: Present and Future Directions. *IEEE Network*, 17, 2003.

[CJ03] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). *IETF RFC 3626*, 2003.

[cle10] Clean slate design of the internet. http://cleanslate.stanford.edu, 2010. Stanford University.

[CPS03] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. *IEEE Security and Privacy*, 2003.

[CPS06] H. Chan, A. Perrig, and D. Song. Secure hierarchical in-network aggregation in sensor networks. *ACM CCS*, 2006.

[CZ95] Brent Chapman and Elizabeth Zwicky, editors. *Building Internet Firewalls*. Orielly & Associates Inc, Cambridge, 1995.

[DDHV03] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. *ACM CCS*, 2003.

[DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. *Crypto '89*, 1989.

[DFS02] Drew Dean, Matt Franklin, and Adam Stubblefield. An algebraic approach to IP traceback. *ACM Transactions on Information and System Security*, May 2002.

[DKB05] D. Djenouri, L. Khelladi, and A. N. Badache. A survey of security issues in mobile ad hoc and sensor networks. *IEEE Communications surveys & tutorials*, (4), 2005.

[DLA02] H. Deng, W. Li, and D. Agrawal. Routing security in wireless ad hoc networks. *IEEE Communications Magazine*, October 2002.

[dro10] DROID by Motorola. http://www.motorola.com/droid, 2010.

[EB06] P. Ebinger and T. Bucher. Modelling and analysis of attacks on the manet routing in aodv. *LNCS*, 2006.

[Ecr09] ECRYPT II Yearly Report on Algorithms and Key Lengths. http://www.ecrypt.eu.org/, 2009. European Network of Excellence in Cryptology II.

[EG02] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. *ACM CCS*, 2002.

[EMT89] D. Estrin, J. C. Mogul, and G. Tsudik. Visa protocols for controlling interorganizational datagram flow. *IEEE Journal on Selected Areas in Communications*, May 1989.

[FIN10] NSF NeTS FIND Initiative. http://www.nets-find.net, 2010. National Science Foundation.

[FMSK09] V. Frias-Martinez, S. J. Stolfo, and A. D. Keromytis. BARTER: Behavior Profile Exchange for Behavior-Based Admission and Access Control in MANETs. *International Conference on Information Systems Security (ICISS)*, December 2009.

[FS98] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. *IETF RFC 2267*, January 1998.

[GAA01] J. Grant, P. Attfield, and K. Armstrong. Distributed firewall technology. In *EWA*, CANADA, 2001.

[glo] GloMoSim simulator. http://pcl.cs.ucla.edu/projects/glomosim/. Mobile Computing Laboratory at UCLA.

[goo10] Google app engine. http://code.google.com/appengine/, 2010.

[GRGSK05] F. J. Galera, P. M. Ruiz, A. F. Gomez-Skarmeta, and A. Kassler. Security Extensions to MMARP Through Cryptographically Generated Addresses. *Lecture Notes on Informatics*, 2005.

[HBC01] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. *MobiHOC*, 2001.

[HC00] T. Hardjono and B. Cain. Key Establishment for IGMP Authentication in IP Multicast. *IEEE ECUMN*, 2000.

[HE03] L. Hu and D. Evans. Using directional antennas to prevent wormhole attacks. *NDSS*, 2003.

[HJP02] Y.-C. Hu, D. B. Johnson, and A. Perrig. Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks. *IEEE Workshop on Mobile Computing Systems and Applications*, 2002.

[HL04] Y. Huang and W. Lee. Attack analysis and detection for ad hoc routing protocols. *RAID*, 2004.

[HPJ02] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *MOBICOM*, 2002.

[HPJ03a] Y. Hu, A. Perig, and D. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. *WiSe*, 2003.

[HPJ03b] Y-C Hu, A. Perrig, and D.B. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. *INFOCOM*, 2003.

[IEE97] Standard for RSA, Diffie-Hellman and related public-key cryptography, March 1997. IEEE Working Group P1363.

[IKBS00] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. pages 190–199, 2000.

[iph10] Apple iPhone. http://www.apple.com/iphone/, 2010.

[JA02] P. Judge and M. Ammar. Gothic: A Group Access Control Architecture for Secure Multicast and Anycast. *INFOCOM*, 2002.

[JMH03] D.B. Johnson, D.A. Maltz, and Y-C. Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). *IETF Draft*, 2003.

[JS07] J. Jaramillo and R. Srikant. Darwin: Distributed and adaptive reputation mechanism for wireless ad-hoc networks. *MOBICOM*, 2007.

[KC09] Jonathan Kirsch and Brian Coan. Intrusion-tolerant group management for mobile ad-hoc networks. *International Workshop on Dependable Network Computing and Mobile Systems*, 2009.

[KGS06] S. Kaul, M. Gruteser, and I. Seskar. Creating Wireless Multi-hop Topologies on Space-Constrained Indoor Testbeds Through Noise Injection. *IEEE Tridentcom*, 2006.

[Kla] J. Klaue. EvalVid - A Video Quality Evaluation Tool-set. http://www.tkn.tu-berlin.de/research/evalvid/.

[KLNY03] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz. Secure Multicast Groups on Ad Hoc Networks. *ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[KV05] Pradeep Kyasanur and Nitin H. Vaidya. Capacity of multi-channel wireless networks: Impact of number of channels and interfaces. *ACM MOBICOM*, 2005.

[KW02] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *In First IEEE International Workshop on Sensor Network Protocols and Applications*, 2002.

[KYN01] R. Kravets, S. Yi, and P. Naldurg. A security-aware routing protocol for wireless ad hoc networks. *ACM MobiHOC*, 2001.

[LdSP09] Michele Nogueira Lima, Aldri L. dos Santos, and Guy Pujolle. A survey of survivability in mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 11(1), 2009.

[Lee93] R. Lee. Netware 4.x performance tuning and optimization: Part 3. *http://support.novell.com/techcenter/articles/ana19931001.html*, October 1993.

[Lev84] Henry M. Levy. Capability-based computer systems. *Digital Equipment Corporation*, 1984.

[LGC99] S.-J. Lee, M. Gerla, and C.-C. Chiang. On-Demand Multicast Routing Protocol. October 1999.

[Li00] W. Li. Distributed firewall. *GeoInformatica*, 2000.

[LN03] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. *ACM CCS*, 2003.

[mge] Multi Generator (MGEN). http://cs.itd.nrl.navy.mil/work/mgen/. Naval Research Laboratory.

[mic10] Microsoft azure. http://www.microsoft.com/azure/default.mspx, 2010.

[MMAK06] M. Muthuprasanna, G. Manimaran, M. Alicherry, and V. Kumar. Coloring the Internet: IP Traceback. *International Conference on Parallel and Distributed Systems*, 2006.

[MMDM04] N. Milanovic, M. Malek, A. Davidson, and V. Milutinovic. Routing and security in mobile ad hoc networks. *IEEE Computer Society*, 2004.

[MNP04] A. Mishra, K. Nadkarni, and A. Pactha. Intrusion detection on wireless ad hoc networks. *IEEE Wireless Communications*, 2004.

[MSPR05] J. McCune, E. Shi, A. Perrig, and M. Reiter. Detection of denial-of-message attacks on sensor network broadcasts. *IEEE Security and Privacy*, 2005.

[Mun09] Muni Wireless. List of US cities and counties with WiFi. http://www.muniwireless.com, March 2009.

[net] Nefilter/IPtables project home page. http://www.netfilter.org.

[nex10] Google Nexus One Phone. http://www.google.com/phone, 2010.

[NIS94] FIPS Publication 186: Digital Signature Standard, May 1994. National Institute of Standards and Technology (NIST).

[NSW+10] J. Naous, A. Seehra, M. Wafish, D. Mazires, A. Nicolosi, and S. Shenker. Defining and enforcing transit policies in a future internet. *Technical Report TR-10-07, Department of Computer Science, The University of Texas at Austin*, Februrary 2010.

[ope] OpenSSL: The Open Source toolkit for SSL/TLS. http://www.openssl.org/.

[orb] Orbit Lab Test-bed. http://www.orbit-lab.org. WINLAB - Rutgers University.

[OTL03] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). *IETF Draft*, 2003.

[par] Parsec: Parallel simulation environment for complex systems. http://pcl.cs.ucla.edu/projects/parsec/. University of California Los Angeles.

[PB94] C.E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *ACM SIGCOMM*, 1994.

[PBRD03] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on demand distance vector (AODV) routing. *IETF RFC 3561*, 2003.

[Per01] C. E. Perkins. *Ad hoc networking: an introduction*. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, 2001.

[PH02] P. Papadimitratos and Z. Haas. *Handbook of ad hoc wireless networks*. CRC Press, 2002.

[PPG05] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. *IEEE Security and Privacy*, 2005.

[Pre09] Bart Preneel. Upgrading cryptographic algorithms for network security. http://homes.esat.kuleuven.be/~preneel/preneel_securecom09.pdf, September 2009. SecureComm - Invited talk.

[PSW+01] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. Spins: Security suite for sensor networks. *MOBICOM*, 2001.

[PWS+07] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. Portcullis: protecting connection setup from denial-of-capability attacks. *SIGCOMM Comput. Commun. Rev.*, 37(4):289–300, 2007.

[RASJ05] S. Roy, V. G. Addada, S. Setia, and S. Jajodia. Securing MAODV: Attacks and Countermeasures. *IEEE Intl. Conf. SECON*, 2005.

[RFS+03] S. Ramaswamy, H. Fu, M. Sreekantaradhya, J. Dixon, and K. Nygard. Prevention of cooperative black hole attack in wireless ad hoc networks. *International Conference on Wireless Networks*, 2003.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, (2), 1978.

[RSA10] What key size should be used? http://www.rsa.com/rsalabs/node.asp?id=2264, 2010. RSA Laboratories.

[RW02] R. Russell and H. Welte. Linux netfilter hacking howto. *http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.html*, 2002.

[SDL+02] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A secure routing protocol for ad hoc networks. *IEEE Intl. Conf. on Network Protocol (ICNP)*, 2002.

[Sim92] G.J. Simmons, editor. *Contemporary Cryptology - The Science of Information Integrity*. IEEE Press, 1992.

[SKC+09] D. Slezak, T. Kim, A. C. Chang, T. Vasilakos, M. Li, and K. Sakurai. Security in Tactical MANET Deployments. *Comm. and NetInt. Conf., FGCN/ACN*, 2009.

[SNW+06] K. Sun, P. Ning, C. Wang, A. Liu, and Y. Zhou. Tinysersync: Secure and resilient time synchronization in wireless sensor networks. *ACM CCS*, 2006.

[SNW+09] A. Seehra, J. Naous, M. Walfish, D. Mazires, A. Nicolosi, and S. Shenker. A policy framework for the future internet. *ACM Workshop on Hot Topics in Networks (HotNets)*, October 2009.

[SP04] E. Shi and A. Perrig. Designing secure sensor networks. *IEEE Wireless Communications*, 2004.

[Ste01] R. Stepanek. Distributed firewalls. In *Seminar on Network Security*, Helsinki University of Technology, Finland, 2001.

[SWKA00] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. *ACM SIGCOMM*, 2000.

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall, 3 edition, 1996.

[Tul06] Bill Tulloh. On the spread of the capability approach. *http://www.nabble.com/On-the-Spread-of-the-Capability-Approach-to5608409.html*, 2006.

[TWKL06] C. Tseng, S. Wang, C. Ko, and K. Levitt. Demem: Distributed evidence-driven message exchange intrusion detection model for manet. *RAID*, 2006.

[WABL94] E. Wobber, M. Abadi, M. Burrows, and B. Lampson. Authentication in the TAOS Operating System. *ACM Transactions on Computer Systems*, 12, February 1994.

[WCWC06] B. Wu, J. Chen, J. Wu, and M. Cardei. A survey on attacks and countermeasures in manets. In *Wireless/Mobile Network Security*, chapter 12. Springer, 2006.

[WTLB07] S. Wang, C. Henry Tseng, K. Levitt, and M. Bishop. Cost-sensitive intrusion responses for mobile ad hoc networks. *RAID*, 2007.

[WY07] X. Wu and D. K. Y. Yau. Mitigating Denial-of-Service Attacks in MANET by Distributed Packet Filtering: A Game-theoretic Approach. *ASIACCS*, March 2007.

[XZ98] M. Gerla X. Zeng, R. Bagrodia. Glomosim: A library for parallel simulation of large-scale wireless networks. *Workshop on Parallel and Distributed Simulation*, 1998.

[YDZZ05] P. Yi, Z. Dai, S. Zhang, and Y. Zhong. A new routing attack in mobile ad hoc networks. *International Journal of Information Technology*, 2005.

[YLY+04] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in mobile ad hoc networks: Challenges and solutions. *IEEE Wireless Communications*, 2004.

[YPS04] Abraham Yaar, Adrian Perrig, and Dawn Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *In IEEE Symposium on Security and Privacy*, pages 130–143, 2004.

[YWA05] Xiaowei Yang, David Wetherall, and Thomas Anderson. A DoS-limiting network architecture. In *In Proceedings of ACM SIGCOMM*, pages 241–252, 2005.

[YZV03] Z. Yan, P. Zhang, and T. Virtanen. Trust evaluation based security solution in ad hoc networks. *Nordic Workshop on Secure IT Systems (NordSec)*, 2003.

[ZCB08] Hang Zhao, Chi-Kin Chau, and Steven M. Bellovin. ROFL: Routing as the firewall layer. *New Security Paradigms Workshop*, September 2008.

[ZH99] L. Zhou and Z. Haas. Securing ad hoc networks. *IEEE Network*, 1999.

[Zim93] P. Zimmermann. Pretty good privacy users guide. 1993.

[ZJCB09] Hang Zhao, Maritza Johnson, Chi-Kin Chau, and Steven M. Bellovin. Source prefix filtering in ROFL. *Technical Report, Department of Computer Science, Columbia University*, July 2009.

[ZL00] Y. Zhang and W. Lee. Intrusion detection for wireless ad-hoc networks. *MOBICOM*, 2000.

[ZSJ03] S. Zhu, S. Setia, and S. Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. *ACM CCS*, 2003.

[ZSJN04] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering false data injection in sensor networks. *IEEE Security and Privacy*, 2004.