12-11-2009

Secure Ad Hoc Routing Protocols With Detection, Identification And Self-Healing Capabilities

Sivakumar Ayeegoundanpalayam Kulasekaran

SECURE AD HOC ROUTING PROTOCOLS WITH DETECTION, IDENTIFICATION AND SELF-HEALING CAPABILITIES

By

Sivakumar Ayeegoundanpalayam Kulasekaran

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science
in the Department of Computer Science and Engineering

Mississippi State, Mississippi

December 2009

SECURE AD HOC ROUTING PROTOCOLS WITH DETECTION, IDENTIFICATION AND SELF-HEALING CAPABILITIES

By

Sivakumar Ayeegoundanpalayam Kulasekaran

Approved:

Mahalingam Ramkumar
Associate Professor of Computer Science and Engineering
(Major Professor)

Rayford B. Vaughn
Professor of Computer Science and Engineering
(Committee Member)

Ioana Banicescu
Professor of Computer Science and Engineering
(Committee Member)

David A. Dampier
Associate Professor of Computer Science and Engineering
(Committee Member)

Yoginder S. Dandass
Associate Professor of Computer Science and Engineering
(Committee Member)

Edward B. Allen
Associate Professor
and Graduate Coordinator
Computer Science and Engineering

Sarah A. Rajala
Dean
James Worth Bagley
College of Engineering

Name: Sivakumar Ayeegoundanpalayam Kulasekaran

Date of Degree: December 11, 2009

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Mahalingam Ramkumar

Title of Study: SECURE AD HOC ROUTING PROTOCOLS WITH DETECTION, IDENTIFICATION AND SELF-HEALING CAPABILITIES

Pages in Study: 147

Candidate for Degree of Doctor of Philosophy

Devices taking part in mobile ad hoc networks (MANET) co-operate with each other to route packets by strictly adhering to the ad hoc routing protocol in use. Malicious nodes taking part in co-operative routing can launch a wide variety of attacks to reduce the utility of MANETs. The aim of secure routing protocols is to ensure that MANETs can continue to function even in the face of malicious nodes. Secure routing protocols should have measures to dissuade attackers by detecting inconsistencies, identifying the perpetrator responsible for the inconsistency, and provide means to inhibit the role of misbehaving nodes. Most existing secure routing protocols try to achieve only first step, viz., detection of inconsistencies. This dissertation research investigates and proposes efficient strategies that substantially enhance the scope of assurances provided by secure MANET routing protocols while keeping the overhead low.

DEDICATION

To my parents A.B. Kulasekaran and K. Tamilarasi who have provided me consistent love, support and encouragement which guided me this far.

ACKNOWLEDGMENTS

I have had a very humble beginning in education. If anyone had asked if I would earn a doctoral degree ten years back, I would have said probably not. Now that seems to be in a very distant past, and I am here to thank everyone who has helped me in many a different way for supporting and guiding me towards my doctoral degree from Mississippi State University (MSU).

First, my sincere and earnest gratitude goes to my major professor and dissertation director, Dr. Mahalingam Ramkumar for his support, knowledge and guidance that he has provided me these many years. I would especially like to mention his patience with me and for his accommodation of my inaccuracies. I am very thankful for the financial support and research opportunities that he has provided me during my tenure at MSU. I could not have asked for a better advisor, colleague and more importantly a mentor.I thank God for giving me such a wonderful advisor with whom I have spend a fruitful last 6 years

I would like to extend my thanks to Dr. Rayford A. Vaughn, Dr. David A. Dampier, Dr. Ioana Banicescu and Dr. Yoginder Dandass for serving as members in my PhD Dissertation Committee and providing me with helpful and insightful suggestions and feedback.

I would also like to thank Dr. Julia E Hodges, for providing me the financial support as the teaching assistant in the department of Computer Science and Engineering. I would

like to thank the departmental staff Ms. Brenda Collins, Ms. Jo Coleson, Ms. Courtney Blaylock, and Ms.Keri Chisolm for their support.

I would like to acknowledge the assistance of some former/current graduate students in the HPCL lab namely, Rikk Anderson, Michael King, and Brian Gaines.

On a personal note, I would like to thank a few of my friends, Ravi Srirangam, Vijay Velusamy, Bharat Krishnan, Raja Veeramani, Veera Vijayaraj and Vinod Rajan who have provided me with valuable advise from time to time.

Last but not least, my heartfelt thanks to my parents for their love, prayers, support and motivation which guided me through this long journey towards earning my PhD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

MANET	Mobile Ad hoc Networks
DSR	Dynamic Source Routing
AODV	Ad hoc On Demand Routing
DSDV	Dynamic Sequenced Distance Vector
RDN	Reliable Delivery Neighborhood
PLN	Private Logical Neighborhood
RREQ	Route Request
RREP	Route Response
RERR	Route Error
MAC	Medium Access Control
RTS	Request To Send
CTS	Clear To Send
HMAC	Hashed Message Authentication Code
KDS	Key Distribution Scheme
KDC	Key Distribution Center
KPS	Key Pre distribution Scheme
DS	Digital Signatures

CHAPTER 1

INTRODUCTION

An ad hoc network, also referred to as mobile ad hoc networks (MANET) [1], is a collection of small-range wireless mobile devices that can communicate with each other without any infrastructure for routing packets. Eliminating the need for an infrastructure is achieved by the mobile devices themselves taking on the responsibility of routing packets between each other. Ad hoc networks have many useful applications like emergency operations in a battle field, or for establishing temporary communication infrastructure in times of natural disasters. Long-lived MANETs can also be useful for remote and rural areas where no communication infrastructure currently exists.

1.1 MANET Routing Protocols

Nodes forming MANETs also double as routers which follow *MANET routing protocols.* MANET routing protocols can be classified into *proactive* protocols that strive to maintain a consistent view of the changing topology at all times, and *reactive* protocols which attempt to discover the necessary topology information only when necessary.

While MANET routing protocols are similar in principle to routing protocols used in wired networks, there are many differences that make realization of good MANET protocols challenging. Some of the essential differences are as follows:

1. mobile nodes are typically resource challenged;
2. links and routes change more dynamically; and
3. unlike routers in wired networks (which are extended physical protection from attackers) some mobile routers (nodes) may be operated by attackers.

1.1.1 Secure Routing Protocols

Secure MANET routing protocols strive to function effectively even in the presence of malicious routers. Providing any assurance of security calls for some overhead. For battery operated MANET devices high overhead cannot be tolerated. Thus, while proactive security features are crucial for smooth functioning of MANETs, such features have to be added with minimal overhead. This makes the design of secure MANET routing protocols even more challenging.

1.2 Hypothesis and Research Strategy

The hypothesis of the research performed towards this dissertation was that comprehensive secure MANET routing protocols with the abilities to

1. *detect* inconsistencies in routing information,
2. *identify* perpetrators responsible for the detected inconsistency,
3. *heal* the network by suppressing the role of non-cooperative nodes

can be designed with overhead comparable to or lower than existing protocols which merely strive to detect inconsistencies.

1.2.1 Research Strategy

This dissertation research approached the problem of developing a comprehensive secure MANET routing protocol by investigating several shortcomings of current routing protocols, and seeking strategies to overcome such shortcomings. The research revealed multiple shortcomings common to many secure routing protocols. Broadly, such shortcomings can be classified into two categories:

1. the extent of assurances offered by secure MANET routing protocols can be substantially higher than what is sought by current secure protocols (or current protocols "aim low").
2. improper choice of cryptographic authentication strategies, resulting in substantial overhead.

1.2.2 Scope of Secure MANET Protocols

All routing protocols are essentially strategies for i) exchanging topology information, and ii) using the topology information to find the best routes or forwarding paths. Secure MANET protocols should include features to *dissuade* attackers from taking part in the protocol. The features necessary in secure MANET protocols to achieve this goal (dissuading attackers) can be seen as three different steps with increasing complexity.

Step 1: The basic requirement of any secure routing protocol is for routers to be able to *detect* inconsistencies in the routing information. In a scenario where an attacker sends a packet indicating some (incorrect) topology information, the receiver should have the ability to detect that "some inconsistency exists" in the routing data. This is possible only if there are redundancies in the routing information.

Step 2: If it is possible for nodes to detect that some inconsistency exists, the offending packet can be dropped. However, unless nodes can *identify* the source of the inconsistency (or *which* router was responsible for sending the offending packet) there is nothing that dissuades the attacker from repeatedly sending false routing information packets. Thus, the second requirement is to have strategies in place to *unambiguously identify* misbehaving nodes.

Step 3: Once a misbehaving node has been identified by one or more nodes, the next step is to ensure that the role played by the misbehaving node is eliminated, or at least reduced. Ideally, misbehaving nodes (which violate the routing protocol) should be "ejected" from the network.

1.2.3 Limitations of Current Protocols

Most MANET protocols, when originally proposed, simply ignored the possibility of malicious nodes. This is understandable, as even if all nodes behave well, MANET protocols are still more challenging compared to their wired counterparts due to higher mobility rates and modest resources.

Most secure MANET protocols in the literature are simply extensions of original MANET protocols with the addition of cryptographic authentication of routing data. However, the scope of cryptographic authentication is usually restricted to only *detecting* inconsistencies. Very little effort has been directed towards extending the scope of the protocols to enable *identification* of misbehaving nodes. Against such protocols which do not have features to identify misbehaving nodes, attackers can inflict a variety of "risk-free" attacks.

Some important issues that need to be considered for detecting and/or identifying misbehaving nodes are the "whos" and "whens":

1. *When* are inconsistencies detected? *When* are malicious nodes identified? By a neighbor of the perpetrator? Or by nodes many hops downstream?
2. *Who* detects inconsistencies? *Who* identifies perpetrators? All nodes? A small set of nodes? Just one node?

Such issues (which are protocol specific) will in turn determine appropriate strategies for reducing the role of misbehaving nodes. An important factor to be considered in determining the whos and whens is the *type* of cryptographic authentication used. The effect of the choice of cryptographic schemes on the "whos and whens" have been ignored in the literature.

For purposes of ejecting misbehaving nodes from the network, strategies to obtain non repudiable proof of misbehavior are essential. Such an approach can also be a severe deterrent for nodes intending to carry out attacks. Many secure MANET protocols employ non-repudiable authentication. However, while non-repudiable authentication is necessary, it is *not sufficient* for providing non repudiable proof of misbehavior. Thus far no protocol in the literature has addressed practical issues in obtaining non repudiable proof of misbehavior.

1.3 Specific Contributions

Some of the specific contributions of this dissertation are as follows:

1: One pitfall common to many secure MANET protocols is in ignoring physical layer issues. Most protocols simply assume that all links are bi-directional. Strategies

to ensure that only true neighbors (with whom bi-directional links exist) can gain access to packets is an access-control issue, which is clearly under the scope of secure MANET protocols. The effects of one-way links on the efficiency and security of current secure MANET protocols were investigated. A proactive strategy which demands low overhead to mitigate or avoid the use of one-way links was developed. This research was presented in an IEEE conference [2].

2: Some recent research results indicate that contrary to the widely-held belief, scalable and light-weight schemes for establishing pairwise secrets are practical even for resource limited mobile devices. This was the motivation for an alternate approach for countering issues related to one-way links: by enforcing private logical neighborhoods (PLN). Apart from addressing issues related to one-way links, mandating a PLN has several other advantages for any secure MANET protocol. A paper describing many of the advantages of imposing PLNs was presented in an ACM conference [3] in October 2009.

3: An important requirement for secure routing protocols is some redundancy in routing information. A efficient strategy for introducing this redundancy is by carrying over authentication. Two popular approaches used for this purpose include i) the per-hop hashing strategy and ii) employing two-hop secrets. The per-hop hashing strategy rests on the assumption that non-neighbors will not gain access to the per-hop hash values broadcast by a node. In [3] we argued that this assumption makes the use of PLN mandatory.

Most strategies in the literature for establishing two-hop secrets demand overhead proportional to the size of the two-hop neighborhood. We showed that a recent broadcast encryption scheme [4] can be used efficiently to establish two-hop secrets for MANETs.

The feasibility of such an approach, for which the overhead is proportional only to the number of neighbors (and *not* the number of nodes in a two-hop neighborhood) was investigated for securing AODV and DSR. Two papers were presented in IEEE conferences: [5] for securing DSR and [6] for securing AODV.

4: Ariadne [7] is a popular secure extension of DSR. We demonstrated a wide variety of risk-free attacks on Ariadne, and a simple extension of Ariadne to overcome such attacks. We showed why replacing the TESLA broadcast authentication protocol with a scheme for pairwise secrets will have a substantial impact on the resiliency of Ariadne. This research was presented in an IEEE conference [8].

5: APALLS, a comprehensive secure routing protocol based on the dynamic source routing (DSR) protocol was designed. APALLS includes elements to detect inconsistencies, identify and route around misbehaving nodes. APALLS also includes explicit features to address passive attacks involving selective participation, and semi-active attacks. APALLS is also the first secure routing protocol which includes the ability to obtain non repudiable proof of active attacks which can be submitted to appropriate authorities to revoke offending nodes from the network. Due to careful choice of cryptographic primitives APALLS achieves all these features with low overhead. This work has been submitted recently to the IEEE Transactions on Information Forensics and Security [9].

1.4 Organization of this dissertation

This dissertation is organized as follows. Chapter 2 is a survey of current secure routing protocols. The primary pitfalls of current protocols, which serve as the motivation for this dissertation, are explained in Chapter 3.

Chapter 4 presents a proactive strategy to inhibit the use of one-way links. Chapter 5 discusses the concept of a *private logical neighborhood* (PLN) and its uses.

Chapter 6 discusses several issues that need to be considered for cryptographic authentication of routing data. Chapter 7 describes the use of multi-source broadcast encryption schemes in authentication of routing data and includes description of two novel MANET protocols that employs broadcast encryption.

Chapter 8 describes describes the comprehensive protocol, APALLS, based on DSR.

Conclusions are offered in Chapter 9. Chapter 9 also points out some possible future extensions to this dissertation research, and publications resulting from this dissertation work.

Some of the notations frequently used in this manuscript are listed in Table 1.1 (other notations which are used only in specific chapters are tabulated in the respective chapters).

Table 1.1

Commonly Used Notations

$A, B, \ldots,$	(uppercase alphabets) node IDs
$\|$	concatenation of fields
$K[M]$	symmetric encryption of a message M using a key K
K_{SD}	shared symmetric key between S and D
$h()$	cryptographic hash function
$h^i()$	repeated application of the hash function $h()$, i times
$h(M, K)$	Hashed message authentication code (HMAC) for a message M using the secret K

CHAPTER 2

LITERATURE SURVEY

Consider a new node joining an ad hoc subnet. It has no idea about topology of the subnet. The new node has to learn topology by overhearing routing traffic around it or by gathering such information from other nodes nearby. Once the node has some knowledge of the topology, it can send packets to other nodes or help other nodes to exchange packets between themselves.

2.1 MANET Routing Protocols

The rules that all nodes are expected to follow for this purpose, is the MANET routing protocol. It is essential that every node follows a uniform protocol to perform the co-operative task of routing.

2.1.1 Requirements and Constraints

Devices taking part in MANETs are usually resource limited battery-operated mobile computers. However, MANET protocols will still have to address many demanding requirements [10].

Adapting to Topology Changes: Mobile nodes are free to move arbitrarily. They can join some neighborhood for some duration and power themselves off or leave neighbor-

hood after some time. Nodes have to realize such changes in local topology and possibly convey such changes to other regions in the network. It is the responsibility of a routing protocol to efficiently handle these changes without disrupting the network functionality. Good routing protocols should adapt to such frequent topology changes with minimal use of its resources [10].

Bandwidth Issues: Unlike traditional wired networks, bandwidth overhead is an important consideration for MANETs. Some nodes may find themselves at a high density area through which most traffic is routed. Also, frequent topology changes can result in large bandwidth overhead [10].

Limited Power: For battery operated mobile devices, power consumption should be limited. Reducing power consumption calls for i) reducing range of devices (leading to more hops and more routing complexity); ii) reducing bandwidth overhead; and ii) deliberately reducing the capability of processors used in mobile devices [10].

2.1.2 Classification of MANET Protocols

Obviously it is not possible to have common routing protocol to cater for all situations. There are many routing protocols designed for MANETs [11]. While the main goal of all routing protocols is to find routes, they differ in specific strategies for finding routes, storing routes, and adapting to changes [11] in topology. MANET protocols can be classified into three major categories:

1. Proactive,
2. Reactive, and
3. Hybrid routing protocols.

Table driven routing protocols take a proactive approach in which route to every other node will be maintained consistently even when they are not in use. Every node is required to maintain a routing table with forwarding paths to all nodes in the network. These tables are updated regularly even when there is only a small change in the network. All nodes are required to share their routing table information with all other nodes in its neighborhood. Table driven protocols are better suited for networks of small size. They are particularly inefficient for large networks with high mobility rates. Some of the well known table driven routing protocols are [11]

1. Distance-Sequenced Distance-Vector protocol (DSDV) [12]
2. Cluster-head Gateway Switch protocol (CGSR) [13]
3. Wireless Routing Protocol (WRP) [14]

Proactive protocols also include protocols based on link-state approaches like the optimized link state routing protocol (OLSR) [15]. DSDV is perhaps the most popular proactive protocol.

In reactive routing protocols [11] nodes find routes one and only when needed. They do not maintain routes to all other nodes in the network. If a node needs to find a route to destination, it initiates the route discovery process and finds the route to that destination. Thus end to end latency (in finding routes between some source and destination) is typically longer than proactive techniques. Some examples of on demand routing protocols are [11]:

1. Ad hoc On demand Distance Vector (AODV) [16]
2. Dynamic Source Routing (DSR) [17]
3. Associativity Based routing (ABR) [18], and

4. Signal Stability Routing (SSR) [19]

On demand routing techniques are more useful in cases where node mobility is high, or in cases where there the network size is large. DSR and AODV are the most popular on-demand protocols.

Both proactive and reactive routing protocols have some unique advantages. Hybrid routing protocols are combinations of proactive and reactive protocols. Some examples of hybrid protocols are

1. Zone Routing Protocol (ZRP) [20], and
2. Sharp Hybrid Adaptive Routing Protocol (SHARP) [21]

2.1.3 Destination Sequenced Distance Vector (DSDV) Protocol

DSDV [12] is a table driven approach based on Bellman-Ford algorithm [22]. Every node maintains a routing table that contains forwarding information to all nodes in the network. Every node periodically broadcasts to its neighbors, the information contained in their routing table. Routing table entries consist of i) neighbor of the source node through which it can reach the destination; ii) distance to the destination node; and iii) information about the last known sequence number for that destination (which was created originally by the destination node). Sequence numbers are used to avoid stale routes and overcome count-to-infinity problem [22].

When a node receives a route update about a destination node, it accepts it only if the sequence number stated in the route update packet is greater than the one which it has. If a source node has two paths to the destination with the same sequence number it chooses the route with a lower hop count value.

Exchanging routing tables can use the "full dump" approach in which a node shares the information of entire routing table to all its neighbors, or the "incremental dump" approach in which each node just periodically updates only the *changes* in topology information. Full dump update is used when there is a major topology change. Incremental update is done when there is a small change in the topology information and can be used in between two full dump updates. Nodes always use even number sequence number to show the freshness of the routes. If a node A is on the path between S and D and moves away from S, node S will update its routing table regarding the path to destination node D with sequence number to odd number greater than the current sequence number it has for node D, and set hop count field to infinity. Every node which receives this update from node S will update their routing table to reflect that node D is unreachable. DSDV protocol might take time to recover from a sudden collapse of the network such as when many nodes move out of the network.

2.1.4 Ad hoc On Demand Distance Vector Routing Protocol

AODV [16] is an on demand routing protocol. When a node desires to send data packets to some other node, it first check its route cache to see if it has a route to that destination. If not, the source node initiates a *route discovery* process. Route discovery process consists of three major steps. They are route solicitation using a route discovery (RREQ) packet, route reply (RREP), and route maintenance (RERR). Sequence number is used in AODV to avoid loops. The RREQ packet indicates the destination node ID,

last known destination sequence number, route request ID and the sequence number of the source. Source node propagates this information to all its neighbors.

Neighbors receiving this information first check to see if they have already received the same RREQ packet from the source. If so, the RREQ is discarded. Else, it checks in its route cache if it has route to the destination. If it has route to the destination and the destination sequence number is greater or at least equal to the one in RREQ packet, this intermediate node generates RREP packet. It copies all the necessary information from the RREQ packet and sends an RREP packet back to the source node. If an intermediate node does not have a route to the destination, then it will increment the hop count value by one and forward the RREQ packet onward. In addition, each node stores (in its route cache) the details about RREQ packet and the previous hop node through which it had received the RREQ. This information will be useful later when a node receives RREP for a particular RREQ, to unicast RREP back to the source via from the node from which it received RREQ.

RREQ message propagates until it reaches the destination node or any intermediate node having path to the source node. Destination node generates a RREP packet and unicasts it back to the source node. Source node receives this RREP and updates it route to the destination. Each RREQ packet has some timer associated with it. If a source node sends RREQ packet and waits for longer period of time for the route reply, it generates another fresh RREQ packet to the destination node and propagates it.

When a node detects a link to neighbor is broken, then it originates RERR message back to the source node informing the particular node is not reachable. In AODV, latency

time in finding routes can be high especially for large network sizes. AODV does not incur as much bandwidth overheads during its operation as compared to DSDV. AODV is most widely used protocol in the current ad hoc community. AODV assumes all links are bi-directional. Each node sends periodic hello messages to all its neighbors to ensure the connectivity among each other.

2.1.5 Dynamic Source Routing Protocol

Dynamic source routing protocol (DSR) [17] is an on demand source routing protocol. DSR is a loop free, multi-path routing protocol. Each node can have multiple paths to a destination as opposed to single path in protocols like AODV. DSR specifies a route discovery process and a route maintenance process. DSR does not mandate link layer acknowledgment or neighborhood discovery. DSR can operate with unidirectional and bidirectional links.

Route Discovery Process: Consider a case where node A needs to find a route to node D. Node A originates route discovery process by creating a route request (RREQ) message in search for a path to the destination D. The RREQ is flooded. The fields in RREQ packets include source and destination IDs (addresses), source sequence number, and hop limit. Source ID and sequence number is used to keep flooding in control. Any node who receives the packet with same source ID and sequence number processes it only once (drops the subsequent ones). Every node involved in flooding the RREQ inserts its ID. The RREQ propagates until the destination or an intermediate node which has a cached path to the destination.

At any point in time, RREQ packets carry the list of all nodes through which the packet has propagated from source node. All nodes cache RREQ packets in their buffer. When node A originates the RREQ packet, it will first be received by all its neighbors. Nodes receiving the RREQ check if they are the destination. If not they check whether they have path to that destination. They append their ID in the RREQ packet and forward it onwards. Destination may receive such RREQ messages from all its neighbors.

Any intermediate node having path to the destination node, replies back to the RREQ message (does not propagate it onwards). Route reply message (RREP) is unicast back to the source node along the path indicated in the RREQ. All nodes in the path cache the newly learned route in their route cache. As the destination node may receive multiple RREQ messages, it may choose to respond to one or to respond to all the RREQs. Recall that in AODV the destination responds to the shortest or the first RREQ it receives instead of responding to all RREQs.

Route Maintenance process: DSR does not explicitly require route acknowledgment from neighbors. Consider a scenario where the route to a destination D from the source node S is $A \rightarrow B \rightarrow C \rightarrow E \rightarrow D$. After a period of time if node C detects that link to node E is broken or it may not able to establish communication with it, it originates route error (RERR) messages back to the source node A. First, DSR tries to salvage the path by itself using local repair techniques. If it fails, it unicasts the RERR message and deletes the path from its cache. Every intermediate node tries to salvage the path before it forwards the RERR messages.

Apart from the basic route discovery and route maintenance process DSR includes many optimizations to reduce routing overheads. DSR uses promiscuous mode (by listening even to packets that are not intended for them) to learn some new routes from their neighbors. DSR also employs route shortening technique by originating a gratuitous route reply message. Consider a case where node J originates an RREQ message and finds an path to destination D. Node B which is not in the path may have heard the those packets containing longer path length. Node B can originate the gratuitous route reply back to the source node J informing about the shorter path.

2.2 Need for a Secure Routing

The network infrastructure in traditional wired networks are provided substantial amount of protection against malicious attacks. Thus, in context of routing protocols, the problem of routers advertising false information is very unlikely. In contrast, in ad hoc networks where every node acts as a router, the issue of how much one could trust other routers to provide accurate routing information, and the consequences of some nodes deliberately providing inaccurate information, have to be considered [58].

Most of the original routing protocols did not take security into consideration. All protocols we have seen thus far (DSDV, AODV and DSR) implicitly assume that all nodes are trusted and will strictly adhere to the routing protocol. In practice malicious nodes may attempt to disrupt routing operations to create instability of the network. They may advertise misleading information, may not participate in the network all the time or they can

cause denial of service to some other nodes in the network. Attacks that can be perpetrated by malicious nodes are explained in the Section 2.4.

2.3 Secure Routing Protocols

Current secure routing protocols can be broadly classified into two categories. In the first category are schemes which employ cryptographic authentication. In the second category are schemes which use first-hand and second-hand observations to construct trust models, and perform routing decisions based on such models.

2.3.1 Authentication

Authentication is required by every node in the network to know whom they are communicating with. Authentication is the basic requirement in a secure routing protocol to avoid impersonation or spoofing of a nodes ID. Authentication may take various forms and will be discussed in Chapter 6. Authentication may aid routing protocol in detecting inconsistencies in the routing process. Some protocols in this category include Ariadne [7], SRP [23], SAODV [24].

Ariadne Ariadne [7] is a secure routing protocol based on the dynamic source routing (DSR) protocol. Ariadne employs a hash-chain based scheme, TESLA [25]. In Ariadne, source and destination are assumed to share a secret. Ariadne assumes some mechanisms to disseminate the commitment values of the TESLA hash chain of every node to all other nodes. Ariadne, assumes all links to be bidirectional. No intermediate node can respond to a route request - only the destination can. Every intermediate node forwarding the RREQ

inserts a hashed message authentication code (HMAC). The key used for calculating the HMAC is a value from the hash chain that is guaranteed to be undisclosed at least till the instant the destination receives the RREQ.

Once the destination signs the path with a HMAC (employing the secret it shares with source) intermediate node cannot modify the path and the HMACs they had introduced in the forward path. The intermediate nodes reveal the hash value used for computing HMACs in the RREQ during RREP. Such values are preimages of the commitments. This makes it possible for the source node to verify the authenticity of every node in the path by verifying the HMAC appended by every node using the revealed hash value. The revealed hash value is also verified to ensure that it repeated hashing of the value yields the commitment. Thus nodes cannot be inserted in the path by an attacker as the attacker cannot produce valid pre-images which were undisclosed until the time the destination receives the RREQ for such nodes.

Ariadne also employs an elegant technique based on hashing a value chosen by the source at every hop by nodes in the path to ensure that nodes cannot be deleted from the path. Thus TESLA (or HMACs based on TESLA) prevent insertion attacks which can be detected after the end of the reverse path. The per-hop hashing strategy permits the destination to detect attacks involving deletion of nodes in the path by an attacker.

Ariadne uses network wide secret to keep the external attackers from participating in the routing. Ariadne defines various classes of active attackers using active x-y model. Active 0-x represents x external attackers who have not been provided with access to any secret. Network wide secret is used to overcome Active 0-x attackers. Active 1-x

represents an attacker who has compromised all the secrets from a device provided such secrets to x colluding nodes. An active y-x attacker has exposed all secrets from y devices which are provided to x colluding attackers. Ariadne was designed to resist multiple non colluding Active 1-1 attackers [70].

SAODV: Secure AODV [24] protocol employs a per hop hashing technique to protect mutable RREQ fields (fields which change during RREQ propagation) and digital signatures of the RREQ source for protecting non mutable fields. The non mutable fields in the RREQ includes a commitment to a random value X chosen by the RREQ source in the form $h^{h_c}(X)$ where h_c is the maximum number of hops the RREQ can be relayed. The value h_c is also indicated in the non mutable fields of the RREQ (the fields specified by the RREQ source, which do not change when the RREQ is propagated). The source releases X along with the RREQ. The nodes one hop away from the source are expected to hash the value X once and increment the mutable hop count value to one. A node two hops away similarly replaces $h^1(X)$ with $h^2(X)$ and sets hop count to 2. Thus a node i hops away from the source receive an RREQ with value $h^{i-1}(X)$ and is expected to forward $h^i(X)$ indicating a hop count i.

Secure Routing protocol (SRP): Secure routing protocol [23] assumes source and destination share a common secret. During route requests intermediate node do not append any authentication. They just append their own ID and forward the RREQ. Once the destination receives this RREQ, it signs RREP with a HMAC using the shared secret it maintains with the source node. SRP does not allow any intermediate node to reply to

the route request [26]. Since, it does not offer any authentication of intermediate nodes, malicious attacker can add or remove nodes during route request propagation. SRP is also vulnerable to invisible node attacks where nodes just forward RREQ packets with out appending their ID [27].

2.3.2 Observation and Trust Models

By observing the behavior of neighbors it may be possible for nodes to detect malicious activities. Consider an example where node A and B are neighbors. Node A sends some information that has to be relayed by B after attaching some information regarding B. Node B just forwards the packet that A has sent without any modifications (or A observes that B has violated the protocol). In such a scenario, node A can disregard packets or information sent by node B in the future. Observation based secure routing techniques include OCEAN [28], Path-rater and Watchdog [29, 30].

Watch dog and Pathrater: Watch dog [29] monitors the neighbors by listening promiscuously to make sure that nodes follow the protocol. Path rater is the technique that can be used to calculate the metrics along the path. Path rater metrics can be used to calculate the trust values or reliability of the path. However, there are many reasons which can reduce the effectiveness of techniques based on observations. Congestion and collision can prevent overhearing and thus introduce errors in judgment.

Trust Models: Many researchers have investigated suitable models of trust and propagation of trust in ad hoc networks [30] - [33]. The central idea behind the use of such trust

models is to promote establishment of routes with more trusted nodes. The basis of such trust models stem from observations (both first and second hand observations) made by nodes during the course of their participation in the ad hoc network.

CONFIDANT [32] is an example of a secure routing protocol (based on DSR) employing trust based metrics for evaluating the neighbors. CONFIDANT contains the following elements: a monitor for watching neighbors, a reputation system for node rating, a path manager for evaluating path, and a trust manager to take of trust based decisions. When a malicious activity is detected by a node using neighbor watch, reputation system is triggered and entries for the neighbor is changed. Path manager deletes the path if some level of suspicious value exceeds on the path. Trust manager just sends out warning to all nodes regarding the path and malicious nodes.

OCEAN [28] (observation based cooperation enforcement in ad hoc networks) maintains a reputation score for its neighbors only based on its own observation. It uses promiscuous mode to observe its neighbors directly. It does not evaluate a node based on evaluation of other nodes (second hand observations). OCEAN needs to observe behavior of nodes for long periods before it can build the reputation table [34].

The fundamental prerequisite for effective use of such trust models is the ability to authenticate nodes. Without authentication, impersonation of a node is possible. Many works in the literature that employ trust models have however disregarded the need for authentication.

2.4 Attacks on Routing Protocols

This section describes various classification of attacks that can be mounted on any routing protocol. Attacks can be launched by internal attackers (who are authorized to participate in the network) or external attackers. The attacks themselves can be classified into *active* attacks and *passive* attacks [30], [35]- [38]. Ultimately, the main goal of most attacks is for denial of service to other nodes. Active attacks do so by violating rules while taking part in the routing protocol. Passive attacks do so by *not* taking part in the protocol.

2.4.1 Active Attacks

Active attacks involve injecting fabricated routing messages or maliciously modified messages during the routing process [35]. Active attacks intend to disrupt the routing protocol, and if possible, render the network unusable. Some specific examples of active attacks include [35]

1. jamming the physical layer
2. violating the medium access protocols to render the medium unusable by other nodes
3. injecting deliberately misleading information
4. inappropriate modification of routing messages introduced by others

There is very little that affected nodes can do to recover from attacks that violate medium access control protocols. Fortunately, the ill effects of such attacks are constrained to small region - only neighboring nodes (within the range of the attacker) are affected. While physical layer attacks also have localized effects, they can be mitigated to some extent by

cryptographic techniques, where cryptographic keys used for anti-jamming measures like code-division multiple access (CDMA) [40] or frequency hopping.

Through active attacks that involve illegal modifications to routing packets malicious nodes can redirect routing packets and cause denial of service or can disrupt the normal routing functionalities. Modification attacks can be done by using falsified information on the routing packets. Some examples of such attacks are as follows.

1) Modifying Sequence Number: Every routing packet might contain sequence number to avoid flooding in the network. Each node may use this sequence number to check whether it has really received the packet before. Malicious node can advertise some values higher sequence number so that all the packets can be made to divert through it. This may prevent legitimate route being established [35].

2) Modifying Source Routes and Hopcount: Hop count is generally used to check the length of the route. When a node has two or more paths to the destination it might choose the path with lower hop count. Malicious node might modify the hop count and send falsified information. Actually, the path may be longer but it might force all other node to use the path since the advertised path has lower hop count values. In DSR protocol, malicious node to be in the path may launch this attack and may later attempt DOS attack [37].

3) Spoofing or Impersonation Attacks: Spoofing attacks in which a malicious node tries to pose itself as some other node and involve in the network activity. Malicious node can send false route replies or send route error messages which are not the part of original routing process. This may cause network instability and soon may cause network

to collapse. Spoofing and impersonation attacks can be dealt by using some cryptographic mechanisms by which it can be controlled [37].

4) Rushing Attacks: Rushing attacks target protocols like DSR and AODV that employ flooding, which in turn uses some technique to suppress duplicates. An attacker may disseminate route requests packets quickly through the network and it reaches faster before the legitimate route request packet arrives there by suppressing good route requests and also achieve source node from being establishing a route to the destination node [39].

5) Blackhole Attacks: Whenever a malicious node receives a route request, it sends a bogus route reply to the source node stating it has the path to destination. This is done in order a malicious node to be in the path between source and destination. Once the malicious node make sure that it is in the path between source and destination, it can launch DOS service such as drop the incoming packets, forward only selected packets and can eavesdrop the communication. Blackhole attacks are difficult to contain. This is can be detected by using promiscuous mode to some extent [35, 36, 37].

2.4.2 Passive Attacks

In passive attacks, [38] the attacker does not actively interfere with the subnet. The intent of the attacker may be to learn the subnet topology, perform traffic analysis, and if possible to eavesdrop on exchanges of data between nodes. It is very difficult to identify passive attackers in the subnet [36]. Passive attacks may not really disrupt the routing process. Passive attackers may present internally or externally in the network. Passive

attackers do not modify the routing messages, so its presence is really difficult to identify. Eavesdropping can be mitigated by designing suitable cryptographic mechanisms.

CHAPTER 3

LIMITATIONS OF CURRENT SECURE ROUTING PROTOCOLS

Two fundamental strategies employed by secure routing protocols are i) monitoring neighbors to detect inconsistent behavior; and ii) cryptographic authentication of routing data. The purpose of monitoring is ultimately to restrict the participation of nodes which are suspected of malicious or selfish intents. Cryptographic authentication is a prerequisite for monitoring. Unless a node A which receives a packet from a node "claiming to be B" can verify that the sender is indeed B, it is not possible for A to come to any conclusions regarding the "behavior of B."

In this chapter we discuss some limitations common to several secure routing protocols.

3.1 One-Way Links

Some routing protocols like temporally ordered routing algorithm (TORA) [41] rely on an underlying mechanism like Internet message encapsulation protocol (IMEP) [42] to identify a reliable delivery neighborhood (RDN). On the other hand, some protocols like DSR do not rely on IMEP. The original version of DSR did *not* require the assumption of bidirectional links. However, most secure extensions of DSR [7, 23, 43] *rely* on the assumption that all links are bidirectional. Unfortunately, such secure extensions simply

assume that all links are bidirectional even while they do not possess any explicit mechanism to *ensure* this requirement.

One common (but incorrect) rationale [43] provided for this justification is that the underlying medium access control (MAC) protocols take care of this requirement by employing a handshake, where the sender sends a short request-to-send (RTS) packet and the receiver responds with a clear-to-send (CTS) packet. However, such exchanges can be used only for *unicast* exchanges that follow the RTS/CTS handshakes (the RTS and CTS packets explicitly identify the sender and a unique receiver). For route request (RREQ) packets that are meant for all nodes within range, such handshakes cannot be used. Thus, an RREQ packet sent by a node B can reach a neighbor C even if the reverse link $C \rightarrow B$ does not exist. If C forwards such RREQs such RREQ packets (which are bound to fail to establish a path) can *preempt* other good RREQ packets (as each node forwards only one RREQ), and thus prevent discovery of alternate good paths.

A quantitative illustration of the effect of one-way links on on-demand protocols like DSR and AODV is depicted in Figure 3.1. Simulations where carried out for random realizations of 200 nodes in a square region with unit edges. To simulate one-way links the range of each node was chosen to be uniformly distributed between 0.09 and 0.11 units. RREQ propagation was simulated between every pair of nodes, separated by different hops lengths (X-axis). The plots depict the fraction of successful node pairs (Y-axis) that discover a path free of one-way links.

For protocols like DSR where the source and destination can discover *multiple* paths, the end points are assumed to succeed in their quest (to establish a path) if at least one path

is free of one-way links. For protocols like AODV where the destination responds only to the first RREQ received the first RREQ should be clear of one-way links. The plots labeled MR and SR depict the fraction of successful route request attempts for the scenarios where i) at least one RREQ should succeed (MR) and ii) the first RREQ should succeed (SR). The plot labelled IDEAL depicts the scenario where some proactive mechanism is employed to inhibit the propagation of RREQs over one-way links.

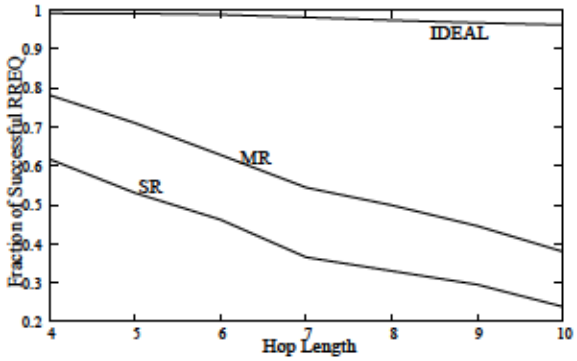

Figure 3.1

Effect of one-way links

3.2 Link Level Authentication

While the danger of unauthorized tapping does exist even in wired networks, this can be easily addressed by establishing a shared secret between the end points. Establishing a shared secret between A and B at two ends of a wired connection is comparatively trivial as key distribution schemes used for such purposes do not need to scale well. Such keys can even be set up manually. Furthermore, even if the overheads for establishing such

secrets are high, it is acceptable as 1) the devices are typically not resource constrained; and 2) the established secrets can be used for very long durations as the end-points very rarely change.

In contrast, establishing a shared secret between two neighbors in an ad hoc subnet is more challenging. Firstly, as the neighbors of a node may change rapidly, the key establishment process has to be performed more frequently. Secondly, a node has no *a priori* knowledge of *who* could end up as a neighbor. Thus, node A should be prepared to accept potentially *every* node in the network as its neighbor. This calls for a key distribution schemes that support very large (or even unlimited) network sizes, that facilitate non mediated establishment of shared secrets. The resource constraints inherent to MANET nodes make this requirement more challenging to meet.

Many popular secure routing protocols like Ariadne [7], SRP [23], SAODV [24] either lack mechanisms, or employ ineffective mechanisms for authentication of neighboring nodes. In the DSR-based secure routing protocol (SRP) [23] only the source and destination share a secret. No mechanism exists for verification of the authenticity of intermediate nodes. Ariadne [7] employs TESLA [25] for authentication of intermediate nodes in the path by the RREQ source. A *network-wide* shared secret, used for encrypting / authenticating all packets sent by every node to keep *external* nodes away, is the only form of link level authentication employed.

In Ariadne a malicious internal node C forwarding an RREQ can insert itself as some node C' in the RREQ. The authentication appended by intermediate nodes can be verified only by the destination (for the scheme in [43]) or the source (for Ariadne with

TESLA). However such RREQs (or RREPs raised in response to such RREQs) which will be dropped by the destination or the source, can still preempt other good RREQs from reaching the destination. In SAODV (where no authentication is required to be appended by intermediate nodes) any node (internal or external) can engage in such attacks. What makes the attack more appealing for attackers is that they face absolutely *no risk* in carrying out such attacks. Without deterrents, the attackers can engage in such attacks unabated.

Figure 3.2 depicts plots illustrating the fraction of node pairs that successfully discover a path free of malicious nodes. Once again the simulations involved random network realizations with 200 nodes, out of which 20 nodes were randomly labelled malicious. For DSR it is assumed that the end points will succeed if at least one path free of the malicious nodes is established (plot labelled MR). For AODV the first RREQ received by the destination needs to be free of malicious nodes (plot labeled SR). To see the extent of suppression of good RREQs, the plot labelled IDEAL depicts the scenario where the 20 nodes do *not* take part in forwarding the RREQ.

3.3 Per-hop Hashing

While the cryptographic authentication appended by a node A is sufficient to convince a verifier that the information does indeed originate from A (assuming no other node has access to secrets of A), it does not provide any assurance that the information provided by A is indeed correct. Some *redundant* information is required to facilitate nodes to gather the *same* information from multiple sources.

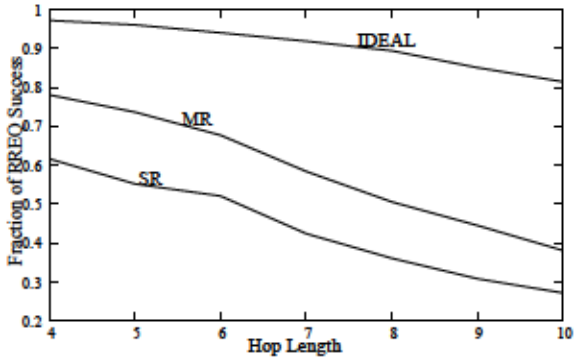

Figure 3.2

Effect of risk-free rushing attacks

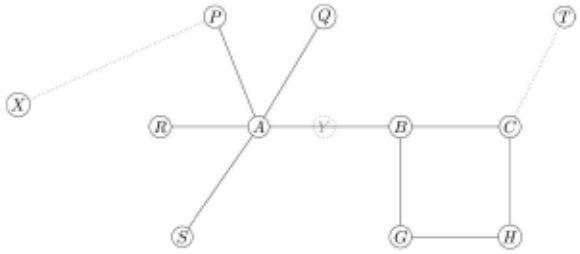

Figure 3.3

Network topology used for illustrations

3.3.1 Carrying Over Authentication

One of the most common strategies for providing this redundancy is by *carrying over* authentication. To illustrate some issues in carrying over authentication, we shall consider the network topology in Figure 3.3. In the description below we use the following notations:

1. $[X, 4]_A$ denotes an authenticated message[1] from A indicating that X is four hops away from A.
2. $[A \leftrightarrow B]_A$ denotes an authenticated message from A indicating that B is a neighbor of A; $[A \leftrightarrow B]_B$ represents an authenticated message from B indicating that A is its neighbor.

Assume that at some t, A broadcasts a message $[X, 4]_A$ indicating that X is four hops away from A. At some time $t_1 > t$ a neighbor B of A broadcasts a message $[X, 5]_B$, which is received by a neighbor C of B. If the message from B is also accompanied by the message from A, viz., $[X, 4]_A$, it would appear at first sight that C has two "independent confirmations" that X is six hops away from A. However, this is not true as it is possible that a node Y exists between A and B; the node Y could have relayed the information sent by A to B along with its distance to X, $[X, 5]_Y$; a malicious B relays the message $[X, 4]_A$ from A instead of the message $[X, 5]_Y$ from Y, and announces a shortened hop-count (or path length) $[X, 5]_B$.

In relaying the message $[X, 4]_A$, note that an implicit claim of B is that "A is my neighbor." The problem is that this claim is *not* verifiable by B's neighbors. One approach is for A to authenticate this information with a message $[A \leftrightarrow B]_A$, which should also be

[1] An authenticated message includes the message and a verifiable authentication token like a HMAC or a digital signature.

broadcast along with the message $[X, 4]_A$. More specifically, in a scenario where A has multiple neighbors (P, Q, R, N, and B) at time t, A should create an authenticated list of all its neighbors.

The authentication appended by A is intended for verification by all two-hop neighbors of A. As A does not necessarily know the identities of all potential verifiers (all two-hop neighbors) a source authentication scheme (like a digital signature scheme) will need to be used for authenticating the message. Furthermore, as the neighbor-list of A may change rapidly, the appended authentication token should be deemed valid only for a small period, and therefore, will need to be refreshed frequently. Obviously, such an approach can introduce substantial overhead, especially in highly dynamic subnets.

An elegant way to address this problem, viz., providing verifiable proof that B is indeed a neighbor of A, with relatively low overhead, is the per-hop hashing strategy.

3.3.2 Per-hop Hashing

Instead of explicitly naming all its neighbors and periodically providing authenticated neighbor lists, node A provides a "per-hop hash value" *only* to its neighbors. That a node (say B) has access to the value provided by the previous hop (A) is "somehow demonstrated" to downstream nodes. The specifics of *how* this is demonstrated, and to *whom*, are however protocol dependent.

For example, in Ariadne [7], this proof is demonstrated only to the destination. The per-hop hash is seeded by a value β_0 which is privy to both the source S and and destination T (β_0 is derived as a one-way function of a secret shared K_{ST} between the source and the

destination). The source S broadcasts this value to all its neighbors. A neighbor A of S replaces the value β_0 with the value $\beta_1 = h(\beta_0, A)$ and broadcasts β_1 to all its neighbors. A neighbor B of A similarly replaces β_1 with $\beta_2 = h(\beta_1, B)$, and so on, at every hop.

When the destination T receives a value β_n with n nodes in the path, it can compute β_0, recursively compute β_n, and verify that no nodes have been removed from the path. In a scenario where B is *not* a neighbor of A, note that B cannot claim that A is its neighbor (by removing Y from the path indicated), as B does not have access to the per-hop hash broadcast by A. Variants of the per-hop hashing strategy are also used in secure extensions of AODV [24] and DSDV [44] to prevent attacks involving shortening of paths. Unlike the per-hop hashing scheme in Ariadne, for the schemes in [24] and [44] computing the per-hop hash does not include the identity of nodes.

While per-hop hashing is an efficient strategy, its security rests on the assumption that *only* neighbors of A have access to the per-hop hash value broadcast by A. Sending the per-hop hash value in the clear for the benefit of all nodes within range (with the assumption that nodes that are not neighbors cannot hear the value anyway) is obviously a loop-hole that can be exploited by attackers. As an example, consider a scenario where a malicious node C simply *pretends* to be out of B's range. Assume that B relays a request originating from S indicating a path (A, B) and a per-hop hash value β_2. Now C waits for the RREQ to be relayed along another path (A, B, G, H). However, with access to β_2 transmitted by B, C has the ability to remove its immediate upstream neighbor H, or both G and H, from the path. In this particular instance C would obviously to remove H from the path, as removing both G and H is tantamount to *admitting* that C can hear B.

CHAPTER 4

MITIGATING EFFECTS OF ONE WAY LINKS

This chapter provides a qualitative and quantitative study of the effects of one way links on the efficiency of route discovery, the implications of security on one way links in route discovery. A simple proactive approach amenable to existing secure DSR-like protocols is proposed to mitigate problems arising due to one way links.

4.1 Wireless Medium

It is well known that the choice and limitations of the medium access control (MAC) layer will have significant effect on the efficiency and security of any ad hoc routing protocol. Wireless MAC protocols like MACA or MACAW [45]- [47] are most popular candidates for unicast messages in broadcast channels. For example, to send a packet to B, the source A will send a RTS (Request to send) packet and wait for a CTS (Clear to send) packet from B. Any node in the vicinity of B which hears a CTS packet (apart from A) would refrain from transmitting for a duration which will be indicated in the CTS packet.

While the transmission is unicast, it is still typically omni-directional. Thus it is possible for any node with in the range of the transmitter A to overhear even unicast transmissions. However RTS/CTS approach cannot be used for transmitting broadcast messages

(for example RREQ packets from A intended for all other nodes within its wireless range) and for transmitting the RTS/CTS packets themselves.

Omni directional wireless broadcasts have the advantage to reach multiple nodes efficiently with a single transmission. At the same time, efficient measures for reducing collisions and ensuring reliable packet delivery to intended nodes is more challenging. This is especially due to the presence of one way links.

4.1.1 One-Way Links

While node A may be within range of node B, due to various reasons like differences in transmission range, receivers sensitivity and aging, node B may be out of the range of $A's$ transmission, even if all transceivers have been manufactured to meet the same specifications [48]-[53]. The other scenarios where a link can be one way is due to some selfish behavior of the node. Some nodes may attempt to conserve their battery by reducing their transmission range.

The original DSR protocol offers some local repair mechanisms to fix paths that include one-way links. However in most secure extensions of DSR and AODV such mechanisms are not possible. This can lead to repeated failures in finding good paths even when paths exist. Furthermore, ignoring the possibility of unidirectional links can also have adverse implications on the security of route discovery process.

The ill effects of unidirectional links on the performance of various ad hoc routing protocols, and solutions to overcome this problem have been studied by various researchers [48] - [53]. In EUDA, [49] each node advertises transmission power and receiver's sen-

sitivity to enable nodes receiving RREQs to determine the range to the transmitter and thus determine if the reverse path exists. Obviously, malicious nodes that seeks to exploit one-way links can easily advertise wrong values or remain silent.

In [54] nodes maintain a neighbor table. For example, when node B hears a transmission from node Y, node Y is added to B's neighbor table. Thus in scenario where $C \rightarrow B$ is one way, C's neighbor would include B, while B's routing table will not include C. Nodes periodically advertise their neighbor tables. Thus a common neighbor of B and C can realize this discrepancy and warn B of existence of C. Obviously, in a scenario where C does not want to reveal the fact that the link is one-way, C will not indicate B in its neighbor table.

Apart from reducing the efficiency of DSR (by mandating repeated RREQs when the first one fails) ignoring the possibilities of unidirectional links also has some implications on the security of the route discovery process. Malicious node can able to engineer attacks either by 1) taking advantage of unidirectional links or 2) even by pretending that some links are unidirectional.

In the rest of this chapter we restrict ourselves to the effect of one-way links on DSR.

4.2 Effects of Asymmetric Links on DSR

In secure DSR schemes such as SRP [23] and Ariadne [7], ensuring integrity of the path relies on the assumptions that 1) RREQ and RREP take the same path 2) the source and destination share a secret. The implication of the first assumption is that the protocols mandate bidirectional paths. The implication of the second requirement is that one way

links discovered during RREP propagation cannot be fixed. Unlike the original DSR where nodes detecting a break in the reverse path can instantiate an RREQ, this is not feasible in the SRP or Ariadne as secure RREQs/RREPs can be accomplished only if the source and destination share a secret a priori. While assuming the existence of such a secret is reasonable for the actual source and destination, it is not reasonable to expect intermediate nodes to shares a secret with source or destination (and thus cannot invoke RREQs to reach the actual source).

First assumption is obviously violated if it is deemed impractical to flood RREQs by individually unicasting it to neighbors. The result is many failed RREPs corresponding to successful RREQs. While it may seem at first sight that the reduction in efficiency may be negligible if the fraction of such one way links are low we show that this is not necessarily true. Also, secure DSR extensions cannot simply afford to assume bidirectional links, proactive approaches are required to identify such one way links and warn downstream nodes to take appropriate action before RREQs are flooded.

4.3 Proactive Measures for Indicating One Way Links

A proactive measure for the link $B \rightarrow C$ which may be one way is for B to explicitly include a warning in the RREQ it forwards, that the link $B \rightarrow C$ may unreliable. The rationale used by B to conclude that link $B \rightarrow C$ is unreliable, will be discussed shortly.

The warning can be realized say by appending a special code λ and indicating such links in the RREQ forwarded by B, as $(S, A, [B\lambda C])$ (instead of just(S, A, B)). In order

to ensure that C cannot delete the warning we can still employ per hop hashing where $\beta_2 = h(\beta_1, [B\lambda C])$.

When C receives such an RREQ, it is expected not to forward the RREQ further. Even if a malicious C disregards this instruction from B, downstream nodes will be honor this request from B. Any RREQ containing both B and C (even if they are separated by many hops) will not be allowed to propagate. Thus if the RREQ contains a warning $[B\lambda C]$, and includes both B and C will not be propagated further.

Note that it is not sufficient just to inhibit RREQs that contain the link $B \rightarrow C$ as any RREQ of the form $(...B, ...C, ...)$ is susceptible to misrepresentations by C. Obviously, a node can easily indicate multiple links to avoid (say by appending $(S, A, [B\lambda C], X]$ to indicate both $B \rightarrow C$ and $B \rightarrow X$ could be one-way).

4.4 Mitigating RREQ Failure

There are many reasons as to why a node B may suggest (by including a warning) that a link as $B \rightarrow C$ or $B \leftarrow C$ as unreliable. Obviously, if the link is truly one way (say, B cannot hear C), B will never get to know the very existence of C unless a common neighbor of both B and C can indicate this possibility to B [54] (however C can still recognize that this indeed even without the help of a common neighbor and drop RREQs from B). It is also possible that the link is bidirectional, but

- C is a malicious node which pretends that it is out of range or
- B is malicious node disseminating misleading information or
- Both B and C or honest but due to collisions in the channel the are not able to conform the existence of bidirectional paths

For three of the four reasons (the link is actually one-way, or C is malicious, or B is malicious) mitigating the damages calls for prevention of propagation of RREQs that include *both* B and C in the path.

4.5 Semi Active Approaches

One of the reasons why DSR is seen as an attractive option (especially for dense deployments) is that periodic exchanges between neighboring nodes are not required - nodes do not exchange periodic "ALIVE" messages to neighbors to maintain link state information. At the same time, this is also the reason why it may not be efficient to unicast RREQ packets to each neighbor (as nodes may have several neighbors). In situations where 1) nodes unicast RREQs or 2) nodes need to identify (and explicitly specify) one way links (as in our scenario, where however we assume that RREQs are not unicast), there is a need for nodes to "know their neighbors".

However, even without active transmissions exclusively for this purpose, it is possible for any node, say F to learn about it's neighbors just by listening to transmissions from neighboring nodes. Thus node F (in due course), will learn that C, L and G are its neighbors. Furthermore, when the local traffic is low (thus preventing nodes from gathering information about its neighbors), it is not a severe disadvantage to send probing messages to solicit responses from neighbors. In other words, instead of mandatory periodic messages to determine the link state information, nodes can simply use such probes when traffic is low.

Thus, each node can maintain a neighbor table [54]. Whenever a node overhears transmission from a new node, a row created for the new node with zero scores and a time stamp. A neighbor C of F will get a positive score id 1) node F sends a broadcast message (for example, A RREQ) and is able to overhear C faithfully rebroadcasting the RREQ after inserting itself on the path 2) F successfully unicasts a packet to C (with RTS/CTS, thus conforming that the link is bidirectional). On the other hand, the neighbor C could receive a negative score when 1) F is not able to perform an RTS/CTS handshake with C, or 2) F does not hear C broadcasting an RREQ (with the same sequence number), or 3) F hears C broadcasting an RREQ with longer path length.

In addition, nodes can still use proactive advertisements of transmission power/receiver sensitivity as in [49], as this could still help in scenarios where nodes are not malicious (and this does not call for extra transmissions-just a small increase in bandwidth of each transmissions). Additionally, the nodes can also take the received signal strength, and rate of fluctuation of signal strength into account for determining the weights of positive and negative scores.

4.6 Classifying Links

Whatever technique is used to classify the path to neighboring nodes as oneway or bidirectional, they will be susceptible to judgment errors, where some bidirectional links to be misclassified as one way and vice versa. The links, as seen by any node can be classified into three broad categories

- Confirmed bidirectional
- More likely to be bidirectional

- More likely to be one way

Note that while we can confirm bidirectional links by exchanging RTS/CTS packets, we can only hypothesize that some links may be one way (depending on the positive and negative scores). However, it may be reasonable to assume that "most" links seen by any node will fall under the first category, especially in scenarios where the nodes taking part in the wireless network are designed to be homogeneous. It is only the second and third categories that have to be analyzed further to improve the chance of making a right decision (classifying them as one way or bidirectional).

The specific strategies used for maximizing the a posteriori probabilities (MAP) [55] will depend on numerous factors including local network density, traffic and node mobility. Unfortunately, algorithm that take into take into account a large number factors (which themselves may not be easy) to provide accurate MAP estimates can be complex.

However, we show that even with very simplistic strategies, a significantly higher success rate of RREQs can be realized. More specifically, we consider a pessimistic approach in characterizing links. Links are deemed one-way unless proactively established as two-way. In other words, we group together the categories "more likely to be oneway" and " more likely to bidirectional" into a single group and declare them to be one way. However, we do not drop RREQ packets with warnings. Such RREQs just suffer additional delays before retransmission.

If node E receives an RREQ from some path containing both B and C, and if B has indicated a potential one way path (by appending [BλC]), the RREQ is delayed for some appropriate duration before further propagation. Meanwhile (during this delay period) if E

receives RREQ for same sequence number with a safe path, the earlier RREQ is ignored. Thus paths that could include oneway links are given lower preference over good paths during propagation of RREQs. However, as many links that carry the warning may still be bidirectional, they can still be used as some RREPs that include links with relevant warnings may still succeed. Furthermore, paths that include multiple links with warnings are delayed more than the path that include single warnings.

4.7 Simulation Results

To obtain a quantitative estimate of the effect of such one way links on the failure of RREQs and the effectiveness of different strategies for reacting to such warnings (ignore, drop or delay), we carried out extensive simulations in a square region with unit edges, consisting of 200 randomly placed nodes. Though there are many reasons like small differences in transmission power, receiver sensitivity, aging and local noise level which could contribute to one-wayness of links, in our simulations we assumed that the range of each node was different. More specifically, it was assumed that the mean range is 0.1 units, but the actual range of any node is a uniformally distributed between 0.09 and 0.11 units (10 percentage swing from the mean).

In most of our realizations, each node had 5 neighbors on an average. Furthermore, the number of one-way links N_0 were substantially smaller than the number of bidirectional links N_b. More specifically, $N_b \approx 700$ and $N_0 \approx 70$, on an average. We simulated propagation of many route requests over different number of hops, choosing random source and destination nodes, for 10 different random realizations of the network. For each ran-

dom realization we considered all possible node pairs separated by a certain number of hops (ranging from 3 to 10), which have a link from the source to the destination, but not necessarily a reverse path. The RREQ propagation was modeled for 5 different cases

1. **B-100:** All bidirectional links are identified. Only bidirectional links are used for sending RREQ (the best case scenario).
2. **B-75:** Only 75% of the bidirectional link are identified and used for RREQ propagation. The remaining 25% are deemed one-way (though only a small fraction of them are actually one-way) and not used.
3. **B-50:** Only 50% of the bidirectional link are identified and used for RREQ propagation. The remaining 50% are deemed one-way and not used.
4. **O:** No proactive measure to identify one-way links. All links are used for RREQ propagation, and
5. **S-50:** Only 50% of the bidirectional link are identified and the remaining 50% are deemed one way.

However, RREQ's that contain paths deemed one-way (while many of which are actually bidirectional) are imposed additional delays. In other words, for case O warnings are not employed, or alternately, warnings (if employed) are *ignored*. In cases **B-100, B-75, B-50** RREQ's with relevant warnings are *dropped*. For case S-50 RREQ's with relevant warnings are *delayed*.

The results are presented in Figure 4.1 terms of the ratio of successful RREQ's to the total number of node pairs chosen. A total of over 250,000 RREQ's (corresponding to randomly chosen node pairs) were simulated, ranging from 25,000 to 40,000 paths for each hop length. In our simulations, collisions and node movement are ignored. We assume that the transmission delay is negligible. Thus all nodes at a distance of t hops, from the source will relay the RREQ at time t units. However, with n nodes ready to send

RREQ at time t we *process* the n RREQ's in a random order, sequentially, among the n nodes.

4.7.1 Dropping RREQs with Warnings:

Ideally, we would like to identify all bidirectional links (and thus ensure that all links identified as one-way are indeed one-way). In practice, the percentage of bidirectional links that can be identified will depend on numerous factors that perhaps cannot be reasonably modeled by simulations. It is for this reason we have opted to take the approach of illustrating three different scenarios where implies 100% (B-100), 75% (B-75) and 50% (B-50) of the actual bidirectional links are recognized as bidirectional. In a scenario where only x% of the bidirectional links are identified, the remaining (100-x)% are classified as one way (even though only a small fraction of them are actually one-way).

Obviously decreasing x leads to lower network connectivity which will result in larger number of failing RREQs. Clearly (for the particular choice of parameters), if we can identify 75% of the bidirectional links (and prohibit use of other links), we can still do reasonably well (not much lower than the best case scenario B-100). However, when x reduces to 50% many RREQs fail, as indicated by plot for B-50.

4.7.2 No Proactive Measures:

Plot O in Figure 4.1 corresponds to the case where no proactive measure is taken to prevent use of one-way links (or scenarios where the possibility of one way links, which exist, are ignored). In this case only a small fraction of RREQs (ranging from 71%

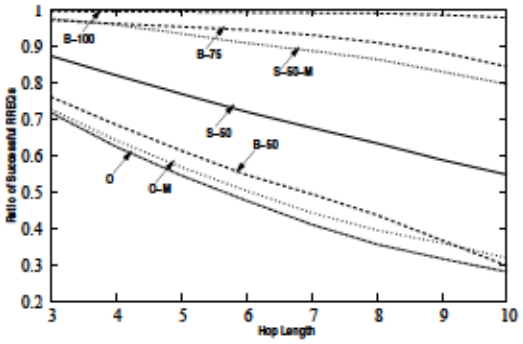

Figure 4.1

Ratio of successful RREQ's with different approaches

for 3 hops paths to 32% for 10 hops paths) were successful, even while in almost every situation a good path *did indeed exist* as obvious from plot B-100). We also considered an extension of scenario O, labeled O-M, where the destination responds (or invokes RREPs) to all RREQs that it receives. In this case, the RREQ is deemed successful even if one of the RREPs reaches the destination (or at least one path did not include one-way links). As can be seen from plot $O - M$, even under such scenario the number of successful RREQs improve only marginally - supporting our intuition that good paths may be preempted by paths with one way links. For the particular choice of parameters , even identifying and using 50% of bidirectional links (B-50) performed better than not taking any proactive measure, thus clearly indicating the need for such measures.

4.7.3 Delaying RREQs with Warnings:

Intuitively, if we can mitigate preemption of good paths by paths that are suspect, we can expect to do significantly better. This was the motivation for delaying RREQ's with warnings, in scenario S-50. The plot labeled S-50 corresponds to the case where only 50%

of the bidirectional are identified (as in B-50). RREQ's with warnings are delayed by a multiple of the hop period (to mitigate preemption). More specifically, an RREQ with w relevant warnings is delayed by w hop periods, at each hop. As can be seen from plot S-50, a substantially larger fraction of RREQ's succeed with this approach. The plot labeled S-50-M (analogous to plot O-M) also takes into account success by responding to multiple RREQ's. Note that while this approach did not yield any substantial improvement from scenario O to O-M (due to preemption of good paths), as expected, the improvement in this case (S-50-M vs S-50) is indeed significant.

4.7.4 Path Lengths

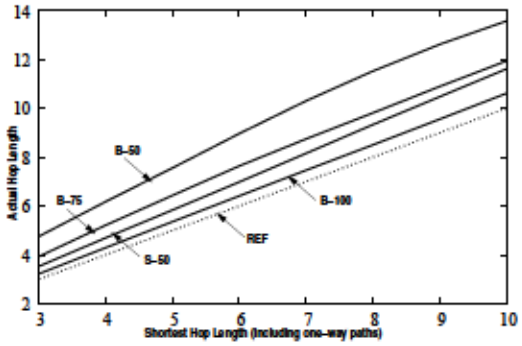

Figure 4.2

Actual path lengths of successful RREQs

Figure 4.2 indicates the actual path length of successful RREQs L_A for nodes which are L_0 (3 to 10) hops away, taking also one way links into account. The dotted line (labeled REF) where $L_0 = L_A$ serves as a reference. The best that we can do is B-100 where we

identify all (100%) bidirectional paths. If we identify only 75% (B-75) obviously some RREQ's will end up using longer paths, The situation gets worse for B-50. However the average path length for the strategy S-50 is even less than B-75. Thus apart from improving the success rate of RREQ's, even the average length of successful RREQ paths are lower.

Obviously, proactive measures for indicating one-way links will increase the size of RREQ packets. If each node appends (on an average) n_w such warnings, the bandwidth of an RREQ packet that has traversed n_h hops will be increased by a factor $n_h n_w b$ bits where b is the number of bits in the ID of each node. In our simulations, for S-50 (and B-50) each node appends (on an average) about 3 additional IDs (or the effect is the same as increasing the size of the ID by a factor 4). This bandwidth overhead may be small compared to size of signatures appended by nodes. The advantages on the other hand viz., significant increase in the success rate of RREQ's (and thereby eliminating the need for repeated RREQs) are compelling.

4.7.5 Rationale for the Simulations

The primary motivation behind our simulations is to be demonstrate that unless proactive measures are employed to inhibit use of one-way paths, the result is significant reduction in the efficiency of DSR. We deliberately chose our parameters (number of nodes, range of each node and deviation from mean range) to result in a *small fraction* of one way links, to highlight the point that even when the fraction of one way links are small, the deterioration in performance can be *significant*. Note that while (on an average) 10%

of the links are one way, only half the links will affect RREQ's (on an average) half the links will inhibit *forward* propagation of RREQs.

Even while only 5% of the links contribute to failure of establishing paths, our simulations clearly demonstrate that the ill-effect of one-way links is still substantial enough and cannot to be ignored.

We opted to perform our simulations "from scratch" without making use of more commonly used network simulation tools like ns2 [56], primarily due to the need to specify different ranges for each node (which is more cumbersome with ns2). The effects of collisions were ignored, as the intention is to serve as a comparison of different strategies, and collisions will affect all strategies.

CHAPTER 5

PRIVATE LOGICAL NEIGHBORHOODS

There are many issues like correctness, simplicity, optimality, robustness and fairness have to be addressed by MANETS [1]. Apart from these issues, MANET routing protocols have to address some additional constraints like

1. the resource constrained nature of mobile devices
2. more rapid change in topology due to mobility and issues specific to wireless (as opposed to wired) links.

It is thus not surprising that the original incarnations of ad hoc routing protocols ignored other practical considerations like the existence of non cooperative or malicious routers. Early attempts at ad hoc routing protocols implicitly assumed that all nodes will strictly adhere to rules.

Many secure ad hoc routing protocols have been proposed later which strive to enforce co-operation by reducing the "degree of freedom" of participants to violate rules. However, while ad hoc routing protocols that ignore security issues typically address salient differences between wired and wireless links, several popular secure routing protocols in the literature have unfortunately ignored such differences. Neglecting these important differences leads to many potential security holes in the protocols.

From a security standpoint, a primary difference between wired and wireless networks is the fact that in wired networks a router is well aware of all routers to which it is phys-

ically connected. While the danger of unauthorized tapping into wires do exist in wired networks, such issues can be easily addressed by establishing a shared secret between the end points. Establishing a shared secret between A and B at two ends of a wired connection is comparatively trivial as key distribution schemes used for such purposes does not need to scale well. Furthermore, even if the overheads for establishing such secrets are high, it is acceptable as 1) the devices are typically not resource constrained and 2) establish secrets can be used for very long durations as the end points really change.

In wireless networks, there is no way for a node A to determine the list of all entities that can hear a packet sent by A. Furthermore, establishing a shared secret between two neighbors A and B is more challenging. Firstly, as the neighbors of a node may change more rapidly the key establishment process has to be performed more frequently. Secondly, a node A has no a priori knowledge of who could end up being its neighbors. A node A should be prepared to accept potentially every node in the network as its neighbors. This calls for a scalable key distribution scheme that facilitates non mediated establishment of shared secrets. The resource constraints inherent to MANET nodes make this requirement more challenging to meet.

Arguably, the first step toward securing MANET routing protocols is the inclusion of some proactive security features to eliminate the difference between wired and wireless networks. Such an approach is especially important in many emerging hybrid networks which may consists of nodes interconnected by wired and wireless links. Following this, we can afford to ignore the differences between wired and wireless links and then pro-

ceed to identify other strategies for securing routing protocols to enable efficient operation under the presence of non cooperative nodes.

5.1 Enforcing Private Logical Neighborhoods

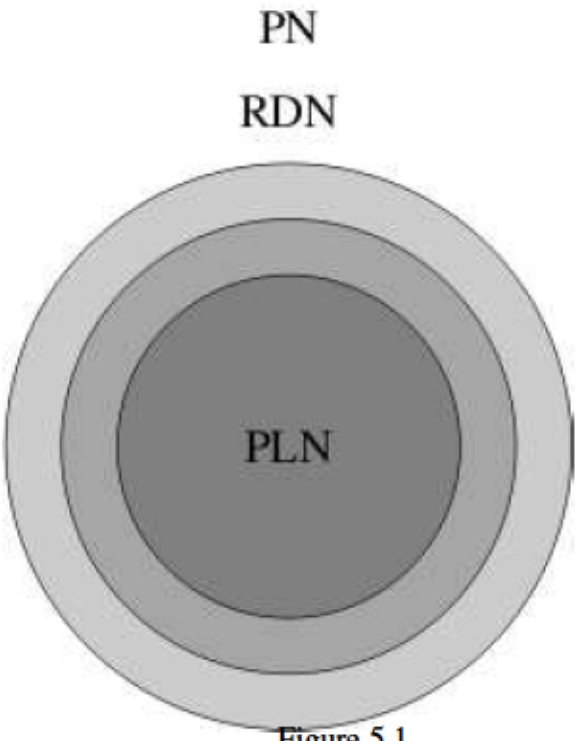

Figure 5.1

Relationship between physical neighborhood (PN), RDN, and PLNs

The reliable deliver neighborhood (RDN) of a node A is a subset of physical neighbors of A with which the existence of bi-directional links has been confirmed. The logical neighborhood of a node A may in general consist of only a subset of nodes in its RDN.

Irrespective of the nodes that may actually be within the hearing distance of A's transmissions, the node A can explicitly specify a subset of such nodes that will be *inducted* into the PLN of A. All other nodes in the physical neighborhood will not have access to the packets sent by A.

When a PLN is imposed, a node A entering an ad hoc subnet sends a probe $[A, c]$ (where c is a randomly chosen challenge) to determine other nodes within range. A node B in the neighborhood responds with $[B, A, K_{BA}(c)]$, where $K_{BA} = K_{AB}$ is a secret that is computable only by A and B. In general, the node A may receive one response from every neighbor within the reliable delivery neighborhood (RDN) of A. Let us assume that A receives responses from 3 nodes X, Y and Z in its RDN. Node A then chooses a random one-hop group secret G_A and broadcasts individual encryptions of the secret as $[X, Y, Z, K_{AX}(G_A), K_{AY}(G_A), K_{AZ}(G_A)]$ to its neighbors. In response, A's neighbors X, Y, Z are expected to send their respective one-hop group secrets (G_X, G_Y, G_Z) to A and disclose their respective user names and public keys.

Subsequently, every packet broadcast by a node A is encrypted with the secret G_A. The packet is prepended with the identity A in the clear to permit receivers to realize that the rest of the packet should be decrypted using the secret G_A. The PLN of a node A is (in general) a subset of nodes in the RDN of A. Node A explicitly invites some nodes into its PLN by providing a secret G_A individually to each node. In a scenario where A has provided its PLN secret G_A to node X, but X did not send its PLN secret G_X to node A (or X did not accept A into its PLN), node A will eject X from its PLN during the next broadcast by A - which is encrypted using a new PLN secret $G_A{}'$, and the secret $G_A{}'$

conveyed only to nodes that A desires to retain in its PLN (and possibly other nodes that A desires to induct into its PLN). Thus, the PLN secret of A is updated whenever a node leaves the PLN (or is ejected by A from its PLN).

The three major limitations described in Chapter 3 can be effectively addressed by enforcing a PLN:

1. As neighbor authentication is implicitly provided simple attacks involving spoofing identities can be thwarted.
2. A node B can simply cut off a neighbor C if it suspects that the link $B \rightarrow C$ or $C \rightarrow B$ is not reliable, by providing a new secret to all other nodes in its PLN. A node C which pretends that it is out of the range of B will not be inducted into the PLN of B.
3. nodes that are not *explicitly* invited into the PLN of B will not gain access to the per-hop hash value sent by B.

Eliminating one-way links and attacks that exploit the lack of link level authentication can result in substantial improvement in the success of successful RREQs. Note that without link-level authentication attackers can afford to carry out rushing attacks by forwarding ill-constructed RREQ packets without facing any risk of being identified. However, if a PLN is imposed, attackers face the risk of being identified by neighbors. Furthermore, establishing a PLN is *mandatory* in scenarios where per-hop hashing strategy is used. More specifically, the per-hop hashing strategy implicitly demands a mechanism to ensure that the per-hop hash is privy only to intended neighbors.

5.2 KDSs for Pairwise Secrets

The fundamental requirement for establishing a PLN is a light weight key distribution scheme that supports ad hoc establishment of pairwise secrets for reasonably large scale

networks. It is important to note the distinction between the scale of an ad hoc subnet and the scale of the network. The network scale N is the number of entities who are eligible to take part in ad hoc networks. The eligibility is afforded to such participants by the key distribution center by providing them with secrets or signing their public key certificates. The network scale could be millions or even billions. The subnet scale N', the number of nodes taking part in some ad hoc subnet, will however rarely exceed a few thousands or even hundreds. It is well recognized that ad hoc subnets do not scale well [57]. However as any of the N eligible nodes can become a neighbor of some node A, the node A should be able to establish a shared secret with each of the N nodes.

Scalable key distribution schemes for ad hoc establishment of pairwise secrets between any two nodes of large networks can be categorized into certificates based asymmetric schemes, ID based asymmetric schemes, and key predistribution schemes which employ only symmetric primitives [71]. Asymmetric certificates based schemes call for large bandwidth overheads for exchanging certificates and computational overheads for performing asymmetric computations to compute a pairwise secret. While bandwidth overheads may be tolerable for purposes of exchanges with neighboring nodes, the high computational overheads also render them susceptible to denial of service attacks.

5.2.1 Key Predistribution Schemes

Key predistribution schemes which employ only symmetric primitives can be broadly classified into non scalable schemes and ID based scalable schemes [71]. Non scalable schemes require $\mathbb{O}(1)$ computational overheads and $\mathbb{O}(N)$ storage overheads. For exam-

ple, for the basic KPS for a network of N nodes represented by $i = 1 \cdots N$, the KDC can choose a master secret K. The node i is provided with $N - 1$ secrets

$$K_{ij} = h(K, i) \oplus h(K, j) = K_{ji}, j = \{1 \cdots N\} \setminus \{i\}.$$

While the $\mathbb{O}(N)$ storage requirements limits the suitability of non scalable KPSs, storage is an inexpensive resource for any conceivable device that could take part in an ad hoc network. Given that 1 million 64-bit secrets call for a mere 8 MB of storage, even network sizes of several tens of millions are easily realizable using the basic KPS.

If unlimited network sizes are called for, scalable KPSs are viable alternatives. Scalable KPSs can achieve unlimited network size N by permitting *susceptibility to collusions*. Scalable ID-based KPSs, which employ only symmetric cryptographic primitives consist of a KDC who chooses a set of P secrets $\mathbb{S}$ and an *unlimited* number of entities with unique IDs. The set of $k \leq P$ secrets assigned to entity A is $\mathbb{S}_A = F(\mathbb{S}, A)$, where $F()$ is a public function. Two entities A and B (with secrets $\mathbb{S}_A$ and $\mathbb{S}_B$ respectively) can discover a pairwise secret K_{AB} using a public function $G()$ as $K_{AB} = G(\mathbb{S}_A, B) = G(\mathbb{S}_B, A)$.

An attacker who has exposed secrets from many (say v) nodes $\{O_1 \cdots O_v\}$ may be able to discover K_{AB} even *without* access to the secrets $\mathbb{S}_A$ (assigned to A) or $\mathbb{S}_B$ (assigned to B) [71]. An n-*secure* KPS can "resist" an attacker who has exposed all secrets from n nodes. Most n-secure KPSs mandate $\mathbb{O}(n)$ computational *and* storage overheads. For such KPSs the $\mathbb{O}(n)$ computational overheads is the bottle-neck which prevents realization of large n. While storage for many millions of secrets is acceptable, a million block-cipher operations (for example) is far from acceptable.

However, some scalable KPSs have been proposed recently [58, 75] for which *only the storage overhead* is $\mathbb{O}(n)$. The computational overhead for such schemes is very low (few tens of block cipher operations), and *independent* of the desired collusion resistance n, or the network size N. The high levels of collusion resistance that can be achieved renders the issue of "fragility" of scalable KPSs irrelevant in practice. For example, for the scheme in [75], achieving resistance to collusions of 100,000 entities pooling their secrets together calls for about 40 MB of storage for each node.

Thus far schemes which employ one hop secrets secrets (for example, [59]) assume that secrets are established by exchanging public keys and performing asymmetric computations. Obviously the overhead for such approaches may render establishment of one-hop secrets impractical. Fortunately, the fact that storage is an inexpensive resource for mobile computers renders scalable light weight schemes for ad hoc establishment of pairwise secrets practical.

5.3 Other Advantages of PLNs

Apart from addressing the three issues that plague many secure MANET protocols, imposing a PLN results in several other benefits.

5.3.1 Cutting-Off Malicious Neighbors

There are many valid reasons as to why a node A may desire to cut-off a *specific* node C (which is physically in the neighborhood of A) from its logical neighborhood. For instance 1) A may have observed consistent misbehavior or non participation by node C,

or 2) suspect a one-way link between A and C or 3) suspect the presence of a semi-active attacker [27] between A and C. Such a suspicion could be triggered if A hears an echo of its own packet and / or if the time delay between RTS / CTS handshakes between A and C seem above normal.

Most existing approaches for mitigating participation by malicious nodes involve propagating accusatory messages regarding misbehavior of nodes. Unfortunately, in most scenarios it may be infeasible for the observer to provide incontrovertible proof of misbehavior of some node, verifiable by all nodes in the network. Thus, such strategies are themselves susceptible to simple denial of service (DoS) attacks where a node can send false accusations to create unnecessary traffic. The ability to cut-off neighbors in the physical neighborhood from the PLN facilitates DoS-free countermeasures to reduce the ill-effects of malicious nodes. Nodes cut off by all neighbors are effectively cut off from the subnet.

5.3.2 Deterring Selfish Behavior

Mandating a PLN can also deter selfish behavior by nodes which would wish to remain silent and not participate in the routing process until there a packet addressed to it. With a logical neighborhood a node will have to be *inducted* into the PLNs of its neighbors before they can monitor traffic. Once inducted, a node C is pressured to participate to the fullest extent due to the fact that it is under constant observation by its neighbors, who may cut C off if they sense selfish participation of C.

5.3.3 High Mobility Scenarios

Mandating PLNs is useful in scenarios involving highly dynamic nodes. Consider a scenario where a mobile device in a fast moving vehicle sends a RREQ packet. If every neighbor simply floods the RREQ onwards it may result in substantial wastage of bandwidth as the RREQ source is very likely to have moved away from the location from which the RREQ originated by the time the response comes back. If nodes enforce a PLN (and induct nodes in their PLN only after a few exchanges) only nodes moving at roughly the same speed (or relatively stationary to each other) will form an exclusive network. Thus a set of northbound vehicles in Interstate I-95 may form an ad hoc subnet that excludes all southbound vehicles (and vice-versa).

5.3.4 Dense Deployments

In dense deployments of wireless devices, and in scenarios where physical layer jamming is an issue, dynamic spread spectrum strategies will be necessary. The shared keys needed for CDMA or frequency hopping can also be derived from the one-hop group secret.

Furthermore, in a scenario where two nodes A and C are situated very close to each other and have identical views of the network, the nodes gain nothing by adding each other to their respective PLNs (unless A and C are end points in an interaction). In a region where a node A has 100 nodes within range, A may decide to include only 10 of them in its PLN as 10 neighbors may be sufficient to provide A with connectivity to all other nodes.

CHAPTER 6

AUTHENTICATION OF ROUTING DATA

Secure extensions of routing protocols add overhead in the form of cryptographic authentications which make use of security associations (SA) between nodes. SAs are in turn facilitated by key distribution schemes (KDS).

While cryptographic security associations could go a long way in enhancing the ability of ad hoc networks to resist attacks, they are not a panacea for every possible attack. For instance, very little can be done about nodes which intentionally introduce random errors in packets that are transmitted. Furthermore, improper use of cryptographic authentication techniques could actually foster denial of service (DOS) attacks. This could be especially serious when nodes employ computationally intensive cryptographic techniques (like asymmetric cryptography). Attackers could just send random bits and use up the resources of neighboring nodes by forcing nodes to perform expensive computations to check the validity of the message sent. Furthermore, that some data that has been cryptographically validated does not imply that data itself is valid [67].

6.1 Network Layer and Application Layer Authentication

In general, cryptographic authentication in MANETs will employ two sets of secrets. The first controlled by the network operators, used for authenticating the network layer

packets (like routing tables, RREQs, RREPs etc). Such secrets should ideally be protected even from the users - only the devices should be privy to the secret - to ensure that malicious users cannot easily send cryptographically well authenticated , but false routing information. The second set of secrets are for authentication of the *application layer* or communications between end points - facilitated by the secrets end users shares with each other.

Ideally, the end users should trust the network operators only to extent required. For this reasons, the two communicating end-points may use out-of-band mechanisms for establishing shared secrets for purposes of mutual authentication. There is no reason for the end points to rely on the network operators for privacy of their interactions. However, for authentication of other nodes who (as far as end points are concerned) are part of the network (as they perform network layer functions like routing), the end points have no options but to rely on the network layer secrets. Furthermore, it should be possible to establish network layer security associations in ad hoc manner, as nodes will in general have no priori knowledge of other nodes with whom they may have to interact with for purposes of routing packets in the subnet.

Authentication can be performed using symmetric cryptographic schemes like Hashed Message Authentication Codes (HMAC) or using asymmetric cryptographic schemes like Digital Signatures (DS). The specific nature of the authentication inserted by any node will determine who can verify the introduced authentication and when it can be verified. If the authentication introduced by A is a hashed message authentication code (HMAC) using symmetric cryptographic primitives, based on some secret K, only entities that share

the secret key K can verify the authentication. For example, if a pair of nodes share a (pairwise) secret, then authentication inserted by A can be based on the secret K_{AD} it shares with the destination D.

In Ariadne, which employs a time sensitive authentication strategy relying on one-way hash chains, only the source of the RREQ at the end of the reverse path can verify the HMACs inserted by intermediate nodes. Some of the secure routing protocols employ authentication just between source and destination nodes like [27]. In some cases, authentication is used only between the neighboring nodes.

6.2 Broadcast Authentication

Cryptographic authentication can be classified into one-to-one and one-to-many schemes. When a one-to-one technique is used the source of the message appends a HMAC (based on a mutually shared secret) for each possible verifier. This is inefficient in scenarios where the information may need to be verified by multiple nodes. Furthermore in many scenarios the source may not know the identities of the potential verifiers a priori. Broadcast authentication schemes which permit any node the verify the appended authentication is thus a very useful strategy for ad hoc networks.

Broadcast (source) authentication [60, 61] a one-to-many security association, is very useful for securing ad hoc routing protocols. If any information transmitted during route request propagation has to be verified by many unknown verifiers, then broadcast authentication is the only choice to achieve this goal.

As an illustrative example of how broadcast authentication (BA) could be used to ameliorate the effects of attacks on routing protocols, we shall consider a specific scenario where a network employs a distance vector based routing scheme. While the principles are equally applicable to source routing based approaches or scenarios where link state information is flooded through out the network, the reasons for choosing a table based approach for illustration is due to the fact that protecting table based routing protocols is more challenging.

With a distance vector routing approach, the table advertised by a node (say node A) will consist of many rows, each row corresponding to a destination for which A has knowledge of the "next hop." An entry for a node H in the table of A is typically of the form $[H \parallel S_h \parallel N_h \parallel S]$, where H is the destination ID, S_h is the sequence number chosen by H (which provides a notion of "freshness" of the routing information), N_h is the number of hops from A to H and S (a one hop neighbor of A) is the next node on the path from A to H.

Node A broadcasts its table consisting of many such rows, to all its neighbors. Following the broadcast of A, a neighbor B broadcasts its routing information to all its neighbors. The broadcast by node B, may contain some information originating from the previous broadcast of A, and possibly information gained from broadcasts by other neighbors of B like C, Y.

There are two fundamental approaches for authenticating the contents of the broadcast by any node. 1) authenticating the entire broadcast or the table with single signature 2) authenticating each row with individual signatures. If the table advertised by node A

is authenticated using single signatures, it is easy to see that the information provided by node B (regarding route to H) cannot be verified for consistency (with information provided by A regarding path to H), by nodes C and Y. Thus either

- Each row has to be authenticated individually-in which case each row in a broadcast by any node, for a destination which is h hops away would have appended h signatures or
- All broadcasts should be flooded (nodes C and Y will receive the tables sent by A, B, S with a single appended signature)

It is easy to see now that link state approaches to routing protocols (in which the broadcast of each node is flooded throughout the network in any case) are thus easier to secure (or involves less overheads) than table based approaches. More specifically, the overheads for securing link state approaches is just a single signature in every link state packet. Securing table based routing however calls for substantially more overheads.

Both approaches - flooding all broadcasts (or using a pure link state approach for ad hoc networks) or appending a signature for every entry at every hop - may be impractical. However, various security complexity trade-offs are possible. A more practical (and a commonly suggested) approach is then to carry over authentication only for a small number of hops.

6.3 Broadcast Authentication: Possible Approaches

If efficient broadcast authentication techniques are available, it is thus possible to improve the security of ad hoc routing protocols. However, in scenarios where computational/bandwidth resources are severely constrained, as we shall see in the following sections, broadcast authentication can be expensive.

6.3.1 Digital Signatures

If digital signature of a A can be verified for authenticity by any node with access to certified public keys of A. Thus each node should append its certificate along with the message it signs, leading to large bandwidth requirement for each signature. The use of digital signatures also increases susceptibility to DoS attacks due to their high computational overhead.

Certificate -less public key methods like identity based encryption schemes (IBE) [62], [63] are seen as being more suitable for ad hoc networks. For IBE schemes, the identity of a node also doubles as the public key. However, such approaches are also susceptible simple DoS attacks due to the high computational overhead. Also, unlike public key schemes like RSA where it may be possible to reduce signature verification complexity by choosing small public exponents (and thereby reduce susceptibility to DOS attacks), such strategies cannot be used for IBE schemes.

6.3.2 One-Way Hash chains

Methods based on one-way hash chains have been investigated in [7], [25]. The source chooses a random K_0 and create a one way hash chains $\{K_0, K_1, ..K_N\}$ where $K_i = H(K_i - 1)$ and uses the value from the hash chain (say k_j) to calculate HMAC for a message to be authenticated. Later the preimage of the value K_j viz $k_j - 1$ is made public. The verifiers are thus assured that the source that transmitted the first message in the first place was the same as the one that released the pre-image. Verification calls for loose time

synchronization [25] between nodes. Furthermore, techniques based on hash chains need to be boot-strapped from a pre-authenticated value.

The process of bootstrapping hash chain based methods calls for some mechanisms for dissemination of commitment keys (the last value in a hash chain K_N) of all nodes. Furthermore, the implication of the security conditions that consists that preimage could not have been released by the source yet is that the messages "signed" in this fashion cannot be reused. Assume that A receives a message M from B (along with a TESLA HMAC). At a time t, when the security condition was satisfied, assume that A receives the preimage used for computing the HMAC. Now while A is convinced of the authenticity of the message M, A cannot convince any other node of the authenticity of M to other nodes.

The main advantage of broadcast authentication using the hash chains is that bandwidth needed for the appended HMAC is low compared to the digital signatures. For this reasons, BA techniques such as TESLA could be well suited for scenarios where authenticating broadcasts originated by a single source (As disseminating commitment keys for a single source may not impose substantial overheads). Another scenario where broadcast authentication using preimages like TESLA can be useful in authenticating the neighborhoods.

6.4 Need for Carrying Over Authentication

If digital signatures are used for authentication, the authentication introduced by node A for instance, can be verified by all nodes downstream of A. For example, in a path (A, B, C, E) from S to D, if a malicious node C modifies any of the mutable fields in-

troduced by B or A, nodes downstream of C can verify that such modifications are not consistent with the signatures of node B or A. However, mandating that every node insert a signature (and perhaps a certificate, if certificates cannot be distributed off-line) before forwarding an RREQ implies very large bandwidth overheads for the RREQ, in additional to the computational overheads imposed by requiring every node to check the signatures of all upstream nodes.

One reasonable trade-off is to carry over the appended authentication only for two hops. For instance, the authentication introduced by A could be verified by B or C and stripped off by C. Similarly, while C forwards the authentication inserted by B onwards, this is stripped by downstream neighbors of C like E. In such scenario where authentication is carried over only for two hops, note that colluding nodes could perpetuate misrepresentations. For example, node B could make some illegal changes to the RREQ sent by A, which will be ignored by C (as C colludes with B).

6.4.1 Per Hop Hashing and Carrying Over Authentication

While verification of appended authentication by downstream nodes can prevent nodes from illegally inserting fictitious nodes in the path or modifying the mutable fields appended by nodes upstream, a simple attack for an attacker is just to remove an immediate upstream node (or a set of upstream neighbors) from the path. As was explained in Chapter 3 the per-hop hashing strategy addresses this issue.

One of the unfortunate side effects of the per-hop hashing techniques is that even renders carrying over authentication impossible. For instance in a path $(A, B, C, ...,)$ between

S and D, node C can no longer verify the authentication appended by A. Specifically, the modifications introduced by A include a quantity β_1 which only one hop neighbors of A should be privy to. Thus if $A's$ authentication includes β_1, C cannot verify the appended authentication. On the other hand, if the authentication appended by A does not include β_1 , the intermediate node B can modify β_1. Thus only neighbors of A, and the destination (which can calculate β_1 as it has access to β_0) can verify the authentication appended by A.

6.5 Issues in Two Hop Authentication

The use of per hop hashing technique is thus not conducive to early detection of RREQ failures. In order to facilitate identification of inconsistent RREQs by intermediate nodes, we need some strategy that does not rely on per hop hashing, but it is still able to prevent insertion attacks.

To see how this can be done, consider the scenarios where a RREQ travels along a path $(A, B, C, E...)$ and node C removes B from the path and announces a path (A, C). What we desire now is for downstream neighbors of C (receiving such RREQ from C) to recognize that A cannot possibly neighbor of C. If this is possible, the bad RREQ will be dropped as desired.

More specifically, for a RREQ received through a path $...B \rightarrow C \rightarrow E$, node E should be able to verify

1. the authentication appended by B and C
2. that C is a neighbor of E and
3. that B is a neighbor of C (to prevent node deletion attacks)

One way of realizing the above requirements is to ensure that every node has complete knowledge of their two hop topology. However a node cannot merely afford to trust their one hop neighbors to provide them with the list of their neighbors, as a malicious C could easily claim that A is also a neighbor. Thus node require "authenticated knowledge "of the two hop topology. This can be achieved if the "neighbor list" information supplied by all neighbors of a node (from which two hop nodes can be detected), also include the authentication appended by every node in the list. Obviously, this is an expensive proposition especially in scenarios involving highly dynamic nodes. We will see later how we realize an efficient strategy for this requirements.

6.5.1 Cost of Broadcast Authentication Schemes

Digital signatures or one way hash chains can be used for broadcast authentication. The problem with the digital signatures is that it requires high bandwidth and computational resources which may not be acceptable for ad hoc networks. In schemes like TESLA [25] the HMAC is based on a preimage which is guaranteed to have not been made public at the instant of time a verifier receives the message. Verification of the security condition calls for time synchronization between the nodes [68].

Furthermore, hash chain based BA schemes need to be boot-strapped from a pre authenticated value. Thus the problem of disseminating commitment keys can be substantially harder problem than disseminating signed public keys. The problem of distributing commitment key among neighbors is more simpler than the distributing over many hops away nodes. Distribution can be done using alternate broadcast techniques such as digi-

tal signatures for bootstrapping purposes (authenticating the commitment key). In other words, complexity involved with digital signatures can be amortized over many signatures which use the preimages of the signed commitment value for authentication of neighbors.

Unfortunately, unlike public key schemes where the public keys are valid for a long durations, for hash chain schemes commitment keys are valid only for much shorter durations depending on the number of pre images that have been calculated beforehand, and the duration of each disclosure interval. Thus the commitment keys have to be re-issued frequently.

In Ariadne the authors mention one possible approach for disseminating commitment keys. Nodes establish a path with an in network KDC and receive encrypted commitment values of all other nodes in the subnet using a secret each node shares with KDC. However this procedure can be cumbersome for situations where nodes can join or leave the network at any time and network life times are significantly longer than the duration of hash chains. Furthermore, in networks where there is little or no involvement of a centralized entity during operation of the network such approaches are not possible.

One possible approach is for each node node to be provided with public/private key pairs in addition to hash chains to permit each node to reissue its commitment values periodically (and authenticating the commitment keys with the digital signatures). However, the in network bandwidth overheads for flooding commitment keys can be unacceptable. Such approaches can also result in increased susceptibility to denial of service attacks (as nodes could just aloof "new commitment keys" to use up computational and bandwidth resources).

CHAPTER 7

EFFICIENT ONE-HOP AND TWO-HOP AUTHENTICATION

We saw in Chapter 5 that enforcing a private logical neighborhood can prevent many risk-free attacks on routing protocols. However mandating a PLN is not sufficient to eliminate all risk-free attacks. To see this, consider a scenario where two nodes S and D are separated by a path A, B, C, E. Let us assume that S invokes a RREQ for D and that the malicious node C receives an RREQ indicating path (A, B) with HMACs $\{M_A, M_B\}$ appended by A and B. Now node C illegally modifies the RREQ and relays an RREQ indicating a path (R_1, R_2 and C) with random HMACs M_{R1}, M_{R2} and M_C. However the RREQ is authenticated in a proper manner to its neighbors using the PLN secret K_C.

Note that even neighbors of C, acting in promiscuous mode cannot detect malicious intentions of C. From the perspective of B, it is indeed likely that C has a neighbor R_2 that B is unaware of. Obviously, if nodes are aware of their two-hop neighbors, such attacks can be avoided.

7.1 Issues in Managing Two Hop Neighborhoods

Similar to maintaining PLN, it is also possible to maintain secure two hop topology to dissuade neighbors (in the PLN) from engaging in malicious activities. There are two ways to maintain secure two hop topology -

For example, if pairwise secrets are employed, then a message should be authenticated with all secrets that a node shares with its two hop neighbors. If there are n one hop neighbors (on an average) the number of two-hop neighbors is n^2 (on an average). The overheads for authenticating messages individually to n^2 verifiers can be unacceptable for dense networks.

Another approach is to maintain a common group secret for all the two hop neighbors. All the messages intended for these nodes will be authenticated using the group secret. The primary overhead here is for *establishing* and *maintaining* the group secret as group secrets have to be provided to new nodes as they enter the two-hop neighborhood. More importantly, when a two-hop neighbor leaves a new group secrets may have to be provided to all other (continuing) two-hop nodes. In volatile networks where node mobility is high, the overheads for constantly modifying group secrets can be high. Nevertheless current schemes in the literature that employ two-hop group secrets employ this approach regardless of the cost associated with it [59].

In our work, we introduce a more efficient technique that employs two hop topology authentication which some a compelling advantage - nodes do *not* need to know the list of two-hop neighbors. In cases where nodes frequently move out of range, nodes just need to maintain the one-hop PLN and not worry about any changes in the two hop neighborhood. The new scheme is based on broadcast encryption (BE).

7.2 Broadcast Encryption

Broadcast encryption (BE) enables a shared secret between $g = G - r$ *privileged* nodes, among a set G nodes: or a set of r *revoked* nodes are denied access to the secret [78].

Typically a set of k secrets are distributed to each of the N nodes before the system is deployed. The source of the broadcast then [78] [71]

1. chooses a broadcast secret K_b (intended for the set of g nodes),
2. encrypts K_b using n keys $K_{e1} \cdots K_{en}$, and
3. transmits n values $E_{K_{ei}}(K_b), 1 \leq i \leq n$.

The keys $K_{e1} \cdots K_{en}$ are chosen in such a way that none of the r nodes can (using their preloaded secrets) can compute *any* of the keys $K_{e1} \cdots K_{en}$, while the remaining $G - r$ nodes will be able to compute *at least one* of the secrets $K_{e1} \cdots K_{en}$, and thereby gain access to the secret K_b.

In most BE schemes in the literature the source of the broadcast is assumed to be the key distribution center. While most schemes can be extended to support broadcast by all nodes - or multi-source BE (MSBE), where any of the G nodes will be able to convey a group secret to $g < G$ nodes, such MSBE schemes typically require asymmetric cryptographic techniques. However there are two exceptions - MSBE schemes which require only symmetric primitives.

For our purposes we are only interested in MSBE schemes since two-hop authentication is used by every node to convey a group secret to some nodes *except* their one-hop neighbors.

7.2.1 Fiat-Noar 1-Resilient Scheme

Fiat and Noar, [64] who were the first to provide a formal definition of BE, proposed various schemes for BE in [65]. In our work, we are interested in the simple and elegant "1 resilient scheme" based on secure one way functions. In this 1-resilient scheme consisting of TA and $N = 2^m$ nodes, the TA chooses a secret k (say b bits) and a secure one way functions $f() : \{0, 1^b\} \rightarrow \{0, 1\}^{2b}$

In other words $f(K) = K_0||K_1$ where K_0 and K_1 are b bit quantities. The values K_0 and K_1 are now the leaves of a binary tree with depth 1. This process is repeated to increase the depth of the tree (K_0 and K_1 yield K_{00}, K_{01},and K_{10}, K_{11} respectively). and so on till $N = 2^m$ leaves of the tree are derived. We shall just label the N leaves as $L_0...L_{N1}(L_0 = K_{00..0}$ and $L_{N-1} = K_{11...1})$

Each of the N nodes with identities $I_0....I_{N-1}$ are provided with $N - 1$ leaf values. Node I_j is provided with all leaves except L_j. Actually, the nodes need to be provided only with $m = \log N$ values to enable them to calculate $N - 1$ leaf values. For example, node I_0 just needs $K_1, K_{01}, K_{001}, K_{00...01}$ to calculate all leaf values except L_0. In other words, each node receives one value from each level of the tree. Node I_3 for instance would receive secrets K_{010}, K_{00} and K_1

In order to establish a secret K_b (which for simplicity we assume is b bits long) not privy to nodes $I_{R1}...I_{Rr}$ a source node (say I_s) could just broadcast $[(I_{R1}..I_{Rr})||K_B]$, where $K_B = K_b \bigoplus L_{R1} \bigoplus L_{R2} \bigoplus ..... \bigoplus L_{Rr}$. which cannot be decrypted by an of the "r" revoked nodes. The header $(I_{R1}..I_{Rr})$ is required for each node to calculate the corresponding leaf secrets and obtain

$K_B = K_b \bigoplus L_{R1} \bigoplus L_{R2} \bigoplus ..... \bigoplus L_{Rr}$. which node of the revoked nodes can, The scheme is 1 resilient as any two nodes can collude to determine all Nl leaf values.

7.2.2 HARPS BE

HARPS (hashed random preloaded subsets), a probabilistic key distribution scheme, was introduced by Ramkumar et. al [66]. A key distribution center (KDC) chooses a set of P secrets $K_1 ... K_P$, and each node is issued a hashed subset consisting of $k = \epsilon p$ keys, where $\epsilon \leq 1$ is the probability with which a key corresponding to some index $1 \leq i \leq p$ is assigned to some node.

A public pseudo-random function (PRF) $f()$ determines if a key corresponding to some index $1 \leq i \leq P$ is assigned to a node, and the hash depth (an integer between 1 and L) of such a key. The hash depth is the number of times a key is hashed repeatedly. Let ${}^{j}K_i = h^j(K_i)$ represent the result of repeated hashing K_{ij} times using the hash function $h()$.

If $\{A_1, A_2, ...A_{k_A}\}$ are the indices assigned to a node A and $\{a_1, a_2, ...a_{k_A}\}$ are the respective hash depths, the set of secrets assigned to a node A are $\left\{{}^{a_1}K_{A_1}, {}^{a_2}K_{A_2}, ... {}^{a_{k_A}}K_{A_{k_A}}\right\}$.

For conveying a group secret K_b to all nodes except the r revoked nodes, the sender employs a subset of all secrets not covered by the union of the r revoked nodes. Let $\mathbb{R}$ the complete set of secrets. Let $\mathbb{S}_R$ be the secrets covered by union of r nodes. All secrets in $\mathbb{R} \backslash \mathbb{S}_R$, which are not privy to any of the revoked nodes can be used to convey the group secret. More specifically, as the source will have access to only a subset of the keys $\mathbb{R} \backslash \mathbb{S}_R$, such keys can be used for distributing the group secret.

In order to establish K_b not privy to r nodes $I_{R1}....I_{Rr}$ a source node could just broadcast $[(I_{R1}....I_{Rr})\,||K_1...K_n]$ where $I_{R1}....I_{Rr}$ represents the IDs of the revoked node and $K_1...K_n$ encryption of the secrets from $\mathbb{R}\backslash\mathbb{S}_R$.

As an example, for HARPS with $(P = 160, k = \epsilon, P = 128, L = 64)$ for a two hop neighborhood of about 100 nodes and one hop neighborhood of 10 nodes), the number of encryptions n necessary for disseminating the broadcasts secret to all two hop neighbors (while protecting the secret from all one hop neighbors) is roughly 14. While HARPS BE is less efficient than 1 Resilient Fiat Noar schemes, the broadcast secrets cannot be obtained even by all revoked nodes pooling the secrets together [69]

7.3 Broadcast Encryption vs Broadcast Authentication

In scenarios where the signatures are carried over two hops, BA serves two purposes - authentication by neighbors, and authentication by two hop neighbors. We saw that BA using hash chains can be used efficiently if it is used only authentication by immediate neighbors. Thus what we desire is a mechanisms for authentication by two hop neighbors. If there exists efficient means for every node to establish a secret with all its two hop neighbors, while at the same time protecting the secret from all one hop neighbors, we can ensure that neighbors cannot modify information that they relay. This is precisely the requirement that MSBE schemes allow us to do.

Using MSBE, it is possible for a node A to establish a shared secret with all its two hop neighbors while protecting the secret from its one hop neighbors. More specifically node A can broadcast a message to all its one hop neighbors, which is then relayed by one-hop

neighbors to their hop neighbors. The message sent by A will contain a secret which none of its one hop neighbors can access (or each node revokes all its one hop neighbors).

The significant reduction in complexity due to the use of BE instead of BA is a result of the following reasons

- BE signatures are computationally far less expensive (both for signing and verification- as its based on a single shared key)
- BE signatures can be much smaller even one or two bytes for each signature is sufficient. BA using digital signatures could be a few hundred bits long.

The second reason perhaps needs further classification. While the key (shared secret) used for computing HMACs (BE signature can be long) and the HMAC evaluated can also be long, it is sufficient to sent few LSBs of the HMAC of the signature. After all, for an attacker who does not have access to BE secrets the only option is to guess the HMAC. Even with one byte HMAC the attacker has a low probability (1/256) of guessing the HMAC. More specifically appending 8 bits of a secure HMAC is not security equivalent to using HMACs based on 8 bit keys.

7.4 Efficient Two Hop Authentication

Every node maintains a PLN. In addition every node constructs a broadcast encryption message which explicitly revokes all nodes in its PLN. Thus a node A with neighbors B, S in its PLN will construct a broadcast encryption message $\mathbb{B}_A$ that revokes nodes B, S (or $\mathbb{N}_A = \{B, S\}$) and contains encrypted version of a secret T_A protected from the nodes in the PLN of A. In other words

Table 7.1

SRD Protocol Abbreviations

$\mathbb{N}_A$	set of one hop neighbors of A in the PLN of A
K_A	secret provided by node A to all its one-hop neighbors (members of the set $\mathbb{N}_A$)
T_A	secret chosen by A that is explicitly *protected* from all one-hop neighbors (all nodes in A's PLN)
$rreq$	non mutable fields of an RREQ message
$rrep$	non mutable fields of an RREP message

1. K_A is a secret provided to all nodes in the RDN of A
2. T_A is a secret protected from all nodes in the RDN of A

The node A provides message to all nodes in its PLN. Thus every node will store as many BE messages as the number of nodes in its PLN. A will have access to PLN secrets K_B, K_S, of its neighbors B, S and the broadcast encryption messages $\mathbb{B}_B$, $\mathbb{B}_S$

Any node forwarding an RREQ include a HMAC based on the broadcast secret. For example, in a scenario where an RREQ reaches D through the path $S \rightarrow A \rightarrow B \rightarrow C \rightarrow E \rightarrow D$, D can verify the authentication appended by C. To provide D with access to T_C, along with the RREQ, E also forwards the BE message $\mathbb{B}_C$ of its predecessor C.

7.4.1 No Explicit Knowledge of 2-Hop Topology

The fact that node E is explicitly revoked in the BE message of C indicates that E is a neighbor of C. For a malicious C which tries to delete node B from the path, C has to produce a BE message from A which revokes C (to convince downstream nodes that A is neighbor of C to advertise the path $A \rightarrow C$). C cannot forge a BE message from A which

include C as a revoked node. Every secret used in such a forged BE message corresponds to secrets C does not possess.

Note that A has no authenticated knowledge of its two hop topology. Even though A has access to B's BE message which indicates that C is a neighbor, this claim is not verified by A. This claim will be verified only if A has the opportunity to forward an RREQ that reaches A through the path $C \rightarrow B \rightarrow A$. In other words two hop secrets are established even while two hop neighbors are identified only when necessary.

Also note that it is *not* assumed that nodes will advertise their PLN correctly in their BE messages. For example B could indicate in its message $\mathbb{B}_B$ a fictitious neighbor U. However, any packet that B claims to have passed through U *should* be supported with a BE message $\mathbb{B}_U$ which revokes B and is authenticated using a secret that B *cannot* have access to. If B cannot produce such a message, B cannot forward any RREQ indicating U as its predecessor.

7.4.2 Overheads for Maintaining Group Secret

It may appear at first sight that a node C which is two hops away from a node A (and had recently gained access to node A's BE secret T_A) can, by moving within one-hop of A, defeat the two-hop authentication process. However, note that when C enters the PLN of A, node A will change both the PLN secret K_A and the two-hop secret T_A. Thus the old T_A is of no use for C. Also note that a node C with the two-hop secret may also become a 3-hop node. However this has no effect on the security of the two-hop authentication process. A node which verifies two-hop authentication based on a BE secret

will also ensure that the immediate predecessor is explicitly revoked in the BE message that conveys the BE secret.

All that any node needs to proactively keep track of is its one-hop neighbors. Changes in two-hop topology for instance 1) a two-hop node vanishing (powering off) or 2) two-hop node becoming a 3-hop node or 3) a two-hop node moving within one-hop has *no* effect on the security of the two-hop authentication process.

The overheads required for two-hop authentication using BE is substantially smaller than the overheads required for maintaining consistent two-hop group secrets. For the latter case, for an average neighborhood size of r which changes (on an average) every t seconds, maintaining a consistent two hop group secret requires bandwidth overheads at a rate $\mathbb{O}(r^2t^2)$ as there will be $\mathbb{O}(r^2)$ nodes at a 2-hop distance and the two-hop topology changes at a rate proportional to $\frac{1}{t^2}$. Furthermore, consistent maintenance of two-hop secrets are required even during "quiet" periods where a node may not relay any RREQ.

On the other hand, when BE is used for two-hop authentication, message exchanges between neighboring nodes occur only when the PLN changes, calling for a rate of $\mathbb{O}(rt)$ instead of $\mathbb{O}(r^2t^2)$. The relay of BE messages to two hop neighbors occurs only if 1) the source of the BE message forwards an RREQ and 2) if the PLN of the source of the BE message has changed. Thus a node C does *not* have to forward the BE message from its neighbor B every time it forwards an RREQ from B. The BE message of B changes only of there is a change in the PLN of B. Node C will have to forward the BE message from B only when it forwards an RREQ with B as the predecessor, *and* if there is a change in either the PLN of B or the PLN of C. More specifically, a change in the PLN of B

invalidates the old BE message from B. A change in the PLN of C may imply that new nodes may be present in the PLN who have not received B's BE message the last time it was forwarded by C.

In a scenario where each node has 5 neighbors on an average and say 25 nodes within two hop radius a BE message will need to include 5 to 10 encryptions of the broadcast secret. Thus a BE message will require bandwidth overheads comparable to that of digital signatures, but very low computational overheads.

7.5 A Secure Route Discovery Protocol for DSR

The secure route discovery (SRD) protocol outlined in this section employs MSBE for two-hop authentication. The protocol assumes the presence of an *off-line* KDC who has distributed encryption and decryption secrets for the MSBE scheme [4]. Further, the SRD protocol also assumes the existence of a suitable KDS for pairwise authentication of nodes. Every node maintains a PLN by providing a group secret to every node in the PLN (by encrypting the group secret individually using pair-wise secrets shared with neighbors).

One of the primary motivations for SRD is to ensure that malicious modifications to RREQ are detected early so that defective RREQs can be dropped as soon as possible. We shall represent the one-hop PLN secret of a node A (provided to all nodes in the set $\mathbb{N}_A$, the PLN of A) by K_A, and the broadcast secret (which is *protected* from all nodes in the set $\mathbb{N}_A$) by T_A.

7.5.1 RREQ Propagation

Let us now consider a scenario where a source S desires to find a path to a destination D. The source creates a RREQ packet with immutable fields

$$rreq = [S \parallel D \parallel seq \parallel h_c]$$

where seq is a sequence number and h_c is the maximum hop count. The node S now broadcasts an RREQ packet

$$\begin{aligned} RREQ_0 &= [S \parallel K_S([rreq \parallel M_0 \parallel h_0])] \\ h_0 &= h(rreq, K_{SD}) \\ M_0 &= h(rreq, h_1, T_S), \text{ where } h_1 = h(h_0) \end{aligned}$$

where K_{SD} is a secret shared between S and D. In other words, h_0 is a HMAC meant for verification by the destination, and M_0 is HMAC for verification by two-hop nodes. Note that all fields of the RREQ packet are encrypted by the one-hop secret of S.

A node A one hop from S decrypts the RREQ packet and broadcasts

$$\begin{aligned} RREQ_1 &= [A \parallel K_A([rreq \parallel (A) \parallel M_0 \parallel M_1 \parallel h_1])] \\ M_1 &= h(rreq, (A), h_2, T_A), \text{ where } h_2 = h(h_1) \end{aligned}$$

A node B one-hop away from A and two-hops away from S decrypts the RREQ, verifies that h_1 sent by A is consistent with the HMAC M_0 appended by S. Having verified M_0, node B strips off M_0 and appends an HMAC M_2 for verification by nodes two hops downstream of B. Thus B airs

$$\begin{aligned} RREQ_2 &= [B \parallel K_B([rreq \parallel (A, B) \parallel M_1 \parallel M_2 \parallel h_2])] \\ M_2 &= h(rreq, (A, B), h_3, T_B), \text{ where } h_3 = h(h_2) \end{aligned}$$

7.5.2 RREP

When the node D receives the RREQ packet with a per-hop hash value h_i and i nodes in the path, D can verify that h_i is consistent with h_0 (which D can evaluate as it is based on a secret K_{SD} shared between the source and destination). Let us assume that the RREP reached the destination through a path (A, B, C, E, D)

The RREP raised by D takes the form

$$rrep = [S \parallel D \parallel seq \parallel (A, B, \ldots, C, E)].$$

Now D unicasts the RREP packet

$$\begin{aligned} RREP_0 &= [D \parallel K_D([rrep \parallel q_0 \parallel M_{DC}]) \\ M_{DC} &= h(rrep, q_1, K_{DC}), \text{ where } q_1 = h(q_0) \end{aligned}$$

Note that the RREP packet has two HMACs - q_0 for verification by the source at the end of the RREP, and M_{DC} for verification by a node C two hops away in the RREP path.

The RREP packets relayed by nodes E and C take the form

$$\begin{aligned} RREP_1 &= [E \parallel K_E([rrep \parallel q_1 \parallel M_{DC} \parallel M_{EB}]) \\ M_{EB} &= h(rrep, q_2, K_{EB}), \text{ where } q_2 = h(q_1) \\ RREP_2 &= [C \parallel K_C([rrep \parallel q_2 \parallel M_{EB} \parallel M_{CA}]) \\ M_{CA} &= h(rrep, q_3, K_{CA}), \text{ where } q_3 = h(q_2) \end{aligned}$$

7.5.3 Simulations

To obtain a more quantitative estimate of the benefits of early detection of inconsistencies in RREQs we have performed extensive simulations to determine the percentage of route discovery attempts (between randomly chosen node pairs) that succeed.

For the simulations we generated many random realizations of 200 nodes in a square with unit edges. The range of each node was assumed to be 0.1 units. Out of the 200 nodes 40 nodes were assigned as malicious. It is assumed that the malicious node will illegally modify the RREQ. In our simulations each node had (on an average) 5 neighbors. Note that each node could receive as many RREQs as the number of its neighbors. We assume that the route discovery attempt between two nodes fail if *every* such RREQ path includes a malicious node.

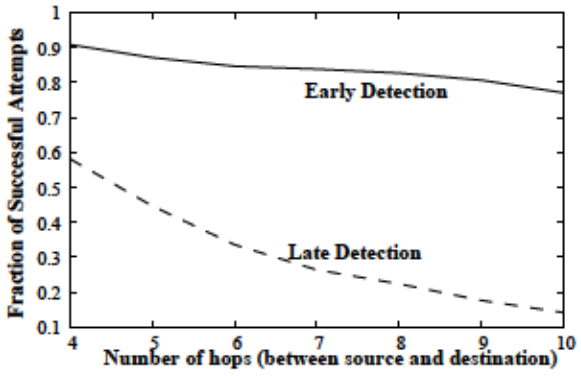

Figure 7.1

Result of early detection of inconsistencies in RREQ packets

We simulated RREQ propagation between every pair of good nodes. The simulation results are shown for two cases ① bad RREQs are detected only by the destination (late

detection) ② bad RREQs are detected within two hops (early detection) and stopped from further propagation. Simulation of RREQ propagation was performed for over 200,000 node pairs, chosen from 5 different random realizations of the network. In each realization 40 nodes were randomly assigned to be malicious. For purposes of comparison between the two cases under different hop counts between the source and the destination, the plots in Figure 7.1 have the hop-count between the chosen source-destination pair (in terms of the number of hops in the shortest path) as the x-axis. The y-axis is the fraction of node-pairs that succeed in the route discovery attempts.

The dashed line represents late detection and the solid line represents early detection. As seen from the plots, early detection of RREQ inconsistencies can substantially improve the performance of any on demand routing algorithm by preventing preemption of good paths by defective RREQs.

7.6 Secure Route Discovery for AODV

A popular secure version of AODV is the SAODV protocol [24]. We introduce some modifications to SAODV (by employing SRD for route discovery) to overcome many limitations of SAODV. We will refer to the modified protocol as SAODV-2.

7.6.1 Route Discovery

Let us consider a scenario where S originates a route request packet for determining a route to D. The immutable fields of the RREQ, represented by $rreq$ consists of

$$rreq \;=\; [S \parallel D \parallel seq_S \parallel seq_D \parallel h_c \parallel \tau_S \parallel s_{h_c}$$

$$\| \text{SIG}_S \| \text{CERT}_S]$$

where seq_D is the last known sequence number of D by S, τ_S is an absolute time after which RREQs for S will not be honored by intermediate nodes that cache the $rreq$, SIG_S is the digital signature appended by S (covering all quantities to the left) CERT_S the public key certificate of S.

As in SAODV [24], in SAODV-2 the source S chooses a *random* s_0 and computes h_c repeated hashes to arrive at $s_{h_c} = h^{h_c}(s_0)$. The broadcast by S which includes $rreq$ takes the form

$$\begin{aligned} S \rightarrow * \quad &: \quad [S \parallel K_S([rreq \parallel 0 \parallel s_0 \parallel M_0])] \\ s_0 \quad &= \quad h(rreq, K_{SD}), \quad s_1 = h(s_0) \\ M_0 \quad &= \quad h(\{rreq, 1, s_1\}, T_S) \end{aligned}$$

All fields in the message aired by S (except the ID S) are encrypted using the secret K_S so that only neighbors in the logical RDN of S can receive and process the RREQ. The value M_0 is for purposes of verification by two-hop neighbors of S (whose identities may not be known to S).

A neighbor A of S 1) decrypts the RREQ transmitted by A, 2) increases the hop count field to 1, and broadcasts

$$\begin{aligned} A \rightarrow * \quad &: \quad [A \parallel K_A([rreq \parallel 1 \parallel s_1 \parallel (M_0 \parallel S) \parallel M_1])] \\ M_1 \quad &= \quad h(\{rreq, 2, s_2\}, T_A), s_2 = h(s_1) \end{aligned}$$

As mentioned earlier, if A had not relayed the current BE message $\mathbb{B}_S$ (which changes whenever the RDN of S changes) earlier, A also includes the message $\mathbb{B}_S$ along with the RREQ. Such messages will also be encrypted with A's one hop secret K_A.

A node B at the next hop decrypts the broadcast by A (using A's RDN secret K_A). If B has not handled the BE message $\mathbb{B}_S$, it extracts the shared secret T_S and verifies the HMAC M_0 appended by S. Note that in verifying M_0 node B is assured of the 1) integrity of the RREQ and that 2) A sent a valid $s_1 = h(s_0)$. On successful verification B strips M_0 and adds a HMAC M_2 for verification by its two hop downstream neighbors. Thus the broadcast by B takes the form

$$\begin{aligned} B \rightarrow * \quad &: \quad [B \parallel K_B([rreq \parallel 2 \parallel s_2 \parallel (M_1 \parallel A) \parallel M_2])] \\ M_2 \quad &= \quad h(\{rreq, 3, s_3\}, T_B). \end{aligned}$$

Every intermediate node also includes the identity of the previous hop to facilitate the next hop to verify two-hop authentication appended. Intermediate nodes cache 1) RREQs they have forwarded and 2) note down the identities of the two predecessor nodes in the RREQ[1].

When the RREQ reaches the destination indicating a hop count of j and a value s_j, the destination can verify independently that s_j is consistent with the commitment s_{h_c} signed by the source and the hop count j indicated in the RREQ. Let us assume that the RREQ reached the destination through the path $\cdots M, N, W, P$. Note that the destination will

[1]Only one in the case of one-hop neighbors of the source.

be aware of the IDs of the two immediate predecessors (W and P) in the RREQ as the destination will verify the HMAC appended by node W.

7.6.2 Route Response

The destination invokes an RREP

$$\begin{aligned} rrep &= [S \parallel seq_S \parallel D \parallel seq'_D \parallel h'_c \parallel \tau_D \\ &\quad \parallel d_{h'_c} \parallel \text{SIG}_D \parallel \text{CERT}_D] \end{aligned} \tag{7.1}$$

where

1. seq'_D is a fresher sequence number of D;
2. $h'_c = j$ (the hop count indicated in the RREQ);
3. $d_{h'_c} = h^{h'_c}(d_0)$ where d_0 is a randomly chosen value by the destination; and
4. τ_D (for use by intermediate nodes responding to RREQs bound for D in the future).

The destination also appends a HMAC M_{DW} based on the secret K_{DW} it shares with the node W two-hops away and unicasts to its neighbor P

$$RREP_0 = [D \parallel K_D([rrep \parallel 0 \parallel d_0 \parallel M_{DW}])] \tag{7.2}$$

$$M_{DW} = h(\{rrep, 1, d_1\}, K_{DW}) \tag{7.3}$$

The RREPs unicasted by nodes P and W along the reverse paths will now take the form

$$\begin{aligned} RREP_1 &= [P \parallel K_P([rrep \parallel 1 \parallel d_1 \parallel M_{DW} \parallel M_{PN}])] \\ M_{PN} &= h(\{rrep, 2, d_2\}, K_{PN}) \end{aligned} \tag{7.4}$$

$$\begin{aligned} RREP_2 &= [P \parallel K_P([rrep \parallel 2 \parallel d_2 \parallel M_{DW} \parallel M_{WM}])] \\ M_{WM} &= h(\{rrep, 3, d_3\}, K_{WM}) \end{aligned} \tag{7.5}$$

Thus the RREP packets are also encrypted in transit with the one-hop group secret, and authenticated using a HMAC to two-hop nodes. Note that RREPs can be efficiently authenticated in the reverse path as nodes already know the identities of the nodes two hops away.

7.6.3 RREP by Intermediate Nodes

Any intermediate node, say C, receiving an RREQ from node S bound for a destination D can raise an RREP if

1. it finds an RREQ or RREP from D in its cache with time τ_D greater that the current time
2. the last known sequence number seq_D indicated by the RREQ source is lower than the sequence number of D in the cached RREQ.

For example, consider a scenario where a node B which is i hops away from a node D and node B has in its cache a signed $rreq$ or $rrep$ from node D. Node B will also have access to the value d_i (the commitment for which can be found in the $rrep$ / $rreq$ and signed by D). When B receives an RREQ for D from some node V, B can invoke an RREP in which the $rrep$ field will take the form

$$rrep' = [V \parallel seq_V \parallel D \parallel RR_D] \tag{7.6}$$

where RR_D is ***either*** the $rreq$ or $rrep$ from D (in its cache). Once again note that node B will have knowledge of the last two hops in the RREQ path from V to B. Thus if the RREQ reaches B through J and H the RREP by node B takes the form

$$RREP_0 = [B \parallel K_B([rrep' \parallel i \parallel d_i \parallel M_{BH}])] \tag{7.7}$$

$$M_{BH} = h(\{rrep', i+1, d_{i+1}\}, K_{BH}). \tag{7.8}$$

As earlier intermediate nodes between B and V unicast the RREP exactly as for the case of RREP instantiated by the destination (Eqs (7.2) - (7.5)).

7.6.4 Security Analysis

Verifying the integrity of any route obtained from any flooding based route discovery protocol calls for the ability to 1) verify that no node can insert itself in the path (or forward an RREQ) without a valid authentication 2) verify that no value inserted by a node can be removed by a downstream node and 3) ensure that the reverse path also exists (or no one-way links).

Node insertion attacks are prevented by mandating authentication from every participating node. Every node authenticates itself to its one hop neighbors (using the group secret) and two-hop neighbors using the BE secret. Thus unless two nodes collude, nodes cannot be inserted illegally.

Prevention of node deletion attacks between any two end points calls for a shared secret between the end points. In other words, two nodes R and B separated by any number of nodes can recognize node deletion attacks if R and B share a secret. Thus while simply carrying over authentication does not prevent node deletion attacks, the ability to establish a secret with two-hop downstream neighbors provides the assurance that one-hop path between the two nodes cannot be modified.

One-Hop Secret: The use of PLN secret (or one-hop secrets) to encrypt all transmissions by any node, are for 1) preventing the use of one-way links, 2) keeping external

attackers out of the network, and 3) protecting the per-hop hash value in any RREQ or RREP received by a node.

Per-hop Hashing: Irrespective of the whether a node receives a per-hop hash value of some node - 1) in a RREQ or 2) an RREP packet or even 3) an RREQ / RREP packet present *inside* an RREP packet when RREP is invoked by an intermediate node - a node i hops away from some node D will only have access to the value d_{i-1}.

While per-hop hashing is strictly not required (as authentication with two-hop secrets prevents attackers from both extending and shortening paths), it provides an independent confirmation for every intermediate node regarding the number of hops the RREQ or RREP have traversed. In other words, even nodes that have "some how" gained access to the BE secret of a node in the RDN cannot *shorten* the path. In other words, under such an eventuality, SAODV-2 offers the same protection as SAODV (except that SAODV-2 still keeps external attackers out). Another important reason for including this value is that the overhead required for computing / appending the per-hop hash value is trivial.

RREP Authentication: SAODV-2 does *not* employ the two-hop BE secrets for authentication of RREP. The primary advantage of BE is that it makes it possible for a node to convey a secret to some nodes that are two hops away even while the conveyor (source of the BE message) has no *a priori* knowledge of the identities of the nodes that are two hops away. This is indeed the situation during RREQ propagation. In the reverse path (RREP) however nodes *do* have *a priori* knowledge of the next two nodes in the path ahead (this information is gained from the RREQ). Thus there is no need to employ a weak group se-

cret for authentication as a stronger form of authentication can be realized using pairwise secrets.

CHAPTER 8

APALLS- ARIADNE WITH PAIRWISE SECRETS AND LINK LAYER SIGNATURES

The design of secure MANET routing protocols need to consider several constraints and goals. Constraints include minimal infrastructural support for distribution of keys, and low overhead for the security features. Goals include effective deterrents to a wide range of malicious behavior, and defensive measures to reduce the ill-effects of nodes, which, in spite of the deterrents, do not co-operate.

Thus far, secure routing protocols have focused merely on strategies to detect inconsistencies in routing packets with the intention of dropping inconsistent packets. In this context, the advantage of the two protocols outlined in the previous chapter - the SRD protocol [5] for DSR and the SAODV2 protocol [6] for AODV - lies in *early detection* of inconsistencies, and thereby reducing the ill-effects of routing packets carrying incorrect information. In addition, due to the use of a PLN, neighbors will be able to monitor nodes. Repeated misbehavior can result in rejection a node from the PLNs of its neighbors. In addition using the PLN can also address passive attacks involving selective participation, and semi-active attacks by invisible relays.

8.1 Motivation for APALLS

An important question that requires further consideration is "who identifies misbehaving nodes?" For example, consider the route establishment process between two nodes S and T through as path $S \rightarrow A \rightarrow B \rightarrow C \rightarrow E \rightarrow F \rightarrow T$. If an intermediate node B discovers that a neighbor C "misbehaved" during the route establishment process, all that B can do is to remove C from its PLN. C may still be included in the PLN of some of its other neighbors, and continue to take part in routing.

Note that if the end-points S or T are able to detect that C misbehaved, they can send a warning to other nodes to disregard C for purposes of establishing paths between S and T. More importantly, to submit such a warning, the end-points do *not* have to supply conclusive proof. After all, end-points have every right to demand who should or should not handle their packets. In addition, "detection by the source S" and "detection by destination T" are not equivalent. For example, if the destination T detects that C is malicious, T can instruct nodes downstream of C (nodes E, and F) to drop packets relayed by C. However, S cannot prevent a downstream node C from forwarding a packet.

Misbehavior by a node can be because a node is under the control of an attacker, or due to malfunctioning of hardware/software. It is obviously desirable for secure routing protocols to have the ability to obtain non repudiable proof of misbehavior by nodes, leading to revocation of nodes from the network. Thus, while the scope existing secure protocols have stopped at detection of inconsistencies, there is a need to widen their scope to permit identification of attackers, strategies to route around such attackers, and obtaining non repudiable proof of attacks.

8.1.1 Ariadne and APALLS

Unfortunately, such strategies will be highly protocol-specific. In protocols like AODV the end-points do not even know the identities of all nodes in the path. On the other hand, in source routing protocols like DSR, and link-state based approaches [15], the end-points get to know the entire list of nodes in the path. Obviously, strategies for enhancing the scope of the secure version of AODV can be substantially different from strategies suitable for DSR. More specifically, it is only for protocols like DSR and OLSR that more consummate security measures (identification, and routing around misbehaving nodes) are possible. Due to the substantial overhead required, link-state based approaches are not seen as suitable for ad hoc networks. For this purpose, we restrict ourselves to developing a comprehensive secure extension of DSR.

One of the most popular secure extensions of DSR is Ariadne [7]. Ariadne does not prescribe strategies for monitoring, and consequently, does not possess mechanisms to address passive attacks involving selective participation. The cryptographic authentication strategies in Ariadne permit the source or destination to detect inconsistencies in path advertisements (in RREQ or RREP packets) resulting from active attacks. However, Ariadne does not attempt to identify the perpetrator responsible for the inconsistency, leading to several risk-free attacks on Ariadne. An important consideration affecting the choice of cryptographic primitives is the MANET deployment model. Many of the pitfalls of Ariadne stem from the lack of consideration of realistic network models, and the precise role of trusted authorities (TA) in MANET deployments.

Ariadne with Pairwise Authentication and Link Layer Signatures (APALLS) overcomes many shortcomings of Ariadne. APALLS provides a severe deterrent for active attacks: the threat of revocation from the network. APALLS prescribes features to effectively route around nodes suspected of misbehaving, and includes elements to keep an effective watch on neighbors to address passive attacks involving selective participation. Due to careful choice of cryptographic primitives APALLS incorporates all these features without placing unreasonable demands on the capabilities of mobile nodes.

8.2 Role of a Trusted Authority in MANETs

Two basic tools employed to thwart attacks in MANET protocols are i) mandating cryptographic authentication; and ii) monitoring neighbors to estimate their trustworthiness. An important pre-requisite for monitoring is the need to know *who* is being monitored. Thus cryptographic authentication is necessary for the ability to monitor.

Facilitating cryptographic authentication requires a TA who conveys secrets to every node and/or public values associated with every node to every node in the network. Such values provided by the TA are necessary for computing verifiable cryptographic authentication tokens. As packets without verifiable cryptographic authentication will be ignored, the TA confers the eligibility upon nodes to take part in the network (or more specifically, in any ad hoc subnet created by any subset of nodes belonging to the network).

The *network size* N is the total number of nodes afforded the privilege of participating in MANET subnets. Any random subset of the N eligible nodes, say N_s nodes (which may accidentally be in the same geographical region), can come together to form a connected

ad hoc subnet. The TA may also revoke the privileges of some nodes, by periodically disseminating revocation lists. The trusted authority (TA) also specifies a set of rules (the secure routing protocol) to be followed by every node.

For retaining the most compelling advantage of MANETs, viz., their reduced need for infrastructural support, it is desirable that the TA is *off-line*[1]. In other words, subnets disconnected from the TA, or any other form of infrastructural support, should still be able to carry out their tasks.

8.3 Salient Features of APALLS

While many researchers have independently addressed different aspects of secure routing protocols, thus far no comprehensive protocol addressing a wide range of features desired of MANET routing protocols has been proposed. APALLS is a comprehensive protocol which addresses several shortcomings of current protocols. Specifically, some of the issues addressed by APALLS are as follows.

1. While non-repudiable authentication is necessary, it is *not* sufficient for obtaining non-repudiable proof of misbehavior. Investigation of issues in obtaining non repudiable proof of misbehavior (in the context of adhering to a MANET routing protocol) has not received attention thus far. APALLS is the first protocol to address this issue.
2. Protocols that focus on monitoring strategies [27], [72], have predominantly ignored the relevance of cryptographic authentication. Likewise, protocols that focus on cryptographic authentication typically ignore strategies for monitoring. Comprehensive routing protocols should productively employ both strategies.
3. While many key distribution strategies have been proposed for ad hoc networks, they have not considered realistic network models. The choice of schemes in APALLS are driven by realistic models for large scale MANET deployments.

[1] In [37] MANET networks for which infrastructural support is required only for predistribution of keys are referred to as *managed open* networks.

8.3.1 Notations Used

In this chapter $\mathcal{Q}_{(q,S)}$ represents an RREQ from source S, with a sequence number q. The notation $\mathcal{Q}^{X}_{(q,S)}$ represents all RREQ fields flooded by a node X. Thus, for an RREQ $\mathcal{Q}_{(q,S)}$ traversing a path $(A, B, C, \ldots)$ from S to T

$$\begin{aligned}
\mathcal{Q}^{S}_{(q,S)} &= \mathcal{Q}_{(q,S)} = [S, q, T, n_h] \\
\mathcal{Q}^{A}_{(q,S)} &= [\mathcal{Q}^{S}_{(q,S)}, (A)] \\
\mathcal{Q}^{B}_{(q,S)} &= [\mathcal{Q}^{S}_{(q,S)}, (A, B)] = [\mathcal{Q}^{A}_{(q,S)}, (B)] \\
\mathcal{Q}^{C}_{(q,S)} &= [\mathcal{Q}^{S}_{(q,S)}, (A, B, C)] = [\mathcal{Q}^{B}_{(q,S)}, (C)], \\
&\vdots
\end{aligned} \tag{8.1}$$

When the RREQ packet reaches the destination, it contains the fields specified by the source, and the list of all nodes in the path traversed by the RREQ. A route-response (RREP) packet from the destination is relayed along the reverse path $(\ldots, C, B, A)$ to S. The source and destination may in general discover multiple paths as the destination T can receive one RREQ from every neighbor (of T). Some of the other notations used in this Chapter are tabulated in Table 8.1

8.4 Ariadne

In the context of DSR "application layer" cryptographic mechanisms are required for end-points to protect the privacy and integrity of data exchanged between them. "Link layer" mechanisms are required for authentication of neighboring nodes. "Network layer"

Table 8.1

Notations Used in Chapter 8

$A, B, C \cdots$	(upper case alphabets) unique node identities.
$\Sigma_A = \langle M \rangle_A$	digital signature of A for a message M.
$h()$	a cryptographic hash function.
$h(M, K)$	a hashed message authentication code (HMAC) for a message M using a secret K.
$C = K[P]$	encryption of a plain-text P using a key K.
$P = K^{-1}[C]$	decryption of cipher-text C.

mechanisms are required for authentication of intermediate nodes that take part in relaying packets between end-points.

How end-points choose to protect application data should be left to the end-points themselves; ideally, such strategies should *not* not be under the control of the TA. On the other hand, uniform strategies for link layer and network layer authentication (which will need to be followed by all nodes, in all subnets) *should* be promulgated by the TA.

8.4.1 Cryptographic Authentication in Ariadne

In Ariadne the end-points share an application layer secret. The authentication strategies facilitated by the TA include

1. a sequence-number hash chain to limit the number of RREQs that can be sent by any node;
2. TESLA broadcast authentication [25] for authentication of intermediate nodes by an end-point; and
3. a network-wide secret at the link layer to deny access to external nodes.

In the sequence-number hash chain of a node A, $\mathcal{S}_A = \{R_A^0, R_A^1, \ldots, R_A^{n_r}\}$, where $R_A^i = h(R_A^{i-1})$ for $1 \leq i \leq n_r$; the *commitment* to the chain, $R_A^{n_r}$, is made known to every node by the TA.

In the TESLA hash chain $\mathcal{H}_{t_0,\Delta}^A = \{K_A^0, K_A^1, \ldots, K_A^{L-1}, K_L^A\}$ of node A with commitment K_L^A, Δ is a time-interval (for example, 1 second), and t_0 is an absolute value of time (for example, Dec 1, 2009, 0200:00, GMT). A is expected to keep the key K_A^{L-i} in its chain private at least till time $t_i = t_0 + i\Delta$. The parameters associated with A's chain (commitment K_L^A, Δ and t_0) are made known to all nodes by the TA. This TESLA chain can be used by A only for a limited segment of time (between t_0 and $t_0 + (L-1)\Delta$).

This key K_A^{L-i} can be used for authenticating a value M by appending a HMAC $h(M, K_A^{L-i})$, provided the HMAC reaches potential verifiers *before* time t_i. Once K_A^{L-i} is made public (*after* time t_i) the verifier can i) verify the TESLA HMAC; and (if consistent) ii) repeatedly hash K_A^{L-i} (i times) and verify that the result is indeed the commitment K_A^L. The verifier can then conclude that the HMAC was computed by A, as only A could have had access to the value K_A^{L-i} before time t_i.

8.4.2 RREQ and RREP in Ariadne

The steps involved in RREQ propagation from a source S to a destination T through a path $(A, B, C, \ldots)$ are depicted in Eqs (8.3 - 8.7) (Left Column). $\bar{K}_{ST}$ is the application layer secret shared by the end-points. The RREQ fields $\mathcal{Q}_{(q,S)}^S$ specified by the source includes a time t_i before which the destination should receive the RREQ, and a value $R_S^{n_r-q}$ from S's sequence number hash chain. Intermediate nodes and the destination will

process the RREQ only if i) (S, q) is fresh (the node had not previously seen an RREQ from S with a sequence number q or higher); ii) hashing $R_S^{n_r - q}$ q-times yields the commitment $R_S^{n_r}$ of S; and iii) the RREQ is received before time t_i.

Apart from the fields $\mathcal{Q}_{(q,S)}^S$ which are carried forward all the way to the destination, the source S broadcasts a value β_S which is intended only for its neighbors. The value β_S is chosen such that the destination T (which shares the secret $\bar{K}_{ST}$ with the source) can also compute β_S.

A node A downstream of S appends two values before rebroadcasting the RREQ as $\mathcal{Q}_{(q,S)}^A = [\mathcal{Q}_{(q,S)}^S, (A, M_A)]$. M_A is a TESLA HMAC computed using the value K_A^{L-i} from A's TESLA chain which will remain A's secret till time t_i. In addition, a per-hop hash value $\beta_A = h(\beta_S, A)$ is also broadcast by A. Similarly, a node B downstream of A broadcasts $\mathcal{Q}_{(q,S)}^B = [\mathcal{Q}_{(q,S)}^A, (B, M_B)]$ and $\beta_B = h(\beta_A, B)$. Unlike TESLA HMACs which are carried forward all the way to the destination, the per-hop hashes are not (they are intended only for neighbors).

When the destination T receives the RREQ it computes β_S (using $\bar{K}_{ST}$) and verifies that the per-hop hash submitted by the last node in the path is consistent with the list of nodes in the path. A consistent per-hop hash assures the destination that i) the values $\mathcal{Q}_{(q,S)}$ specified by the source were not modified en-route; and ii) no node was deleted from the path. Without the per-hop hash a node C can simply remove the values (B, M_B) in the RREQ. With the per-hop hash, to trick the destination into accepting a path $(A, C, \ldots)$ as valid, C will need access to the per-hop hash β_A (which is privy only to neighbors of A, and C is not one).

The destination will invoke an RREP only if the per-hop hash is consistent. The RREP includes all fields of the RREQ packet and is authenticated to the source using a HMAC based on secret $\bar{K}_{ST}$. After time t_i the destination relays the RREP along the reverse path. Every intermediate node can now release the value from the TESLA chain used for computing the TESLA HMACs during the forward path. At the end of the reverse path the source i) verifies the TESLA HMACs, and that ii) the TESLA keys are consistent with the time t_i and their respective commitments. Mandating authentication prevents illegal node insertions. Thus, node deletion attacks can be detected by the destination at the end of the forward path; node insertion attacks can be detected at the end of the reverse path, by the RREQ source.

8.5 Application Model for APALLS

As an application of MANETs consider a (fictitious) network operator, Tingular, who desires to provide subscribers with a wide variety of services with minimal investment in infrastructure. Tingular subscribers can find other Tingular subscribers within range and form multi-hop MANET subnets. Nodes (Tingular subscribers) in a connected subnet can exchange messages with each other. Apart from communicating with other nodes in the subnet, any node with wide-area connectivity (for example, access to the Internet) can act as a gateway and extend Internet access to all other nodes in the subnet.

The total number of Tingular subscribers at any time t, say $N(t)$, can be of the order of several millions (and may also vary with time t as new subscribers may join, and some may leave the Tingular network). Any small subset of current subscribers who happen to

be in a geographical region like a mall, or an airport, can come-together to form a Tingular subnet (a subnet can have as little as two nodes). Several Tingular subnets may operate simultaneously at different locations, created and torn down in an ad hoc manner.

While some subnets may have one or more nodes with wide-area network access, some may be completely isolated from the rest of the world. Tingular subnets will be able to function even during periods of natural disasters, when other communication infrastructure fail.

The main tasks to be performed by Tingular for *managing* the network are

1. to specify rules (the secure routing protocol) to be followed by every node (a one-time process);
2. induct subscribers into the Tingular network, by providing them with secrets, possibly in exchange for a small subscription fee (performed once for every node inducted into the Tingular network); and
3. (periodically) revoke subscribers who violate rules, or do not renew their subscriptions, and disseminate revocation lists.

Inducting a subscriber can be as simple as providing a SIM card (subscriber identity module) to the subscriber, preloaded with the secrets necessary for the node to communicate with other Tingular subscribers, and thus take part in any Tingular subnet. A "Tingular node" can be any WiFi enabled general-purpose computer (hand-held, laptop or desktop) which can house the SIM card, and can run Tingular software. The "Tingular software" dictates the rules to be followed by Tingular nodes. Revoking nodes can be through periodic dissemination of revocation lists posted in Tingular's website. Due to the modest investment required, Tingular can afford to provide useful services for a low subscription

fee. Furthermore, even if the "Tingular infrastructure" suffers a failure, current subscribers will still be able to employ the network.

8.5.1 Threat Model and Goals

Some Tingular nodes in a subnet may engage in active attacks. Some attacks may result from mere malfunctioning of nodes. Some may be perpetrated by Tingular nodes under the control of attackers. One goal of APALLS is to obtain non repudiable proof of active attacks. Such proofs can be submitted to the off-line TA at a convenient time (as access to the TA may not be available from some ad hoc subnets), leading to revocation of such nodes from the Tingular network.

The revocation process will involve i) submission of proof to the TA, ii) verification of such proofs by the TA, and iii) dissemination of revocation lists by the TA. While the threat of revocation can be an effective deterrent, the process of revocation may not provide immediate relief from attackers in the subnet. For this purpose, APALLS will include explicit features to improve the chance of finding paths free of *suspected* active attackers.

Passive attacks include acts like not forwarding RREQ packets, not accepting RREP or data packets, etc. Selfish subscribers may not desire to take part in the subnet unless they require to communicate with another node. There may also exist other external attackers (who may not be Tingular nodes) performing semi-active attacks. While obtaining non repudiable proof of misbehavior is not possible for passive and semi-active attacks,

APALLS will possess tangible measures for promoting self-less behavior, and mitigating the ill-effects of semi-active attackers.

8.6 Key Distribution for APALLS

In the context of DSR, "application layer" cryptographic mechanisms are required for end-points to protect the privacy and integrity of data exchanged between them. "Link layer" mechanisms are required for authentication of neighboring nodes. "Network layer" mechanisms are required for authentication of intermediate nodes that take part in relaying packets between end-points.

How end-points choose to protect application data should be left to the end-points themselves; ideally, such strategies should *not* not be under the control of the TA. On the other hand, uniform strategies for link layer and network layer authentication (which will need to be followed by all nodes, in all subnets) *should* be promulgated by the TA.

The operator of the Tingular network employs a web-server, with a certified trustworthy module (for example, a cryptographic co-processor) at the back end. The trustworthy module is the TA. Tingular also possesses a facility for securely preloading SIM cards with some secrets provided by the TA. The TA (the trustworthy module)

1. generates an asymmetric key pair (R_{TA}, U_{TA}), and
2. a master secret μ.

Potential subscribers create a Tingular user account. On payment of the subscription fee subscribers receive a unique identity, and a Tingular SIM card with some secrets corresponding to the identity.

The identity A assigned to a subscriber is of the form $A = [Q_A \parallel A']$ where Q_A is a serial number, and A' may be the user name. The node A is also issued[2] an asymmetric key pair (R_A, U_A), and a certificate $C_A = E(R_{TA}, A \parallel U_A \parallel T_e)$, where T_e is time of expiry of C_A. The certificate can be decrypted[3] by any any node with access to the public key U_{TA} as $[A \parallel U_A \parallel T_e] = D(U_{TA}, C_A)$.

The subscriber sequence number is issued in a chronological order. The sequence number for the first subscriber is 1. A node A with (say) $Q_A = 1,000,000$ is the millionth subscriber. The values included in the SIM card provided to A are

1. a secret $K_A = h(\mu, A)$;
2. the private key R_A;
3. the public key of the TA, U_{TA}; and
4. the certificate C_A.

In addition to the values included in the SIM card, A receives $Q_A - 1$ *public* values (which need not be kept a secret) of the form

$$P_{AX} = h(K_A, X) \oplus h(K_X, A), \forall X = [Q_X \parallel X'] | Q_X < Q_A,$$

where any $X = [Q_X \parallel X']$ represents the identity assigned to a subscriber inducted *before* A (or $Q_X < Q_A$). The public values could be downloaded from Tingular web site or mailed by Tingular (in a flash card / optical disc) over the postal network. The millionth subscriber A will receive 999,999 such public values (which can be stored in a flash card in the mobile device).

---

[2]If desired subscribers can also *generate* their own key pairs. Some trade-offs are discussed later in this paper.

[3]The specifics of the functions $E()$ and $D()$ depend on the type of asymmetric primitive employed.

The shared secret K_{AB} between two subscribers $A = Q_A \parallel A'$ and $B = Q_B \parallel B'$ is computed as follows. If $Q_A < Q_B$ then B has access to the public value P_{AB} (and A does not). The secret K_{AB} is computed as

$$K_{AB} = \begin{cases} h(K_A, B) & \text{by } A \\ h(K_B, A) \oplus P_{AB} & \text{by } B \end{cases} \tag{8.2}$$

The scheme described above for facilitating pairwise secrets is based on the modified Leighton-Micali scheme (MLS) [73], which is itself an extension of a scheme in [74]. If the public values (and consequently, the pairwise secrets) are 128 bits long, the ten-millionth subscriber will require about 160 MB of storage. The maximum network size is not a hard limit. Newer subscribers (with larger sequence numbers) will just need more storage for public values.

If *unlimited* network sizes are desired, scalable KPSs are viable options. Unlike MLS, scalable KPSs are susceptible to collusions. For an (n, p)-secure KPS, an attacker with access to secrets of n nodes can pool such secrets together to compute a fraction p of all possible pairwise secrets. As an example, if the subset keys and identity tickets (SKIT) [75] scheme with parameters $m = 32$ and $M = 2^{16}$ is used, every subscriber will receive a SIM card with $m = 32$ secrets. Each subscriber will also download mM encrypted tickets from Tingular's website (for 80-bit tickets each subscriber will need about 20 MB of storage). For computing any pairwise secret $m = 32$ hash/block-cipher operations will be required (instead of one for MLS). An attacker who has exposed secrets of $n = 45,000$ nodes can illegitimately compute one in 2^{64} pairwise secrets. If subscribers can afford 100 MB storage we can choose $m = 32$ and $M = 2^{16} \times 5$ to realize a scheme for

which $p(225,000) \approx 2^{-64}$ and $p(422,000) \approx 2^{-30}$. An attacker desiring the exploit the collusion susceptibility of SKIT will have to successfully tamper with and expose secrets from several hundred thousand SIM cards.

8.7 APALLS Protocol: Joining a Subnet

After receiving the secrets and the required public values, a subscriber can take part in any ad hoc subnet created by any subset of Tingular subscribers.

A subscriber A sends a probe $[A, c]$ (where c is a random challenge) to determine other subscribers within range. A node B in the neighborhood responds with $[B, A, K_{BA}[c]]$. In general, the node A may receive one response from every neighbor within the reliable delivery neighborhood[4] (RDN) of A. Let us assume that A receives responses from 3 nodes X, Y and Z in its RDN. Node A then chooses a random one-hop group secret G_A and broadcasts individual encryptions of the secret as $[X, Y, Z, K_{AX}[G_A], K_{AY}[G_A], K_{AZ}[G_A]]$ to its neighbors. In response, A's neighbors X, Y, Z are expected to send their respective one-hop group secrets (G_X, G_Y, G_Z) to A.

Every packet aired by a node A is encrypted with the secret G_A. In other words, every node enforces a private logical neighborhood (PLN) [3]. The PLN of a node A is (in general) a subset of nodes in the RDN of A. Node A explicitly invites some nodes into its PLN by providing a secret G_A individually to such nodes. If a node A, with nodes X, Y and Z in its PLN, suspects X of i) violating the protocol, or ii) acting in a selfish manner, or iii) suspect a semi-active attacker between A and X, then A can simply cut-off X from

[4]The RDN of A includes all neighboring nodes with which A has confirmed bi-directional links.

its PLN by providing a new secret to Y and Z (and withholding the secret from X). In a scenario where A had provided its PLN secret G_A to X, but X did not reciprocate by sending its PLN secret G_X to A (or X did not accept A into its PLN), A will eject X from its PLN during the next broadcast by A - which will be encrypted using a new PLN secret $G_A{}'$, and the secret $G_A{}'$ conveyed only to nodes that A desires to retain in its PLN (and possibly other nodes that A desires to induct into its PLN). The PLN secret of A is updated whenever a node leaves the PLN, or is ejected by A from its PLN.

Every transmission by A is monitored by all nodes in the PLN of A, and checked for consistency with the protocol. Nodes that are observed to violate rules face the risk of being ejected from the PLN of the observer. Every node also maintains a revocation list periodically disseminated by the TA. The revocation list can be a list of sequence numbers signed using the TA's private key. Nodes are expected check the revocation lists before accepting a node into their PLNs. Nodes also broadcast their public key certificates to their PLN neighbors. The certificates are decrypted, and the authenticated public key of neighbors are cached.

8.8 RREQ Propagation in APALLS

We shall now look at the the sequence of operations involved in RREQ propagation from a source S to a destination T, through a path $(A, B, C, \ldots)$. The sequence of operations are shown in Eqs (8.3 - 8.7). To facilitate comparison with Ariadne the sequence of operations performed in Ariadne are also shown in the left column of Eqs (8.3 - 8.7).

8.8.1 RREQ Source S:

The RREQ $\mathcal{Q}^S_{(q,S)}$ from source S specifies a maximum hop-count n_h. The RREQ can (optionally) include a list of nodes (BL) black-listed by the RREQ source. The RREQ $\mathcal{Q}^S_{(q,S)}$ includes the digital signature Σ_S of the source. In addition, the broadcast by S includes a per-hop hash β_S (as in Ariadne) intended only for nodes in the PLN; In Table II values which are not carried forward (and intended only for nodes in the PLN) are shown enclosed in flowered braces.

At Node S

$$
\begin{array}{ll}
\mathcal{Q}_{(q,S)} = [S, q, T, t_i] & \mathcal{Q}_{(q,S)} = [S, t, T, n_h, \langle BL \rangle] \\
\beta_S = h(\mathcal{Q}^S_{(q,S)}, \bar{K}_{ST}) & \beta_S = h(\mathcal{Q}_{(q,S)}, K_{ST}), u = h(\beta_S) \\
 & \Sigma_S = \langle \mathcal{Q}_{(q,S)}, u \rangle_S \\
\mathcal{Q}^S_{(q,S)} = [\mathcal{Q}_{(q,S)}, R_S^{n_r - q}] & \mathcal{Q}^S_{(q,S)} = [\mathcal{Q}_{(q,S)}, u, \Sigma_S] \\
S \rightarrow * \; K_U[\mathcal{Q}^S_{(q,S)}, \{\beta_S\}] & S, G_S[\mathcal{Q}^S_{(q,S)}, \{\beta_S\}]
\end{array}
\tag{8.3}
$$

The PLN secret is used for encrypting all transmissions to neighbors. Every transmission by a node X is prepended with the identity X of the node (in the clear) to enable the receiver to determine that the secret G_X should be used for decrypting the packet.

8.8.2 Intermediate Nodes

Intermediate nodes verify the signature of the RREQ source. The public key certificate C_S required for verification of the signature Σ_S of the source can be included in the first RREQ sent by S in the subnet. Certificates are cached by other nodes in the subnet. If a node receiving the RREQ from S does not have access to C_S, it requests the upstream node

to provide the certificate. Nodes will not rebroadcast the RREQ until they gain access to the public key of the source. If an intermediate node is included in the list BL, it simply ignores then RREQ. Similarly, if an intermediate node receives an RREQ forwarded by a node in the list, the RREQ is ignored.

A neighbor A downstream of S decrypts the broadcast from S (using the PLN secret G_S provided by S) and verifies the signature Σ_S. Every intermediate node appends three values that are carried forward all the way to the destination. The fields $\mathcal{Q}^A_{(q,S)}$ broadcast by A include the triple (A, M_A, ν^S_A). M_A is a HMAC for the destination T computed using the secret K_{AT}. The value $\nu^S_A = K_{AT}[\beta_S]$ (which is not present in Ariadne) is the "encrypted upstream per-hop hash" [8].

At Node A	
$\beta_A = h(\beta_S, A)$	$\beta_A = h(\beta_S, A)$
	$\nu^S_A = K_{AT}(\beta_S)$
$M_A = h(\mathcal{Q}^S_{(q,S)}, \beta_A, K^{L-i}_A)$	$M_A = h(\mathcal{Q}^S_{(q,S)}, \nu^S_A, \beta_A, K_{AT})$
$\mathcal{Q}^A_{(q,S)} = [\mathcal{Q}^S_{(q,S)}, (A, M_A)]$	$\mathcal{Q}^A_{(q,S)} = [\mathcal{Q}^S_{(q,S)}, (A, \nu^S_A, M_A)]$
	$\Sigma_A = \langle \mathcal{Q}^A_{(q,S)}, \beta_A \rangle_A$
$A \rightarrow * \; K_U[\mathcal{Q}^A_{(q,S)}, \{\beta_A\}]$	$A, G_A[\mathcal{Q}^A_{(q,S)}, \{\beta_A, \Sigma_A, NULL, NULL\}]$

(8.4)

At Node B

$$
\begin{array}{ll}
\beta_B = h(\beta_A, B) & \beta_B = h(\beta_A, B) \\
 & \nu_B^A = K_{BT}(\beta_A) \\
M_B = h(\mathcal{Q}_{(q,S)}^A, \beta_B, K_B^{L-i}) & M_B = h(\mathcal{Q}_{(q,S)}^A, \nu_B^A, \beta_B, K_{BT}) \\
\mathcal{Q}_{(q,S)}^B = [\mathcal{Q}_{(q,S)}^A, (B, M_B)] & \mathcal{Q}_{(q,S)}^B = [\mathcal{Q}_{(q,S)}^A, (B, \nu_B^A, M_B)] \\
 & \sigma_A = h(\Sigma_A), v_B = h(\sigma_A, NULL) \\
 & \Sigma_B = \langle v_B, \mathcal{Q}_{(q,S)}^B, \beta_B \rangle_B \\
B \rightarrow * \; K_U[\mathcal{Q}_{(q,S)}^B, \{\beta_B\}] & B, G_B[\mathcal{Q}_{(q,S)}^B, \{\beta_B, \Sigma_B, \sigma_A, NULL\}]
\end{array}
\tag{8.5}
$$

Intermediate nodes also append 4 values which are intended only for nodes in the PLN. The 4 values broadcast by a node C are i) the per-hop hash β_C (as in Ariadne); ii) hash of the signature of the previous hop B - or $\sigma_B = h(\Sigma_B)$; iii) a value v_B, which is a one-way function of signatures appended by all nodes upstream of B; and iv) the signature Σ_C, computed over the values $\mathcal{Q}_{(q,S)}^C$, $v_c = h(v_B, \sigma_B)$, and β_C. For the first node in the path (A) the values σ, and v are not required. For notational consistency values that are not required are indicated as NULL. For nodes like B that are in the PLN of the A, the value v_A is NULL (as there is no intermediate node upstream of A).

At Node C

$$
\begin{array}{ll}
\beta_C = h(\beta_B, C) & \beta_C = h(\beta_B, C) \\
 & \nu_C^B = K_{CT}(\beta_B) \\
M_C = h(\mathcal{Q}_{(q,S)}^B, \beta_C, K_C^{L-i}) & M_C = h(\mathcal{Q}_{(q,S)}^B, \nu_C^B, \beta_C, K_{CT}) \\
\mathcal{Q}_{(q,S)}^C = [\mathcal{Q}_{(q,S)}^B, (C, M_C)] & \mathcal{Q}_{(q,S)}^C = [\mathcal{Q}_{(q,S)}^B, (C, \nu_C^B, M_C)] \\
 & \sigma_B = h(\Sigma_B), v_C = h(\sigma_B, v_B) \\
 & \Sigma_C = \langle v_C, \mathcal{Q}_{(q,S)}^C, \beta_C \rangle_C \\
C \rightarrow * \; K_U[\mathcal{Q}_{(q,S)}^C, \{\beta_C\}] & C, G_C[\mathcal{Q}_{(q,S)}^C, \{\beta_C, \Sigma_C, \sigma_B, v_B\}]
\end{array}
\tag{8.6}
$$

At Node D

$$
\begin{array}{ll}
\beta_D = h(\beta_C, D) & \beta_D = h(\beta_C, D) \\
 & \nu_D^C = K_{CT}(\beta_C) \\
M_D = h(\mathcal{Q}_{(q,S)}^C, \beta_D, K_D^{L-i}) & M_D = h(\mathcal{Q}_{(q,S)}^C, \nu_D^C, \beta_D, K_{DT}) \\
\mathcal{Q}_{(q,S)}^D = [\mathcal{Q}_{(q,S)}^C, (D, M_D)] & \mathcal{Q}_{(q,S)}^D = [\mathcal{Q}_{(q,S)}^C, (D, \nu_D^C, M_D)] \\
 & \sigma_c = h(\Sigma_C), v_D = h(\sigma_c, v_C) \\
 & \Sigma_D = \langle v_D, \mathcal{Q}_{(q,S)}^D, \beta_D \rangle_D \\
D \rightarrow * \; K_U[\mathcal{Q}_{(q,S)}^D, \{\beta_D\}] & D, G_D[\mathcal{Q}_{(q,S)}^D, \{\beta_D, \Sigma_D, \sigma_C, v_C\}]
\end{array}
\tag{8.7}
$$

In both Ariadne and APALLS (and more generally, any DSR-based protocol) a node with k neighbors will receive k RREQs (one from each neighbor) with the same source and sequence number. In Ariadne an intermediate nodes needs to store only one RREQ (corresponding to which an RREQ was relayed by the node), for a small duration, within which

it is reasonable to expect an RREP. In APALLS an intermediate node with k neighbors will need to verify $k+1$ signatures (the signature of the RREQ source and the signature appended by k neighbors). All k RREQs (along with the 4 one hop values) are cached for a small duration. Each node broadcasts one signed RREQ. While the signature of the RREQ source is verified by all nodes, signatures of intermediate nodes are verified *only* by PLN neighbors.

8.8.3 Route Response

In Ariadne if any inconsistency is detected the destination simply drops the RREQ. In APALLS the destination takes some proactive steps to isolate the problem.

The destination computes $\beta_S = h(Q_{(q,S)}, K_{ST})$ in the same manner computed by the RREQ source. The destination then proceeds to check the *self-consistency* and *consistency* of every node in the path. The values appended by an intermediate node C, viz., M_C and ν_C^B are deemed self-consistent by the destination T if

$$M_C = h(\mathcal{Q}_{(q,S)}^B, (C, \nu_C^B), h(K_{CT}^{-1}[\nu_C^B], C), K_{CT}). \tag{8.8}$$

A self-consistent node C, with an upstream node B is deemed *consistent* only if the per-hop hash C claims to have received from B, viz, $h(K_{CT}^{-1}[\nu_C^B])$, matches what B claims to have broadcast, viz., $h(K_{BT}^{-1}[\nu_B^A], B)$. Verifying the consistency of C is possible only if its upstream node B is found to be self-consistent, or

$$M_B = h(\mathcal{Q}_{(q,S)}^A, (B, \nu_B^A), h(K_{BT}^{-1}[\nu_B^A], B), K_{BT}). \tag{8.9}$$

Corresponding to every path through which the destination T receives an RREQ with *no* inconsistencies, the destination invokes an RREP of type $SUCC$. For an RREQ received through a path (A, B, W, G, H, P), the RREP is of the form

$$\mathcal{P}_q = [(u, \beta_S), \{SUCC : (A, B, W, G, H, P)\}] \tag{8.10}$$

The RREP is authenticated to the source using a HMAC based on the secret K_{ST}. The value u (which was also included in the RREQ) is a convenient index to the RREQ. That the RREP includes β_S (the preimage of u) informs the intermediate nodes that the destination did indeed invoke an RREP.

Some paths may contain one or more inconsistent nodes. As an example, consider a scenario where an RREQ indicates a path (A, B, C, D, E, F), and the destination determines that while D, E and F are consistent, C's consistency cannot be verified the values appended by B were not self-consistent. The RREP sent by T is then

$$\mathcal{P}_q = [(u, \beta_S), \{FAIL : ((B\lambda C), (D, E, F))\}] \tag{8.11}$$

The RREP is authenticated individually (using individual HMACs) for verification by the consistent nodes D, E, F and the last self-consistent node C. The special code λ indicates that an inconsistency was detected in the RREQ. Nodes that are identified as consistent (D, E and F) consider this RREP as an instruction to drop subsequent RREQs from S to T *if* they include C or B in the path (for example, if the source S sends a second RREQ after the first one times out). This is also considered by nodes D and C as a request to store RREQs they had received from their respective upstream nodes (C and B) in non

volatile storage, for submission to the TA at the earliest opportunity (for example, when the node has access to the Internet).

8.9 Rationale and Security Analysis

Some significant differences between Ariadne and APALLS are i) use of pairwise secrets instead of TESLA; ii) mandating the RREQ source to append a digital signature; iii) enforcing a PLN; iv) use of an additional upstream per-hop hash; and v) a digital signature appended by every node broadcasting an RREQ, which is intended only for one-hop neighbors.

8.9.1 Choice of Cryptographic Authentication Strategies

While Hu et al [7] preferred TESLA, Ariadne was also designed to support pairwise secrets (Ariadne-PS) or digital signatures (Ariadne-DS) instead of TESLA.

In Ariadne-DS every intermediate node appends a digital signature (instead of an HMAC). The signatures are carried forward all the way till the destination. The obvious disadvantage of using digital signatures is the overhead. This is exacerbated in Ariadne due to the fact that signatures and public key certificates appended by every node in the RREQ path have to be carried over all the way to the destination. Another disadvantage is the increased susceptibility to trivial denial of service (DoS) attacks due to the high verification complexity. An attacker can send random "signed packets" which can be recognized as bogus only after the expensive verification process.

When pairwise secrets are used (Ariadne-PS) the HMACs by intermediate nodes are computed as in APALLS. The end-points S and T can readily use the secret K_{ST} for authenticating RREQ/RREP instead of relying on an out-of-network mechanism for establishing an application layer secret $\bar{K}_{ST}$.

The primary differences in rationale for the choice of key distribution strategies in Ariadne and APALLS stem from the differences in the assumed network model. More specifically, Ariadne assumes a trusted entity in every subnet. APALLS on the other hand, motivated by the fact that one of the most compelling advantages of MANETs is the ability to operate without infrastructural assistance (like a trusted entity in every subnet), assumes an open-managed network model.

The reason Hu et al [7] preferred TESLA over pairwise secrets was that "broadcasting N commitments is easier than distributing $\binom{N}{2}$ secrets." This is indeed true if a trusted entity is available in every subnet to broadcast commitments. Broadcasting say $N_s = 200$ (where N_s is the number of nodes in a specific subnet) TESLA commitments within a subnet is indeed preferable to unicasting $\binom{N_s}{2} = 19,000$ values (or unicasting 199 unique values to every node).

For large scale dynamic open-managed MANET networks ($N(t)$ of the order of several millions) distributing TESLA commitments can be substantially more expensive. As no TA is available in the subnet to broadcast authenticated commitments, authenticated TESLA commitments have to be distributed by the nodes themselves. The only practical approach may be for each node to possess a certified asymmetric key pair which is used for their own authenticating TESLA commitments. As TESLA HMACs can be verified

only at the end of the reverse path it is not necessary to flood TESLA commitments to all nodes in the subnet. An intermediate node X will need to release (during the RREP), a value from its TESLA chain, and in addition, i) the TESLA commitment; ii) a digital signature of X for the commitment; and iii) a certificate (issued by the TA) authenticating the public key of X. For RREPs with long paths the overhead can be substantial.

8.9.2 Pairwise Secrets Instead of TESLA

For Ariadne-PS no additional values will need to be released during the reverse path. As the destination can verify the authentication appended by intermediate nodes, both deletion and insertion attacks can be detected at the end of the forward path by the destination. Furthermore, issues related to the delay sensitivity of TESLA based authentication [8] can also be avoided. Note that when TESLA is used the source (which in general may not know the distance to the destination) will have to choose a conservative (large) value of time t_i to ensure that the RREQ will reach the source before t_i. Unfortunately as the pre images can be released by the intermediate nodes only after time t_i, the reverse path will have to be delayed till time t_i (even if the RREQ reached the destination well before t_i).

Most ad hoc routing protocols, in their original incarnations, included several optimizations for improving their performance. However, as such protocols implicitly assumed that all participants are trustworthy, they were not constrained by the need to actually *enforce* the additional (more complex) rules associated with such optimizations. For example, the original DSR protocol includes several optimizations that employ cached routes to facilitate route response by intermediate nodes. Some attempts to salvage broken paths [44]

during the RREP have also been considered, although without considering security issues. In most secure extensions of such protocols (including Ariadne), all such optimizations cannot be used. However, one of the more compelling advantage of using pairwise secrets instead of TESLA is that it is now feasible to use some optimizations (that cannot be used in Ariadne-TESLA).

The reason that optimizations involving salvaging of routes cannot be used in Ariadne-TESLA is due to the fact intermediate nodes do not share a secret with *all* nodes. Consider a scenario where an RREP instantiated in response to an RREQ indicating a path (Q, R, J, G, M, N, W, P) fails when G is unable to unicast the RREP to J. This situation may occur if J had moved away from the path, or if J is a malicious node which simply ignores the RREP. However, even if nodes cache RREQs only for small duration, G may have in its cache other RREQs from S indicating an alternate path, say (A, B, C, F), from S. To use this hitherto untested partial path $G \rightarrow F \rightarrow C \rightarrow B \rightarrow A \rightarrow S$ to relay the RREP around J, it is necessary for G to share a secret with S. This is obviously not possible when we rely only an *application* layer shared secret between end-points (as in Ariadne-TESLA) as G would not have a readily available shared secret (as G has no prior knowledge of its need to establish a path to S). However, for Ariadne-PS, G can salvage the path as it shares a network layer secret K_{GS} with S.

8.9.3 RREQ Signatures

In Ariadne the choice of the strategy for controlling RREQ floods is also based on the implicit assumption that a trusted entity is available in every subnet, or more specifically,

that the authenticated commitments distributed by the TA are constrained in scope to a specific subnet. In the absence of such an entity in each subnet, an attacker can collect sequence-number hash chain values made public in one subnet to flood RREQs in other subnets. Thus, for the assumed open-managed network model, sequence-number chains are not useful.

In Ariadne the value from the RREQ hash chain released by a node A for a RREQ with sequence number q can be verified by an node to be consistent with S and q. Thus it provides implicit authentication to two fields (source and sequence number). However, it does not protect other fields like the destination or the time t_i. Thus, if an active attacker forwards the RREQ after changing the destination field, such an RREQ will be propagated by downstream nodes. Only the destination will be able to detect the inconsistency as it will compute an inconsistent β_S.

To authenticate all RREQ fields specified by the source Hu et al suggested the use of chained one-time signature (OTS) schemes [76]. In a chain $C_0, C_1, \ldots C_L$, values $(C_i, C_{i+1}), 0 \leq i \leq L-1$ are keys pairs of an OTS scheme. The signer can use C_i to compute a signature for the $L - i^{\text{th}}$RREQ, which can be instantaneously verified by any node with access to C_{i+1}. The next RREQ from the source will make C_i public and use C_{i-1} to sign the RREQ. However, this approach also implicitly relies on the same assumption - that the network is the subnet (if different subnets can exist at the same time as in the open-managed model, a value made public in one subnet can be used in another subnet).

Mandating digital signatures by RREQ source can prevent such attacks. Furthermore, as RREQs have to be authenticated, if a node S floods too many RREQs other nodes will simply drop RREQs from S in the future.

8.9.4 Rationale for PLN

Perhaps the most important implication of the fact that inexpensive schemes for computing pairwise secrets exist (MLS requires only one hash computation) is that it becomes practical to enforce PLNs, and derive many tangible benefits out of this ability. More specifically, enforcing the PLN is intended to achieve the following goals: i) eliminate trivial risk-free active attacks; ii) validate the assumption behind the security of the per-hop hashing strategy; iii) avoid one-way links and "links" created by semi-active attackers; iv) deter selfish behavior.

The PLN also prevents simple DoS attacks. Note that APALLS employs two independent forms of link-layer authentication: one based on the one-hop PLN secret, and one based on digital signatures. The PLN authentication serves as a protection against DoS attacks that could be launched due to the higher computational overhead required for verifying digital signatures. RREQ packets relayed by a node C are first decrypted using PLN secret G_C. Only if the resulting packet is a valid[5] RREQ packet will the receiver proceed to verify the signature Σ_C.

[5]Valid RREQ packets may start with a magic number. In addition, for a packet from C, the last entry in the path field should be C.

8.9.5 Identifying and Revoking Active Attackers

Enforcing a PLN does *not* address *all* risk-free attacks. As an illustration, consider a scenario where C receives a RREQ (from S to T) along a path (A, B), and relays the RREQ indicating a path (Q, R, C) instead, where Q and R are fictitious nodes inserted by C. Nodes downstream of C have no reason to suspect that Q and R do not exist, and B does not have access to β_S and hence $\beta_Q = h(\beta_S, Q)$ or $\beta_R = h(\beta_Q, R)$ to verify that the value β_C is indeed inconsistent.

Note that *if* C had instead advertised a path (A, R, C) with a random β_C, B (which has access to β_A) can determine that $\beta_C \neq h(h(\beta_A, R), C)$. Similarly, if C had modified any of the fields specified by the "real" upstream nodes (A and B), then B can recognize such attempts. Thus, while there are some *blatant* active attacks which can be easily be recognized by neighbors, some subtler attacks can not.

Assume that the destination receives the tainted RREQ indicating a path (Q, R, C, D, E, F, G) and a per-hop hash β_G. Assume that the actual reason for the inconsistency in the RREQ was that C had performed an active attack. In Ariadne all that the destination can detect at this point is that "the per-hop hash β_G is inconsistent." In Ariadne-DS and Ariadne-PS T can also conclude with certainty that node G exists (as T can verify the HMAC / signature of G). However, T cannot verify the authentication appended by F as T does not access to the value β_F (which had gone into the computation of the authentication appended by F). Thus T cannot even determine if the node F actually exists in the path.

If T desires to determine *who is responsible* for perpetrating this attack, it can come to several likely conclusions: like i) G is a malicious node and every other node in the path

has been maliciously inserted by G; or ii) G is a good node, but F may have maliciously inserted nodes (A, B, C, D, E) in the path; or iii) both G and F are good nodes and the node E may have inserted nodes (A, B, C, D) in the path; and so on.

In Ariadne-DS the destination can demand all intermediate nodes (A, B, C, D, E, F, G) to produce the per-hop hash they had received from *their* upstream neighbor, which is simultaneously consistent with the signature of the upstream node (which was already included in the RREQ sent to the destination). Now node D can produce a value β_C consistent with the signature Σ_C, and $\beta_D = h(\beta_C, D)$ consistent with D's signature Σ_C. Likewise, all nodes that had *not* violated the protocol can also do so.

However, the attacker C cannot produce a value β_R consistent with the "signature Σ_R." The obvious recourse for C is to not respond to this demand (C could just power off or leave the subnet). Note that it is not possible to compute the value β_R (which according to C, was sent by R) from $\beta_C = h(\beta_R, C)$. It is thus possible that Σ_R is indeed consistent with that β_R. If C has to be convicted based on its inability to provide an "affirmative defense" (providing β_R consistent with Σ_R) it is indeed possible that an innocent D, which had suddenly crashed (and thus loses the value β_C) can also suffer the same fate.

8.9.6 Nonrepudiable Proof of Active Attacks

The encrypted upstream per-hop hash ν in APALLS serves two purposes. Firstly, it makes it possible for the destination to narrow down active attackers. Assume that in the path (A, B, C, D, E, F, G) the destination is able to determine that nodes (D, E, F, G) were consistent, and C, while self-consistent, cannot be verified to be consistent (as B is

self-inconsistent). The destination can now narrow down the active attacker to B or C. Secondly, when used in conjunction with one-hop signatures, it facilitates unambiguous identification of active attackers, and avoids the need for nodes to provide affirmative defense.

In other words, even without carrying over all signatures (thereby saving bandwidth overhead for signatures and public-key certificates) APALLS can provide non repudiable proof of active attacks. Irrespective of the nature of the active attack, a signed packet from the attacker (stored temporarily by a neighbor, and submitted to the TA at a convenient time) can be used for this purpose.

Note that the values broadcast by C is effectively a non repudiable statement to the effect "the fields $\mathcal{Q}^B_{q,S}$, $\beta_B = K^{-1}_{CT}[\nu^B_C]$, and v_B, were broadcast by B, and verified by me (C) to be consistent with the signature of B (Σ_B), the preimage of σ_B."

When the values stored by D (the contents of the RREQ broadcast by C) are submitted to the TA, the TA takes the following steps:

① Verify that Σ_C is consistent with $\mathcal{Q}^C_{q,S}$, β_C, σ_B and v_B;

② Check if B is a valid node in the network; if not, C is an active attacker (C had inserted a nonexistent node in the path);

③ If B is a valid node, compute the signature $\Sigma_B{}'$ for the values $\mathcal{Q}^B_{q,S}$ and $\beta_B = K^{-1}_{CT}[\nu^B_C]$ and v_B (which according to C, were broadcast by B), and

④ Verify if $h(\Sigma_S{}') = \sigma_B$. If so, B is an active attacker (as B advertised self-inconsistent values (B, M_B, ν_B)). If not, C is the active attacker (as C had accepted a packet with an invalid signature).

If the TA has access to the private keys of all nodes the TA can simply compute $\Sigma_B{}'$. If private keys are *not* escrowed by the TA, the TA will need to request B to produce a verifiable signature $\Sigma_S{}'$ for the values $\mathcal{Q}^B_{q,S}$ and $\beta_B = K_{CT}[\nu^B_C]$ and v_B. Thus even in scenarios where the private keys are not escrowed by the TA, unlike Ariadne-DS, nodes will only need access to their private key to avoid being penalized (revoked[6]) accidentally.

An advantage of escrowing private keys by the TA is that the verification of proof of attacks can be performed immediately. This is especially useful in scenarios where access to the TA is available (for example, if at least one node in the subnet has Internet access), as the revocation message (signed by the TA) can be immediately distributed within the subnet.

8.9.7 Routing Around Attackers

In scenarios where access to the TA does not exist, nodes in the subnet will have to "live with" active attackers for some (indefinite) duration. APALLS includes two strategies for improving the ability to route around nodes suspected of active attacks. The first is by using black-lists specified by the RREQ source. The second is by employing RREPs with a $FAIL$ code.

The list of nodes in S's black list can include nodes which were possibly S's neighbor at some time in the past, and observed by S to violate the protocol, or engage in selfish behavior. The list can also include nodes which have been recognized as active attackers when S was a destination node in some RREQ. That a node X is black-listed by a node

[6]Any node which claims to not have access to its private key should be revoked in any case.

S is not interpreted by other nodes to mean that "X is malicious." All this means is that the source S desires to avoid X in paths where S is an end-point. Thus, the black-list of S will only influence routing of RREQ packets in which S is the source or the destination.

RREPs with $FAIL$ code are intended to improve the success of the second RREQ that may be sent by the source S after the first RREQ times out. For instance, in a scenario where the $FAIL$ RREP indicates $[(B\lambda C), (D, E, F, G)]$, during the second RREQ nodes (D, E, F, G) will drop RREQs that include C or B. Note that without this measure (and if the subnet topology has not changed) the second RREQ may suffer the same fate as the first.

To evaluate the benefit of employing $FAIL$ RREPs simulations were performed with random realization of subnets. Such subnets include $N_s = 200$ nodes with randomly distributed x and y coordinates, in a square region with unit edges. The transmission range of the nodes was chosen as 0.1 units. Of the $N_s = 200$ nodes, b randomly chosen nodes were labeled malicious. RREQ propagation was simulated between every pair of nodes.

Three different realizations of the network were simulated with different sets and numbers of "bad" nodes. In Figure 8.1 the y-axis is the fraction of node-pairs that succeed in discovering a path free of bad nodes. RREQ propagation was simulated between over 400,000 pairs of nodes. Path discovery between a pair is assumed to succeed if at least one of the established paths does not include any of the b "bad" nodes. The x-axis represents the shortest number of hops between the pair of nodes (between which RREQ propagation was simulated).

In Figure 8.1 plots labelled S1 depict the success rates of first RREQs. Simulation results are shown for $b = 15$ (S1-15) and $b = 30$ (S1-30). The plots labelled S2 indicate fraction of successful node pairs after the second RREQ (either the first or the second RREQ attempt succeeds). As can be seen from the simulation results the success rate after the second RREQ for the scenario with 30 bad nodes (S2-30) is comparable to the success of the first RREQ with just 15 bad nodes (S1-15). Note that in the absence of this strategy, the second RREQ has only as much chance of succeeding as the first. Thus, in this particular instance, it can be argued that using the $FAIL$ code RREP helps in realizing a two-fold improvement in resistance to malicious nodes in the subnet.

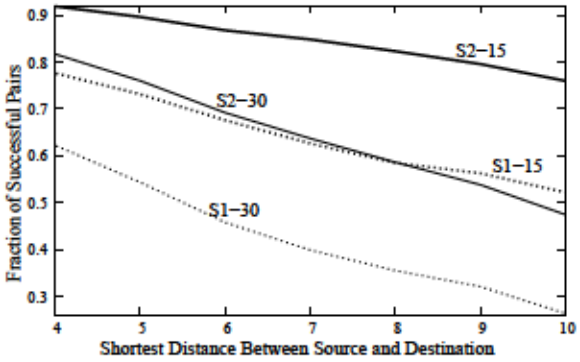

Figure 8.1

Simulation results depicting the utility of using RREPs with $FAIL$ code

8.9.8 RREP Authentication

In Ariadne the RREP is authenticated by the RREQ destination T to the source S using a HMAC employing the secret $\bar{K}_{AB}$. This HMAC is indistinguishable from a random

number for all intermediate nodes that relay the RREP. This can be exploited by attackers to send spurious RREPs over long (fictitious) paths to cause unnecessary bandwidth overhead for other nodes in the subnet. Nodes specified in the path will simply forward the RREP along the path specified.

This attack is particularly effective in Ariadne (if it is used for the open-managed network model assumed in this paper) as every intermediate node will need to release a TESLA key *and* a certificate for a commitment. Consider a scenario where an RREQ from a source S to some destination T indicates t_i (as the upper limit before which the destination T should receive the RREQ). Assume that such an RREQ through a path (K, L, M) is over-heard by an attacker W. Just by overhearing any RREP packet in response to *any* RREQ (not necessarily a response for the RREQ from S) *after* time t_i, it is possible for the attacker W to harvest a preimage K_X^i corresponding to time t_i of some node X. The node X may even be many hops away from W. A malicious W can now send a fictitious RREP indicating a path (K, L, M, X) to M with a random HMAC by "destination T". All that nodes (K, L, M) can verify is that K_X^i is indeed i^{th}pre-image of K_X^0. Obviously this serves very little purpose without the ability to recognize the authentication appended by the RREP destination (which conveys the crucial information that the HMACs were seen *before* time t_i by the destination). Effectively, *any* node can send such spurious RREP packets in response to any RREQ packet, impersonating some other node which may be several hops away.

In APALLS the destination includes a value β_S in the RREP which was until then known only to the source and destination. Thus, even while supercilious RREPs can be

sent by nodes (which will be detected by the source as inconsistent), such RREPs can be raised only by nodes which had actually seen an RREP from the destination. Furthermore, such an attack is not worthwhile for any attacker as the RREP overhead is small in any case in APALLS (as no additional values need to be released by intermediate nodes).

8.10 Conclusions

We have outlined a comprehensive secure routing protocol, APALLS, based on DSR. To the extent of our knowledge, APALLS is the first secure routing protocol which is designed to provide non repudiable proof of active attacks.

Non repudiable authentication, while necessary, is not sufficient to provide non repudiable proof of active attacks. In general, any active attack involves violation of the prescribed protocol. The protocol prescribes the steps that a node (say) C should take in response to a packet sent from a neighbor (say) B. For example, in distance vector protocols, if a node B announces a distance of 5 to a node S, the neighbor C downstream of B is expected to announce a distance 6. In a scenario where C advertises a distance 7, proving that C did (or did not) violate the protocol requires several other pieces *contextual* information like (for example) i) if B was indeed a neighbor of C at that time; ii) the distance advertised by B at that time ; iii) if C did indeed process the information advertised by B (the packet broadcast by B did not suffer collision), etc..

Thus, even while some ad hoc routing protocols like ARAN [37] and Ariadne-DS employ non repudiable authentication, they do not address the issue of *how* a packet sent from a node can be used as a proof of an active attack. As pointed out in this paper, even

while Ariadne-DS carries forward all signatures, it still has practical issues in providing non repudiable proof.

One motivation for APALLS stems from the fact that the main advantage of MANET based networks is their ability to operate without any infrastructural support. Ideally, while we would desire to eliminate even an off-line TA, this is simply not possible to do so as an authority is required to i) specify the rules (the protocol) that should be followed by every node; and ii) to boot-strap cryptographic associations between nodes.

While APALLS borrows some features from Ariadne, the major differences between Ariadne and APALLS stem from the network model. Several elements in Ariadne like i) the preference of TESLA over pairwise secrets; ii) the choice of the strategy to suppress RREQ floods; and iii) ignoring the risk of supercilious RREPs (RREP bandwidth can be high if Ariadne is used for a managed-open network) assume the presence of a TA in every subnet. While APALLS can take advantage of access to TA (when at least one node in the subnet has access to the Internet) for quickly disseminating revocation lists, APALLS can operate effectively even in subnets that may be completely isolated from the rest of the world for long periods. The only limitation in such subnets is that nodes will not be able to receive fresh revocation lists until they have access to a wide-area network.

The choice of cryptographic authentication schemes in APALLS are also driven by the need to keep the overhead low. Storage is an inexpensive resource for mobile devices; any mobile device can easily afford several GBs of pluggable storage. However computation and bandwidth are expensive for battery operated devices. This renders key predistribution schemes for pairwise secrets (which impose low computation and bandwidth overhead)

well suited even for dynamic large scale networks. More specifically, schemes like MLS [73] are well suited for scenarios where a soft limit on the maximum network size (a few tens of millions) is acceptable. Scalable key predistribution schemes (KPS) like SKIT [75] are well suited for unlimited network sizes. While scalable KPSs are susceptible to collusions, SKIT can realize very high levels of collusion resistance by taking advantage of abundant storage resources.

Digital signatures are used only at the link-layer in order to keep the overhead (for signatures and certificates) under control. That digital signatures appended by intermediate nodes are verified only by neighbors renders just about any scheme adequate for this purpose. More specifically, it also opens up the feasibility of non repudiable one-time signature (OTS) schemes[7] which require only block-cipher/hash operations. That only neighbors need to verify the signature renders the scheme proposed by Merkle et al [77] for constructing infinite OTS trees substantially more efficient. That OTS schemes require only block-cipher/ hash operations implies that even very low complexity SIM cards can perform the operations required for this purpose. Such low complexity SIM cards can be realized at a lower cost.

[7]This does *not* include chained OTS schemes which cannot be used for non repudiation as private keys are revealed eventually.

CHAPTER 9

CONCLUSIONS AND FUTURE RESEARCH

This chapter summarizes the contributions of this dissertation to improve secure routing protocols and outlines some possible extensions of this research.

9.1 Contributions

This dissertation makes several contributions to improve secure routing protocols.

The first contribution of the dissertation is a generic three-step requirement model for secure routing protocols, consisting of

1. detection of inconsistencies;
2. identification of perpetrators; and
3. defensive strategies to reduce participation of attackers in the subnet routing protocol.

This generic requirement model enabled us to immediately identify that the main shortcomings of existing (when this dissertation research began) routing protocols in the literature, viz,

1. existing protocols cater only for the first step, viz., detection of inconsistencies (and dropping packets with inconsistencies).
2. even the first step was not assured as i) they ignored the need for proactive measures to ensure that wireless links between nodes are bidirectional; and ii) did not possess strong link-level authentication.

3. some important questions in the context of detecting inconsistencies have been ignored; specifically, the whos and whens (who detects inconsistency and when) have not been considered by existing protocols.

9.1.1 Ensuring Bidirectional Links

Our research led to two strategies for ensuring bidirectional links. The first based on propagating warnings, which led to a conference publication [2]. The second approach was by enforcing private logical neighborhoods (PLN). While our initial results indicated that the first approach (disseminating warnings) is better, some recent developments indicated that one of the main perceived disadvantages of imposing a PLN - the overhead for maintaining a private logical neighborhood - is not high. Key predistribution schemes provide a viable and low overhead strategy for establishing PLNs. Apart from simultaneously addressing some of main pitfalls of many secure routing protocols (by catering for link-level authentication and addressing attacks that exploit one-way links) mandating PLNs also provides a sound basis for the security of efficient strategies like the per-hop hashing for carrying over authentication. Our research also identified several other benefits accrued by mandating PLNs. This research on PLNs and their uses lead to a refereed ACM conference publication [3].

9.1.2 Early Detection of Inconsistencies

Another important issue related to the problem of detecting inconsistencies is the issue of "who (detects inconsistencies) and when (such inconsistencies are detected)." Several factors that influence the choice of cryptographic authentication strategies for MANETs

were investigated. Sub optimal choice of cryptographic authentication strategies were found to contribute to several deficiencies and high overhead for many current secure routing protocols. While source authentication schemes seem more desirable than pairwise authentication schemes, they can demand very high overhead. Furthermore, much of the utility of source authentication schemes is lost in protocols which use per-hop hashing to protect node deletion attacks.

Our research demonstrated that broadcast encryption techniques have some unique advantages over approaches that use pairwise secrets or source authentication. The strategy of employing broadcast encryption was investigated for secure extensions of two popular protocols: AODV and DSR. In the two protocols that were developed multi-source broadcast encryption was used for two-hop authentication along with PLNs for one-hop authentication. The protocols were published in two conference articles - the secure route discovery (SRD) protocol based on DSR in [5], and the secure AODV protocol (SAODV2) based on AODV in [6].

Like most existing protocols, the scope of the protocols in [5] and [6] stop with detection of inconsistencies (and dropping offending packets). In this context, the approaches based on broadcast encryption facilitate *early* detection of inconsistencies, and thus reducing the ill effects of routing packets carrying incorrect information. Furthermore in [5] and [6], due to the use of PLN, neighbors will be able to monitor nodes. Repeated misbehavior can result in rejection a node from the PLN's of its neighbors.

9.1.3 Identification and Healing

The last two requirements of our three-step model, viz., i) identifying attackers, and ii) reducing their role in the subnet routing protocol are however intricately tied to the specifics of the routing protocol. For this reason our research focus was narrowed down to DSR. More specifically, our research focused on strategies to improve Ariadne [7], a popular secure extension of DSR, by adding explicit features for identification and healing.

The research identified many advantages of using pairwise secrets instead of TESLA (which is used by Ariadne). Unlike Ariadne with TESLA where only the source and destination nodes can establish a private channel, using pairwise secrets permits any two nodes to do so. We argued that this feature can be leveraged to improve the resiliency of Ariadne to malicious nodes. More specifically, this feature can be used to reduce the ambiguity in identification of active attackers and facilitate strategies to route around suspected active attackers. This research was presented in an IEEE conference [8].

While the improvements to Ariadne in [8] address all three steps it was realized that it is possible to identify (instead of merely narrowing down) attackers and also obtain non repudiable proof of misbehavior (instead of simply routing around nodes suspected of misbehavior). Our quest for a comprehensive secure routing protocol led to the development of APALLS [9]. Some of the issues addressed in the design of APALLS included i) efficient use of non-repudiable authentication to provide non repudiable proof of misbehavior; ii) advantageously employing the two fundamental tools (authentication and monitoring) to achieve the goals of the protocol; and iii) considering realistic network models for large

scale MANET deployments. This research lead to an article [9] submitted to an IEEE Transaction.

9.2 Scope for Future Work

Some of the obvious future extensions are applying the three-step requirement model to design comprehensive secure routing protocols based on other popular routing protocols like DSDV, AODV, and TORA.

For adding security features to routing protocols, two of the basic tools are i) cryptographic authentication and ii) monitoring. A third approach also exists in scenarios where it is possible for every node to employ a tamper-responsive chip or module. Design of protocols which employ all three tools, and the possibility of adapting existing protocols to productively employ such modules is a promising area of research.

Many of the issues associated with secure MANET protocols also apply in the more general problems of ubiquitous computing and autonomic computing. Like MANETs, autonomic computing networks require self-management, self-adaptation and self-protection. Application of some results of this research to securing ubiquitous and autonomic computing systems appears to be a promising avenue for future research.

9.3 Publications

The following publications resulted from the research performed towards the dissertation:

1. K.A. Sivakumar, M. Ramkumar, "On the Effect of Oneway Links on Route Discovery in DSR," Proceedings of the IEEE International Conference on Computing, Communication and Networks, ICCCN-2006, Arlington, VA, October 2006.

2. K.A. Sivakumar, M. Ramkumar, "Safeguarding Mutable Fields in the AODV Route Discovery Process," the Sixteenth IEEE ICCCN-07, Honolulu, HI, Aug 2007.

3. K.A. Sivakumar, M. Ramkumar, "An Efficient Secure Route Discovery Protocol for DSR," IEEE Globecom 2007, Washington, DC, Nov 2007.

4. K.A. Sivakumar, M. Ramkumar, "Improving the Resiliency of Ariadne," IEEE SPAWN 2008, Newport Beach, CA, June 2008.

5. K.A. Sivakumar, M. Ramkumar, "Private Logical Neighborhoods for Wireless Ad Hoc Networks," to be presented in ACM Q2SWinet, Tenerife, Canary Islands, Spain, October 2009.

6. K.A. Sivakumar, M. Ramkumar, "APALLS: A Secure MANET Routing Protocol," Submitted to the IEEE Transactions on Information Forensics and Security.

CHAPTER 10

GLOSSARY

MANET: Mobile Ad hoc Networks - a collection of small range portable devices that can communicate among themselves with out the need for any infrastructure.

DSR: Dynamic Source Routing protocol - an ad hoc on demand routing protocol used to establish the routes between two different nodes; DSR carries the entire list of nodes through which the route request has traversed.

AODV: Ad hoc on demand distance vector routing protocol; AODV does not carry the entire path in its route request packets.

DSDV: Destination Sequenced Distance Vector routing protocol; this is a proactive routing protocol based on Bellman-Ford algorithm; a route to every destination will be maintained at all times by every node.

RDN: Reliable delivery neighborhood - a set of nodes in a neighborhood with whom the existence of bidirectional links have been confirmed.

PLN: Private Logical Neighborhood - a subset of nodes in the RDN; nodes can choose set of nodes in their RDN.

RREQ: Route Request - a packet flooded through out the network in search of a path to a particular destination.

RREP: Route Response - a packet send by the destination or any intermediate node back to the source in response to the RREQ packet.

RERR: Route Error - a packet send by a node if it finds a link between nodes is broken.

MAC: Medium Access Control - provides addressing and channel access control mechanisms.

RTS: Request to Send - RTS message (identifies the sender and the intended receiver) and is used by a node to indicate its wish to transmit a frame to all neighbors.

CTS: Clear to Send - the receiving node can allow transmission by sending a CTS.

HMAC: Hash Message Authentication Code - generally employs cryptographic hash function along with a secret key; used to verify the authenticity of the message.

KDS: Key Distribution Scheme - strategy to provide secret keys or authenticated public values that can be used by two parties for secure communications.

KDC: Key Distribution Center - A system that manages, creates and distributes secrets.

TA: Trusted Authority - An entity trusted by all nodes in the network.

DS: Digital Signatures - Used to verify the authenticity of the sender.

REFERENCES

[1] Web Link, http://www.ietf.org/html.charters/manet-charter.html.

[2] K.A. Sivakumar, M. Ramkumar, "On the Effect of Oneway Links on Route Discovery in DSR," Proceedings of the IEEE International Conference on Computing, Communication and Networks, ICCCN-2006, Arlington, VA, October 2006.

[3] K.A. Sivakumar, M. Ramkumar, "Private Logical Neighborhoods for Wireless Ad Hoc Networks," ACM Q2SWinet, Tenerife, Canary Islands, Spain, October 2009.

[4] M. Ramkumar, "Broadcast Encryption Using Probabilistic Key Distribution and Applications," *Journal of Computers*, Vol **1** (3), June 2006.

[5] K.A. Sivakumar, M. Ramkumar, "An Efficient Secure Route Discovery Protocol for DSR," to be presented in IEEE Globecom 2007, Washington, DC, Nov 2007.

[6] K.A. Sivakumar, M. Ramkumar, "Safeguarding Mutable Fields in the AODV Route Discovery Process," the Sixteenth IEEE ICCCN-07, Honolulu, HI, Aug 2007.

[7] Y. C. Hu, A. Perrig, D. B.Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks," *The 8th ACM International Conference on Mobile Computing and Networking*, Atlanta, Georgia, September 2002.

[8] K.A. Sivakumar, M. Ramkumar, "Improving the Resiliency of Ariadne," IEEE SPAWN 2008, Newport Beach, CA, June 2008.

[9] K.A. Sivakumar, M. Ramkumar, "APALLS: A Secure MANET Routing Protocol," Submitted to the IEEE Transactions on Information Forensics and Security.

[10] Web Link , http://www.ietf.org/rfc/rfc2501.txt

[11] E. M. Royer and C. K. Toh, " A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, Apr. 1999.

[12] C E. Perkins and P Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," *ACM SIGCOMM Symposium on Communication, Architetcures and Applications*, 1994.

[13] C. C. Chiang, H. K. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks with fading channel," *Proceedings of IEEE Singapore International Conference on Networks (SICON 97)*, pages 197–211, April 1997.

[14] S. Murthy and J. J. G. L. Aceves, " An efficient routing protocol for wireless networks, " *ACM Mobile Networks and App. Journal., Special issue: routing in mobile communications networks*, Oct. 1996, pp. 183-97

[15] T Clausen and P Jacquet, "Optimised Link State Routing Protocol," *IETF Draft*, 2003. http://www.ietf.org/rfc/rfc3626.txt

[16] C. E. Perkins, E. M. B. Royer, and I. Chakeres," Ad hoc On Demand Distance Vector (AODV) Routing," *IETF Internet draft*, draft-perkins-manet-aodvbis-00.txt,Oct 2003.

[17] P. Johanson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, Kluwer Publishing Company, 1996, ch.5 , pp. 153-181.

[18] C-K. Toh, " Associativity-Based Routing for Ad-Hoc Mobile Networks," *Wireless Pers. Commun.,* vol. 4, no.2, Mar. 1997, pp. 1-36

[19] R. Dube et al., " Signal Stability based Adaptive Routing (SSA) for Ad-Hoc Mobile Networks," *IEEE pers. Commun.,* Feb 1997, pp. 36-45.

[20] Z. J. Haas and M. R. Pearlman, "The Zone routing Protocol (ZRP) for ad hoc networks", *IETF Draft*, 2002, http://www.ietf.org/proceedings/02nov/I-D/draft-ietf-manet-zone-zrp-04.txt.

[21] V. Ramasubramanian,Z. J. Haas and E. G. Sirer, " SHARP: a hybrid adaptive routing protocol for mobile ad hoc networks," *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing*,maryland ,USA,2003.

[22] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "*Introduction to Algorithms Second Edition,* " MIT PRESS, Cambridge, MA, 2001.

[23] P. Papadimitratos and Z. H. Haas, " Secure Routing for Mobile Ad hoc Networks, " *In proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, San Antonio, TX, January 27-31, 2002.

[24] M. G. Zapata, N.Asokan, " Securing Ad hoc Routing Protocols," *Proceedings of the ACM workshop on Wireless security*, Atlanta, Georgia, September 2002.

[25] A. Perrig, R. Canetti, J. D. Tygar, D. Song, "The TESLA Broadcast Authentication Protocol," RSA Crypto bytes, Summer 2002.

[26] Web Link, http://www.ctvr.ie/docs/secure-adhoc-routing.pdf

[27] J.Marshall, V.Thakur, A.Yasinsac,"Identifying flaws in the secure routing protocol," Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference, 2003.

[28] S. Bansal and M. Baker, "Observation based Cooperation Enforcement in Ad hoc Networks," *Technical Report*, 2003. http://citeseer.ist.psu.edu/article/bansal03observationbased.html.

[29] S. Bucheggar and J-Y. Le Boudec, "Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks," *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, Spain, 2002.

[30] P. G. Argyroudis and D. O'Mahony, " Secure routing for mobile ad hoc networks," *IEEE communications Surveys and Tutorials*, vol. 7, no. 3, pp. 2-21, 2005.

[31] M. Y. Alsaadi and Yi Qian, "Performance Study of a Secure Routing Protocol in Wireless Ad hoc Networks," *Second International Symposium on Wireless Pervasive Computing, ISWPC'07* 5-7 Feb 2007.

[32] Sonja Bucheggar and J.Y.L Boudec," Performance analysis of the CONFIDANT protocol," Mobihoc, Switzerland, 2002.

[33] A. A, Pirzad, Chris McDonald and Amitva Dutta, "Performance Comparision of Trust Based Reactive Routing Protocols," *IEEE Transactions on mobile Computing,* vol5, No 6, June 2006.

[34] Web Link, http://www.cse.fau.edu/jie/research/publications/Publicationfiles/trust.pdf.

[35] Y. C. Hu and A. Perrig, " A Survey of Secure Wireless Ad Hoc Routing," *IEEE Security and Privacy*, Volume 02, Issue 3, May -June 2004, Pages(s): 28-39.

[36] http://ntrg.cs.tcd.ie/ argp/public/adhoc-secure-routing-slides.pdf

[37] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields and E M.Belding Royer, "A secure routing protocol for ad hoc networks," *Proceedings 10th IEEE International Conference on Network Protocols*, 2002.

[38] J. Kong, X. Hong and M. Gerla, " A new set of passive routing attacks in mobile ad hoc networks," *In proceedings of the IEEE Militray Communications Conference MILCOM'03*, 2003.

[39] Y-C Hu, A. Perrig, D.B. Johnson, "Rushing Attacks in Wireless Ad Hoc Network Routing Protocols," *WiSe 2003*, San Diego, CA, September 2003.

[40] http://www.bee.net/mhendry/vrml/library/cdma/cdma.htm

[41] V. Park, S. Corson, " Temporally-Ordered Routing Algorithm (TORA) Version 1," *Functional Specification, Internet Draft*, IETF MANET Working Group, June 2001,

[42] S. Papademetriou, P. Papadopoulos, V. Park, A. Qayyum, "An Internet MANET Encapsulation Protocol (IMEP) Specification," *Internet Draft*, August 1999.

[43] J. Kim and G. Tsudik, " SRDP: Securing Route Discovery in DSR," *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2005*, 17-21 july 2005.

[44] Y-C Hu, D. B. Johnson and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad hoc Networks," *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, 2002.

[45] E. M. Royer, S. Ju- LEE and C. E. Perkins, " The Effects of MAC Protocols on Ad hoc Communication Protocols," *Proceedings of IEEE WCNC 2000*, chicago, IL, September 2000.

[46] R. R. Choudary, N. H. Vaidya, " MAC layer anycasting in ad hoc networks," *ACM SIGCOMM Computer Communication Review* volume 34, Issue 1, pp 75-80, jan 2004.

[47] S. Radosavac, N. Benjamin, J. S. Baras, " Cross layer attacks in Wireless ad hoc networks," *Conference on Information Sciences and Systems (CISS-04)* Princeton University, NJ 2004.

[48] I. D. Aron and S. K. S Gupta, " A Witness Aided Routing Protocol for Ad hoc Networks with Unidirectional Links," *First International Conference on Mobile Data Access*, LNCS 1748, pp 24-33, 1999.

[49] Y-B Ko, S-J Lee and J-B Lee, "Ad Hoc Routing with Early Unidirectionality Detection and Avoidance," Personal Wireless Communications, Springer, 2004.

[50] M. K. Marina, S. R. Das, " Routing Performance int he presence of Unidirectional Links in Multihop Wireless Networks," *Proceedings of the 3rd International Conference on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, jun 2002.

[51] R. Prakash, " A routing algorithm for wireless ad hoc networks with unidirectional links," *ACM/Kluwer Wireless Networks*, Vol 7, no.6, pp. 617-625.

[52] R. Prakash, " Unidirectional Links prove Costly in Wireless Ad Hoc networks," *Proceedings of the DIMACS Workshop on Mobile Networks and Computers*, pp 15-22, 1999.

[53] V. Ramasubramanian, R. Chandra, D. Mosse, " Providing Bidirectional Abstraction for unidirectional Ad Hoc Networks," *Proceedings of the 21st IEEE INFOCOM, June 2002*

[54] S. Nesargi and R. Prakash, " A Tunneling Approach to Routing with Unidirectional Links in Mobile Ad Hoc Networks," *Proceedings of the Ninth International Conference on Computer Communications and Networks*, 2000.

[55] A. Papoulis, and S. U. Pillai, *Probability , Random Variables and Stochastic Processes*, 4th Edition, 2001, McGraw Hill.

[56] http://www.isi.edu/nsnam/ns/

[57] P. Gupta and P. R. Kumar, " The capcity of Wireless networks," *IEEE Trans Inform Theory*, vol. 6 , no.2 , pp. 388404, Mar 2000.

[58] M. Ramkumar, "Trustworthy Computing Under Resource Constraints With the DOWN Policy," *IEEE Transactions on Dependable and Secure Computing,* Jan 2008.

[59] X. Du , Y. Wang and J. Ge, " A method for Security Enhancements in AODV protocol," *In proceedings of the 17th International Conference on Advanced Information Networking and Applications, AINA* 2003.

[60] M. Ramkumar, "An efficient broadcast authentication scheme for ad hoc routing protocols," *IEEE ICC 2006*, Istanbul, Turkey, Jun 2006.

[61] M. Ramkumar, "Broadcast Authentication With Hashed Random Preloaded Subsets," *Cryptology ePrint Archive*, May 2005.

[62] M. Bohio and A. Miri, " An Authenticated Broadcasting Scheme for Wireless Ad hoc Network," *Second Annual Conference on Communication Networks and Services Research CNSR 04*, pp69-74, 2004

[63] Y. Zhang, W. Liu, W. Lou, Y. Fang, " AC-PKI: anonymous and certificates public key certificateless public key infrastructure for mobile ad hoc networks," *IEEE International Conference on Communications (ICC'05), Seoul, korea, May 2005.*

[64] A. Fiat, M. Noar, "Broadcast Encryption," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **773**, pp 480–491, 1994.

[65] D. Noar, M. Noar, J. Lotspiech, "Revocation and Tracing Routines for Stateless Receivers," Lecture Notes in Computer Science, Advances in Cryptology, Springer-Verlag, **2139**, 2001.

[66] M. Ramkumar, N. Memon, "HARPS: HAshed Random Preloaded Subset Key Distribution", *Cryptology ePrint Archive*, August 2003.

[67] M. Ramkumar, An efficient broadcast authentication scheme for ad hoc routing protocols," IEEE ICC 2006, Istanbul, Turkey, Jun 2006.

[68] M. Ramkumar, "Broadcast Authentication With Preferred Verifiers, International Journal of Network Security," pp 166-178, Vol 4(2), March 2007.

[69] M. Ramkumar, N. Memon, "A DRM Based on Renewable Broadcast Encryption," Visual Communications and Image Processing (VCIP), Beijing, China, July 2005.

[70] M. Ramkumar, " Securing ad hoc networks with "asymmetric" probabilistic key predistribution schemes," Seventh IEEE IA Workshop, West Point, NY, Jun 2006.

[71] M. Ramkumar, "On the Scalability of a "Nonscalable" Key Distribution Scheme," IEEE SPAWN 2008, Newport Beach, CA, June 2008.

[72] S Marti, T. J. Giuli, Kevin Lai and Mary Baker,"Mitigating routing misbehavior in mobile ad hoc networks," Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston,2000.

[73] M. Ramkumar, "On the Scalability of a "Nonscalable" Key Distribution Scheme," IEEE SPAWN 2008, Newport Beach, CA, June 2008.

[74] T. Leighton and S. Micali, "Secret Key Agreement without Public -Key Cryptography," *Advances in Cryptology- CRYPTO 1993*, pp 456-479, 1994.

[75] M. Ramkumar, "On the Complexity of Probabilistic Key Predistribution Schemes" IEEE Embedded Systems and Communication Security (ECDS) Workshop, Niagara Falls, NY, Sep 2009.

[76] Y-C Hu, A. Perrig, D.B. Johnson, "Efficient Security Mechanisms for Routing Protocols," Symposium on Networks and Distributed Systems Security (NDSS), 2003.

[77] R.C. Merkle, "A digital Signature based on Conventional Encryption Function," Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, Lecture Notes In Computer Science; **293**, pp 369 – 378, 1987.

[78] M. Ramkumar, "Broadcast Encryption with Random Key Pre-distribution Schemes," Cryptology ePrint Archive, May 2005.